# Neural Network Control and An Optoelectronic Implementation of A Multilayer Feedforward Neural Network

Thesis by

Alan Akihiro Yamamura

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1992

(Submitted November 22, 1991)

*This thesis is dedicated*

*to the memory of my grandparents,*

*Munetsugu Yamamura,*

*and Harue and Sadao Nakamoto.*

## Acknowledgement

I would like to thank my thesis advisor, Professor Demetri Psaltis, for all his encouragement, advice, and friendship. He set an example of enthusiasm and creativity that I can only hope to someday match. Although he demanded the highest of standards at work, he showed great care and understanding towards me as well.

I would like to acknowledge the help of my colleagues, Professor Athanasios Sideris from whom I learned all about control theory, Dr. Chuanyi Ji for her help with generalization in neural networks, and my friends, Dr. Mark Neifeld and Seiji Kobayashi, with whom I had the great pleasure to work on optical disks (without Seiji, the optical disk work would not have been possible).

I would like especially to thank Dr. Jeffrey Yu and Dr. Fai Mok for "taking me under the wing" when I first arrived at Caltech. Their friendship and advice were invaluable.

I would like to thank my friends and colleagues, Dr. Nabeel Riza, Dr. David Brady, Dr. John Hong, Dr. Ken Hsu, Dr. Scott Hudson, Dr. Claire Gu, Dr. Cheol-Hoon Park, Dr. Steven Lin, Sidney Li, Subrata Rakshit, Yong Qiao, Kevin Curtis, Annette Grot, Robert Denkewalter, Dr. Amir Atiya, Dr. Ruth Erlanson, Dr. Kelvin Wagner, Dr. Eung Gi Paek, Su McKinley, Helen Carrier, Charlie Stirk, Francis Ho, Dr. Gabriel Sirat, Dr. Eric Dufresne, Dr. Joseph Goodman, Fredda Psaltis, Cathy Hayes, Suk Wong, Tomoko Kobayashi, Bruce and Kathy Betts, Jennifer Joh, and Edward Vail.

I would like to gratefully acknowledge the John and Fannie Hertz Foundation for their generous support during my many years at Caltech.

Most importantly, I made it through because of the love and support of my family, my parents Robert and Mitsuko Yamamura, and my sister and brother,

Eirene and Eugene as well as my grandmother Sui and my many aunts, uncles, and cousins.

Last but not least, I would like to thank Sylvia Chin for all her love and support, and her mother, father, sister, and brother who already treat me as one of their own.

## ABSTRACT

Artificial neural networks are a computational paradigm inspired by biological neural systems. By modeling neural networks to a certain degree after their counterparts in nature, it is hoped that they can capture those aspects of biological neural systems that allow them to outperform more conventional processing systems in tasks such as motor control and pattern recognition. A brief overview of neural networks is provided in Item 1, concentrating on those aspects pertinent to the remainder of this thesis.

The application of neural networks to control is examined in Item 2. A general control system can be divided into feedforward and feedback components. Specifically, the use of neural networks in learning to generate the feedforward control signal for unknown, potentially nonlinear, plants is examined. A class of learning algorithms applicable to feedforward networks is developed, and their use in learning to control a simulated two-link robotic manipulator is studied

An optoelectronic implementation of a multilayer feedforward neural network, with binary weights and connections, is described in the final part of this thesis. The neurons and connections are implemented electronically on a custom VLSI chip. The pattern and strength of the connections is controlled, through photodetectors placed in the connections, by a pattern of light illuminating the chip. This pattern is read out, in parallel, from an optical disk. Issues concerning parallel readout of information from optical disks are discussed in Item 3, while Item 4 contains a description of both the design of the Optoelectronic Neural Network Chip (ONNC) and experiments involving the optical disk and neural network chip.

## Table of Contents

**Abstract** ..... vi

**Table of Contents** ..... vii

**List of Figures** ..... ix

**List of Tables** ..... xv

**1 Computation and Artificial Neural Networks** ..... 1

1.1 Introduction ..... 1

1.2 Computational Power and Computer Architecture ..... 2

1.3 Programming 6

1.4 Biological Neural Networks ..... 7

1.5 Artificial Neural Networks ..... 8

**References** ..... 23

**2 Neural Network Control** ..... 25

2.1 Introduction ..... 25

2.2 Feedforward Control and Functional Inversion ..... 38

2.3 Experimental Results ..... 47

2.4 Future Directions ..... 77

**References** ..... 80

**3 Parallel Readout from Optical Disks** ..... 82

3.1 Introduction ..... 82

3.2 Characterization of the SONY Sampled-Format Disk ..... 91

3.3 Parallel Readout Techniques ..... 108

3.4 Applications and Future Directions ..... 153

**References** ..... 157

**4 An Optoelectronic Multilayer Feedforward Neural Network** .....161

4.1 Introduction ..... 161

4.2 An Optoelectronic Multilayer Feedforward Neural Network .....166

4.3 Future Directions ..... 211

**References** .....217

**5 Conclusions** .....220

## List of Figures

1.1 Von Neumann Architecture ..... 2
1.2 Memory Hierarchy ..... 3
1.3 Multiprocessor Memory Organization ..... 5
1.4 Interprocessor Communications ..... 6
1.5 Neuron ..... 7
1.6 Perceptron ..... 10
1.7 XOR Problem ..... 12
1.8 Probability That a Given Dichotomy of $N$ Points in $d$ Dimensions Will Be Linearly Separable ..... 12
1.9 Multilayer Network Solution to XOR ..... 14
1.10 Multilayer Feedforward Neural Network ..... 15

2.1 General Plant ..... 26
2.2 General Control System ..... 27
2.3 Trajectory Planning and Control ..... 28
2.4 System Step Response ..... 29
2.5 Use of Probing Signals for Parameter Estimation ..... 37
2.6 Generalized Learning Architecture ..... 39
2.7 Specialized Learning: Unknown Target ..... 40
2.8 Specialized Learning with Feedback Controller ..... 41
2.9 Neural Model of the Plant ..... 42
2.10 Back Error Propagation Through the Plant ..... 44
2.11 Providing Input Dynamics Using a Tapped Delay Line ..... 45
2.12 Squared Error for Coordinate Transformation with Generalized Learning . 50
2.13 Squared Error for Coordinate Transformation After 1000 Iterations of BEPing Through the Plant ..... 51

2.14 Squared Error for Coordinate Transformation After 100 Iterations of Generalized Learning and 1000 Iterations of BEPing Through the Plant ..... 52
2.15 Squared Error for Coordinate Transformation After Switching Regions of Specialization ..... 52
2.16 Links and Joints ..... 53
2.17 One-Link Manipulator ..... 54
2.18 Squared Error for One-Link Manipulator Using Fixed Training Points .... 56
2.19 Squared Error for One-Link Manipulator Using Random Training Points ..58
2.20 Comparison of Squared Error for One-Link Manipulator using Fixed and Random Training Points ..... 59
2.21 Two Link Manipulator ..... 60
2.22 Learning on Specific Trajectory ..... 62
2.23 Random Learning Over All Space ..... 63
2.24 Random Learning Over All Space With a Split Network ..... 64
2.25 Random Learning with Three Layer (+1 Hardwired) ..... 65
2.26 Character Generation ..... 67
2.27 Right- and Left-Hand Degenerate Configurations ..... 68
2.28 Trajectory Modification ..... 69
2.29 Trajectory Modification Using Tapped Delay Line ..... 70
2.30 Feedforward Trained Handwritten Characters ..... 72
2.31 Incrementally Trained Handwritten Characters ..... 74
2.32 Test of Neural Network Controller Trained On-Line ..... 76
2.33 Generalization to Different Speeds ..... 77
2.34 Contribution of Dynamic Terms to Total Torque ..... 78
2.35 Generalization to Different Positions ..... 78
2.36 Incremental Learning ..... 79

3.1 Data Organization for Parallel Readout ..... 88
3.2 Across-Track Coherence ..... 89
3.3 Prototype Sony Sampled-Format Disk System ..... 91
3.4 Sony Write-Once Recording Medium ..... 93
3.5 Metal Alloy Composition After Exposure ..... 94
3.6 Index and Absorptivity Variations After Exposure ..... 94
3.7 Experimental Setup to Measure Optical Disk Reflectivity ..... 96
3.8 Magnetooptic Disk ..... 97
3.9 Pixel Locations on Sony Sampled-Format Disk ..... 97
3.10 Timing and Tracking Information ..... 99
3.11 Optical Disk as Linear System Followed By Hard Limiter ..... 100
3.12 MTF of Optical Disk ..... 101
3.13 Fizeau Interferometer ..... 102
3.14 Optical Flatness of Disks ..... 103
3.15 Amplitude and Magnitude Squared of Diffraction Envelope ..... 105
3.16 $L = 2$ Diffraction Gratings Recorded on the Disk ..... 107
3.17 Expected and Observed Diffraction Patterns for $L$=2 ..... 109
3.18 Parallel Readout Techniques ..... 112
3.19 Schlieren Imaging ..... 114
3.20 Imaging the First Diffracted Order ..... 114
3.21 Image Contrast Enhancement by Imaging a Single Diffracted Order ..... 115
3.22 Superpixels for Grayscale Recording ..... 116
3.23 Pixel Averaging Over Superpixels ..... 117
3.24 Stochastic vs. Fixed Choice of Pixel Number ..... 117
3.25 Stochastic vs. Fixed Choice of Pixel Position ..... 118
3.26 Grayscale Image Recorded on Disk ..... 119
3.27 Modeled Pixel Shape ..... 120

3.28 Recording and Reconstructing an On-Axis Hologram ..... 124
3.29 Recording and Reconstruction an Off-Axis Hologram ..... 125
3.30 Recording and Reconstructing a Fourier Transform Hologram ..... 127
3.31 Lohmann Hologram ..... 129
3.32 Lee Hologram ..... 130
3.33 Iterative Quantization ..... 133
3.34 Reconstruction of Fresnel Hologram of Recorded on Write-Once Disk .... 137
3.35 Reconstruction of Fresnel Hologram Recorded on Magnetooptic Disk .... 137
3.36 Reconstruction of Fourier Hologram Recorded on Write-Once Disk ..... 138
3.37 Fourier Hologram Superpixel ..... 138
3.38 Reconstruction of Kobayashi Hologram ..... 139
3.39 Optical Disk - Neural Network Chip Misalignment ..... 141
3.40 Alignment Considerations for Imaging Readout ..... 141
3.41 Alignment Considerations for Fresnel Holographic Readout ..... 143
3.42 Alignment Considerations for Fourier Holographic Readout ..... 144
3.43 Effect of Disk Rotation on Fourier Hologram Reconstruction ..... 145
3.44 Rotation Angle of Reconstruction vs. Disk Rotation ..... 146
3.45 Coordinate Transformation Axes ..... 150
3.46 Optical Disk Correlator ..... 154
3.47 Optical Database Preprocessor ..... 155
3.48 Optical Disk Storage and Implementation of Neural Connections ..... 155
3.49 Volume Holographic Storage in Optical Disks ..... 156

4.1 Optical Disk/Chip Architecture ..... 167
4.2 Crossbar Configuration of Neurons and Synapses ..... 167
4.3 Available Components ..... 170
4.4 Circuit Diagram of Synapse ..... 171

4.5 Layout of Synapse ..... 172
4.6 Photograph of Synapse ..... 172
4.7 $V_g$ vs. $I_p$ for PFET and NFET pulldowns ..... 174
4.8 Synapse Conductance vs $V_g$ and $V_{in}$ ..... 178
4.9 Neuron Circuit Diagram ..... 180
4.10 Pads ..... 181
4.11 The ONNC ..... 182
4.12 Pinout of ONNC ..... 182
4.13 Measurement of Synapse Switching Time ..... 184
4.14 Synapse Switching Times ..... 185
4.15 Synapse Switching Time vs. Incident Intensity ..... 187
4.16 Synapse Reset Time ..... 189
4.17 Expected Synapse Current vs $V_0$ ..... 190
4.18 Measured Synapse Current vs $V_0$ ..... 191
4.19 Measurement of $V_{ref}$ ..... 192
4.20 Synapse Leakage Current ..... 193
4.21 Neuron Input Voltage vs +1 Input Position ..... 194
4.22 Neuron Input Voltage vs -1 Input Position ..... 195
4.23 Parasitic Resistance in the ONNC ..... 196
4.24 Foil Mask-ONNC Setup ..... 196
4.25 Foil mask in Optical System ..... 197
4.26 Microscope Alignment of Mask Pattern and Chip ..... 197
4.27 Neuron Input Voltage ..... 199
4.28 Neuron Switching Time ..... 204
4.29 Recall of Second Vector Heteroassociation ..... 206
4.30 Recall of Vector Heteroassociations ..... 207
4.31 Optical Disk-ONNC Setup ..... 208

4.32 Photograph of Optical Disk-ONNC Setup ..... 208

4.33 Microscope Alignment of Disk and Chip ..... 209

4.34 Imaging-Mode Alignment of Disk and Chip ..... 210

4.35 Recall of Vector Autoassociations ..... 212

4.36 Alternative Optoelectronic Synapse Designs ..... 213

4.37 Bipolar Synapses with 5-Bits of Dynamic Range ..... 215

4.38 Photograph of 5-Bit Bipolar Synapse ..... 214

4.39 Optical Disk Implementation of Neural Connections ..... 216

## List of Tables

1.1 Truth Table for Exclusive-Or ..... 11

2.1 One-Link Error after Learning on 5×5 Grid ..... 56
2.2 One-Link Error after Learning on 10×10 Grid ..... 56
2.3 One-Link Error after Random Learning ..... 57
2.4 Generalization vs. Number of Hidden Units ..... 73

3.1 Index and Absorption of Write-Once Disk Materials ..... 93
3.2 Predicted and Measured Disk Diffraction Efficiencies for $L$=2 Alternating Track Grating ..... 111
3.3 Predicted and Measured Disk Diffraction Efficiencies for $L$=10 Alternating Track Grating ..... 111
3.4 Misalignment in Parallel Readout From Optical Disks ..... 146
3.5 Dwell Time for Parallel Readout ..... 147
3.6 Translational Misalignment Speed for Parallel Readout ..... 148

4.1 Switching Time and Energy ..... 183
4.2 Synaptic Node Capacitance Parameters ..... 188
4.3 Neuron Switching Error ..... 202
4.4 Vector Heteroassociations ..... 205
4.5 Heteroassociative Connection Matrices ..... 205
4.6 Vector Autoassociations ..... 211
4.7 Weights ..... 212

# 1 Computation and Artificial Neural Networks

## 1.1 Introduction

Computers are everywhere in our society; in our homes and offices, our transportation, news, and entertainment, *etc.* Most of the computers that are a part of our everyday lives fall into a class of digital electronic computers with a Von Neumann-type architecture. By digital electronic, we mean that they represent information as strings of 1s and 0s and perform computation by storing and moving electrons within their components. By Von Neumann, we mean that they contain a single processing element connected to a memory bank as shown in Fig. 1.1. A processor in a Von Neumann machine accesses data in memory, performs some operation on the data, and stores the result back into memory. To a lay person, it may seem quite remarkable that machines based on such a simple model can be used to perform such complex tasks as controlling a nuclear power plant or animating a movie; however, getting a computer to perform these tasks is simply a matter of breaking a complicated process into successively smaller pieces that can eventually be accomplished by a sequence of operations using a single processor and memory.

Artificial neural networks represent an alternative computing paradigm (Sec. 1.5) that have been studied for a number of years. In the 1950s, Rosenblatt introduced his perceptron model (Subsec. 1.5.3). However, it was shown that the perceptron was incapable of solving the parity problem–a problem easily solved algorithmically.

More recently, there has been a revival of interest in neural networks. This

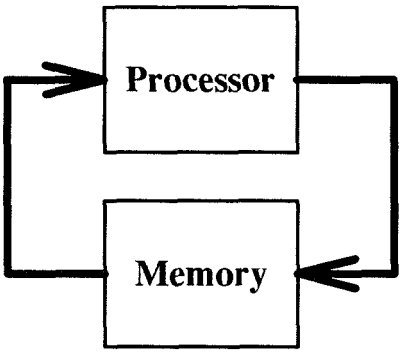



Fig. 1.1 - Von Neumann Architecture

increased interest has been sparked in part by new neural network models, new training algorithms, and a new theoretical understanding of neural networks. In this chapter, we begin by placing into perspective, the prospective benefits of neural network research on computer architecture, computer "programming," and our understanding of biological systems. We conclude by reviewing the development of artificial neural networks as it relates to the rest of this thesis.

## 1.2 Computational Power and Computer Architecture

The continuously increasing complexity of the tasks that computers are asked to perform has fueled the demand for increasing computational power, while increasing computational power opens the way for even more complex tasks which we ask them to perform. Various techniques some evolutionary, some revolutionary as described below are being pursued to provide greater computational power.

### 1.2.1 Incremental Increases in Performance

Until recently, the demand for increased computational power has been met by the incremental development of both faster processors capable of performing more complex instructions and ever larger and faster memories. As the minimum feature size (the smallest wire we can fabricate on the chip) has steadily decreased, the number of transistors we can pack on a chip has increased while the clock speed has increased as well. However, we will eventually reach physical limits on the minimum

size of wires and transistors on chips.

### 1.2.2 Cache Memories, Pipelining, and RISC Machines

Most recently however, more radical approaches have been adopted to provide even greater gains in computational power. The organization of memory has also been affected by splitting it into a pyramidal hierarchy (Fig. 1.2) with a small number of high-speed registers built into the processor, one or more levels of fast cache memory, a large store of slower dynamic memory, and finally low-speed mass memory. By keeping the most used pieces of information "closer" to the processor in higher-speed memory, the amount of time spent waiting to retrieve information can be reduced. Less used pieces of information are not placed in high-speed memory partly because of the increased cost of high-speed components and partly because the actual speed of high-speed memory decreases as the size of the memory increases.

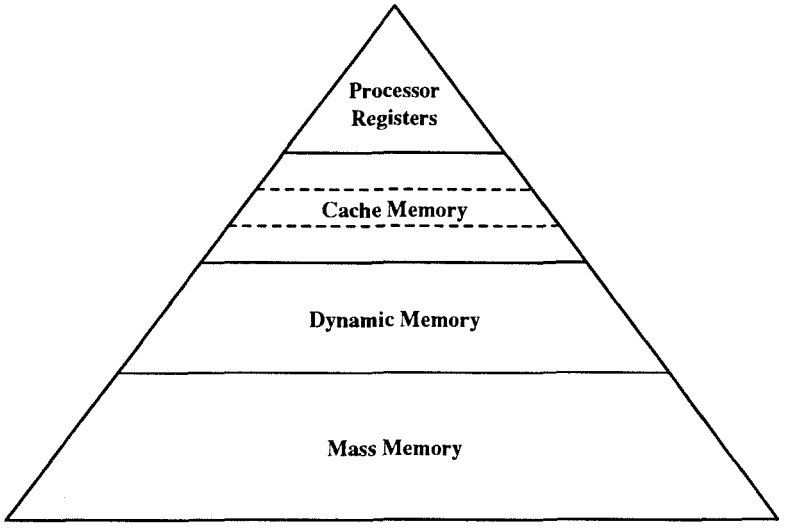



Fig. 1.2 - Memory Hierarchy

Because the time it takes to fetch an instruction or piece of data from memory can be significant, pipelining (typically in the form of prefetching instructions and data) has become popular. In pipelining, the processor predicts what instructions and data will be required and requests them from memory ahead of time and places

them in a "pipeline" so that they will be available when needed. If the incorrect instruction or data is prefetched (for instance because of a branch), the pipeline must be cleared and the correct instruction and data fetched before continuing.

It has also been determined that by reducing the complexity of instructions that the processor can perform, its speed can be increased enough to offset the reduction of instruction complexity. This has led to the development of so-called *Reduced Instruction Set Computers* or RISC machines.

### 1.2.3 Parallel Processing

Perhaps the greatest gains in computational power are to be reaped by abandonment of the single-processor Von Neumann architecture in favor of multiprocessor architectures. Ideally, increasing the number of processors should result in a proportional increase in computing power. Unfortunately, many questions must be answered before a parallel processing computer can be designed. One important question is how to divide the task into pieces that can be performed in parallel. If the pieces must be performed serially, no actual gain in computational power will be realized. We must also ensure that no processor will have to wait long periods of time for another processor to provide it with intermediate results.

Another question is whether the processors will operate synchronously with a global clock or asynchronously with respect to one another. In the latter case, interprocessor communications become more difficult while in the former we must develop techniques to broadcast the clock so that clock-skew (variations in the time it takes the clock signal to reach a processor) does not become a problem. If the processors act synchronously, certain applications may allow us to operate them in lock-step using the same instruction at each clock cycle. In this case, we have a *Single Instruction Multiple Data* or SIMD machine; if they use different instructions, it is a *Multiple Instruction Multiple Data* or MIMD Machine.

Another question is how to organize the memory; if all processors share the same memory through a single port (Fig. 1.3a), there will be contention between them over use of the memory and little gain will be achieved. On the other hand, if each processor has either its own port to main memory (Fig. 1.3b) or its own local memory (Fig. 1.3c), we must develop means to maintain data integrity–*i.e.*, prevent one processor from changing the value of a piece of data while another processor is using the old value.

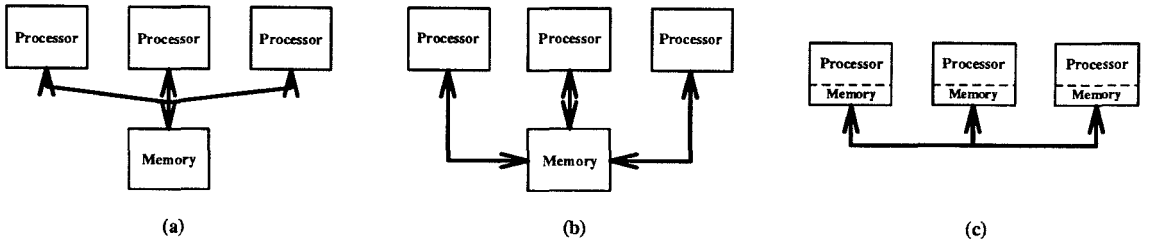



Fig. 1.3 - Multiprocessor Memory Organization
(a) Single-, (b) Multi-port, or (c) Distributed

As we increase the number of processors, communications becomes an increasing problem. If every processor can communicate directly with every other one (Fig. 1.4a), we have a virtual explosion in the amount of hardware that must be dedicated to communications. On the other hand, if we decide to restrict the communications, we must determine how this will be done. In a hypercube architecture (Fig. 1.4b), we place each processor at the corner of a hypercube and allow it to communicate directly only along the edges of the cube. In a systolic array (Fig. 1.4c), we place the processors in a grid and allow it to communicate only with its nearest neighbors.

These questions have been answered in one way or another in the multiprocessor and multicomputing systems commercially available today. These systems vary from those like the Cray Y-MP with eight powerful processors to the Connection Machine[1] with 65,636 one-bit SIMD processors. As we shall see, neural networks

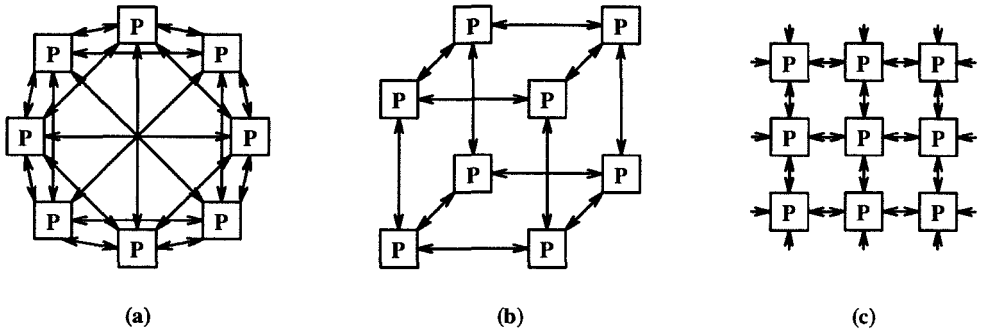



Fig. 1.4 - Interprocessor Communications: (a) Fully Interconnected, (b) Hypercube, and (c) Systolic Array

provide yet another set of answers to the questions above, that though closer to the Connection Machine in appearance, are actually quite different in spirit. The reason for the large variety in parallel processing architectures is that there is simply no single best architecture; the optimal tradeoff between cost and performance as well as raw performance itself depends to a great extent on the application.

## 1.3 Programming

No matter how powerful the computer, it cannot solve a single problem without the proper programming. As long as a problem and its solution are well understood, no matter how complex, it should be possible to write a program to implement the solution. Thus, we find computers solving large systems of equations or controlling complicated systems like the space shuttle about which we have adequate models.

However, there are tasks that we would like to perform with computers that we have not yet been able to do. We are less interested in those tasks which have an algorithmic or software solution but currently lack the hardware to solve the problem given real-world constraints. We are more interested in those problems for which we do not have good algorithmic or software solutions.

Pattern recognition, machine vision, and the like are problems whose solutions are still not well understood. Some researchers look to Artificial Intelligence (AI) techniques to attack these types of problems. Expert-systems that incorporate the

knowledge of so-called human experts in a rule-based approach to processing the given information are one solution typical of AI.

However, because problems like pattern recognition and machine vision are routinely solved in nature by some of the simplest life forms, some researchers look to biology to provide a solution. By modeling parts of our solutions on our understanding of biological systems, we hope to capture those aspects of biological neural computation that lend themselves to solving these types of problems.

## 1.4 Biological Neural Networks

The basic unit in a biological neural network is the neuron (Fig. 1.5). The neuron is a highly specialized cell. Typically, a neuron will have many protrusions, one of which is called an axon. Electrochemical signals can propagate down the axon by the motion of ions into and out of the cell membrane. These signals typically take the form of an electrical pulse train.

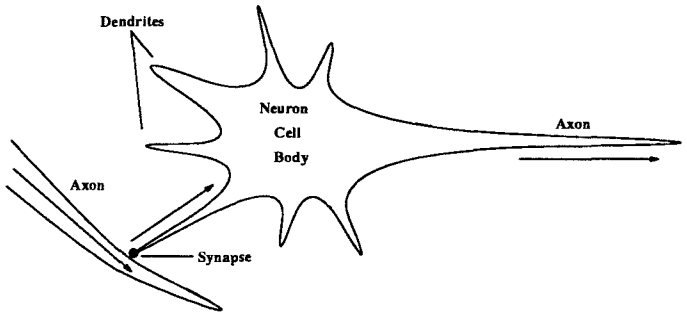



Fig. 1.5 - Neuron

The other protrusions, called dendrites, often connect to the axons of other neurons at points called synapses. A given synapse then connects the axon of the presynaptic neuron to the dendrite of the postsynaptic neuron. A passing electrical pulse stimulates a reaction at the synapse which results in a change in the post-synaptic neuron. A typical neuron can connect to between 1000 and 10000 other neurons. The frequency and pattern of pulses emitted by a neuron depend on the

type of the neuron and the stimulus that it receives through its many synaptic connections to other neurons.

There have been many different types of neurons that have been identified in the nervous system. The structure of neurons and their connections is known in simple networks such as the central pattern generators in lower life forms[2] as well as in highly regular networks such as the early vision layers in the visual system. By developing artificial neural network models, those studying them hope to gain a better understanding of how the biological neural networks work.

### 1.5 Artificial Neural Networks

One may quibble over the definition of the term "neural network." However, we will define them as any computing technique, implemented in hardware or software, that is in part inspired by our understanding of the central nervous system and other biological neural systems. Although the above definition is rather loose, we generally expect to find the following characteristics in artificial neural networks:

- Large number of processing elements
- Large number of connections
- Functionality determined by pattern and strength of connections

Many artificial neural networks also have very simple processing elements, typically single-input single-output and a thresholding transfer function. However, those networks that more closely follow biological models can have a significant degree of complexity.

Research into artificial neural networks is highly interdisciplinary. The study of these networks can lend themselves to the solution of a whole host of problems. For example, they can provide a model for the organization of processors and communications in a parallel processing architecture. They can inspire techniques for attacking problems such as pattern recognition and machine vision that,

though seemingly solved with ease in nature, remain intractable using conventional algorithms running on even the most powerful general purpose computers available today. Finally, they can help us understand the design and operation of the actual biological systems on which they are modeled. In the remainder of this chapter, we review the development of artificial neural networks as it relates to the rest of the thesis.

### 1.5.1 The McCulloch-Pitts Neuron

McCulloch and Pitts studied the behavior of neurons in squids. They noticed that pulses from a presynaptic neuron tended to stimulate or inhibit a postsynaptic neuron. Furthermore, if the level of stimulation exceeded a certain threshold, the postsynaptic neuron would begin to emit a train of pulses along its axon as well. They thus modeled the neuron as a single-output thresholding device.[3] If the stimulation from other neurons to which a given neuron connected exceeded a certain threshold, that neuron would generate an output that would stimulate other neurons to which it connected.

### 1.5.2 Hebb

Hebb modeled the input of the neuron as a weighted sum of the outputs of other neurons. He determined that information could be stored in a neural network within the strength and pattern of the connections between neurons.[4]

### 1.5.3 Perceptron

The perceptron model, developed by Rosenblatt,[5] consists of a single neuron implementing a hard threshold on the weighted sum of the perceptron inputs as shown in Fig. 1.6. We assume the following form for the transfer function of the perceptron:

$$y = \begin{cases} 1 & \underline{w}^T \underline{x} > 0 \\ 0 & \underline{w}^T \underline{x} < 0 \end{cases} \tag{1.1}$$

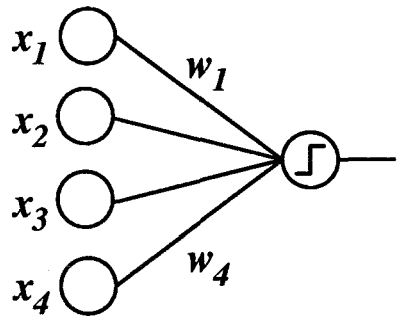



Fig. 1.6 - Perceptron

In order to train the perceptron, we begin with a training set, a set of ordered pairs of input vectors $x$ and their known associated outputs $y^d$: $\{(\underline{x}[1], y^d[1]), \ldots (\underline{x}[N], y^d[N]\}$. We then use the following algorithm:

1) Present the first training vector $\underline{x}^{(k=1)}$ to the input of the perceptron.

2) Modify the weight vector using the following equation:

$$\underline{w}[k+1] = \begin{cases} \underline{w}[k] + \alpha[k]\underline{x}[k] & \underline{w}^T\underline{x} < 0 \text{ and } y^d > 0 \\ \underline{w}[k] - \alpha[k]\underline{x}[k] & \underline{w}^T\underline{x} > 0 \text{ and } y^d < 0 \\ \underline{w}[k] & \text{otherwise.} \end{cases} \tag{1.2}$$

3) Go back to step 1. Repeat for all training vectors until we can correctly classify all vectors.

The key to this algorithm is in step 2. If we examine the change in the response of the presynaptic input to the training vector, we see that the weight vector is adjusted in a way that will drive the presynaptic input in the direction required to properly classify the training vector:

$$\underline{w}^T[k+1]\underline{x}[k] - \underline{w}^T[k]\underline{x}[k] = \begin{cases} \alpha[k]||\underline{x}[k]||^2 & y[k] = 0 \text{ and } y_d[k] = 1 \\ -\alpha[k]||\underline{x}[k]||^2 & y[k] = 1 \text{ and } y_d[k] = 0 \\ 0 & y[k] = y_d[k] \end{cases} \tag{1.3}$$

In the original perceptron algorithm, $\alpha[k]$=1. This does not guarantee that the training vector will be properly classified by the new weight vector, only that the presynaptic input will move in the desired direction. We could also dynamically adjust $\alpha[k]$, as shown below, so that the training vector will be properly classified by the new weight vector:

$$\alpha[k] > \frac{\underline{w}^T[k]\underline{x}[k]}{||\underline{x}[k]||^2} \tag{1.4}$$

Any adjustment to the weight vector to properly classify a given training vector can lead to misclassification of a training vector that was properly classified by the old. It has been shown, however, that as long as a weight vector exists that properly classifies the training vector, the perceptron will converge to a solution.

The perceptron classifies points by placing a hyperplane in input space that passes through the origin and is normal to the weight vector. If we provide the perceptron with an additional degree of freedom via an adjustable threshold, the hyperplane can be placed arbitrarily and no longer needs to pass through the origin.

Minsky demonstrated a weakness of perceptrons by showing that they could not solve the XOR or parity problem.[6] In this problem, we try to train a perceptron to implement the truth table shown in Table 1 known as the exclusive-or.

Table 1.1 - Truth Table for Exclusive-Or

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Although this problem is easily solved algorithmically, we can see graphically why a solution does not exist using the perceptron. Figure 1.7 shows the position of the training vectors in input space with dark circles for those with $y_d = 0$ and light circles for $y_d$=1. We see that it is impossible to place a single line that will separate the dark circles from the light.

This leads to an interesting question. Given $N$ randomly selected points in $d$ dimensions, what is the probability that they can be separated by a single $(d-1)$-dimensional hyperplane. The answer to this question has been derived and is given below:

$$P_{sol}(N \text{ pts in } d \text{ dims}) = \begin{cases} 2^{1-N} \sum_{i=0}^{d} \binom{N-1}{i} & N > d \\ 1 & N \leq d \end{cases} \tag{1.5}$$

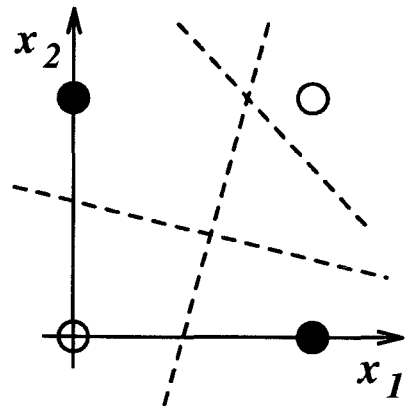



Fig. 1.7 - XOR Problem

Fig. 1.8 shows a plot of this function for $d$=3, 10, 25, and 100. Note that as d grows very large, there will almost always be a solution for $N < 2d$ and almost never a solution for $N > 2d$. Because the perceptron can always find a solution if one exists, we say its capacity $C = 2d$.

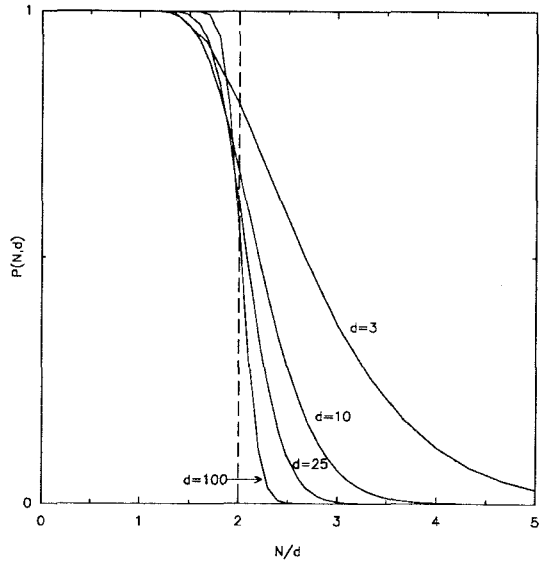



Fig. 1.8 - Probability That a Given Dichotomy of $N$ Points in $d$ Dimensions Will Be Linearly Separable

### 1.5.4 Binary Perceptron

Although we can speak theoretically of neural networks with analog connections of unlimited dynamic range, a real hardware implementation will have a limited dynamic range. To understand the consequences of a finite dynamic range, we can consider the limiting case of binary ($\pm 1$) connections. The binary perceptron is structurally identical to the regular perceptron (Fig. 1.6), but the weights are limited to values of $\pm 1$. Mok has proposed a training algorithm[7] for the binary perceptron that proceeds as follows:

1) Find an analog solution vector using the regular perceptron algorithm (Subsec. 1.5.3). Binarize the result by thresholding the weights.
2) Present a training vector.
3) If it is misclassified, search through all weight vectors within a Hamming distance of $d = 1$ (by randomly flipping a single bit) for a vector that classifies the input properly.
4) If no such solution exists, expand the search radius to $d = 2$.
5) Present the next training vector and repeat until all vectors are classified properly

### 1.5.5 Adaline

In the area of signal processing and adaptive filters, Widrow developed another learning rule for an architecture similar to that of the perceptron called the ADAptive LINEar device or adaline. In the adaline, we replace the nonlinear transfer function of the neuron depicted in Fig. 1.6 with a unity gain $y = \underline{w}^T \underline{x}$. We can derive the adaline learning rule by defining an error $\varepsilon$ equal to the squared error at the output of the adaline summed over the entire training set, which consists of ordered pairs of $\{(\underline{x}[n], y[n])\}$ of inputs $\underline{x}$ and corresponding outputs $y$:

$$\varepsilon = \frac{1}{2} \sum_n (y[n] - \underline{w}^T \underline{x}[n])^2 \tag{1.6}$$

We then apply the gradient descent algorithm to adjust the weight vector $\underline{w}$ in the local direction that leads to the greatest reduction of error (Eq. 1.7). Typically, gradient descent is approximated by cycling through the training vectors and adjusting the weights after each presentation (Eq. 1.9). The quantity $\rho$ in Eq. 1.9 represents the stepsize or learning rate of the adaline. As we increase $\rho$, learning proceeds at a faster rate, but the gradient is not followed precisely.

$$\Delta \underline{w} \propto -\nabla_{\underline{w}} \varepsilon \tag{1.7}$$

$$\propto \sum_{n} (y[n] - \underline{w}^T \underline{x}[n]) \underline{x}[n] \tag{1.8}$$

$$\underline{w}[k+1] = \underline{w}[k] + \rho(y[k] - \underline{w}^T[k]) \underline{x}[k] \tag{1.9}$$

### 1.5.6 Multilayer Feedforward Neural Networks

In Subsection 1.5.3, we encountered the XOR problem and found that the perceptron could not solve it. If we allow neurons to use the outputs of other neurons in a feedforward fashion, we find that the resulting networks can implement a larger variety of functions than the single perceptron. For example, the network shown in Fig. 1.9 is able to solve the XOR problem.

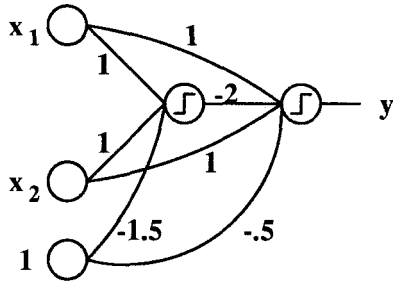



Fig. 1.9 - Multilayer Network Solution to XOR

A multilayer feedforward network is a class of neural networks in which neurons are grouped into layers. Neurons in each layer receive inputs from neurons in the previous layer only. Specifically, each neuron generates its output by applying a possibly nonlinear transfer function to the weighted sum of the outputs of neurons

in the previous layer. Figure 1.10 shows the structure of the multilayer feedforward neural network. Equations 1.10 and 1.11 describe the function of the network.

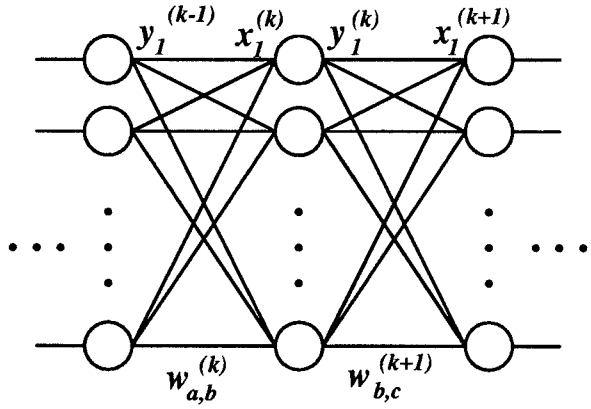



Fig. 1.10 - Multilayer Feedforward Neural Network

$$\underline{x}^{(k)} = \mathbf{W}^{(k)}\underline{y}^{(k-1)} \tag{1.10}$$

$$y_b^{(k)} = f_b^{(k)}(x_b^{(k)}) \tag{1.11}$$

Typically, we augment the neurons in each layer with a neuron whose output is always 1. The weights for this neuron allow us to shift the origin of the transfer function. We often use sigmoidal functions such as $f(x) = (1 + e^{-x})^{-1}$ for the transfer function of all the neurons, but different functions may be used for the output neurons in order to limit the range of the outputs. For example, if a given output is Boolean, we can give the corresponding neuron a hard-thresholding transfer function. If a given output is analog with an unknown range, we can give the corresponding neuron a linear transfer function.

The multilayer feedforward neural networks, then apply a series of alternating linear and possibly nonlinear transformations. As mentioned previously, these networks can implement a large variety of mappings. In fact, it has been shown that it is possible to learn any arbitrary mapping using a network of only two layers given a sufficient number of resources (neurons and connections).[8,9]

### 1.5.7 Back Error Propagation

A large number of training algorithms can be used to train multilayer networks to implement arbitrary mappings. Perhaps the most popular is the Back Error Propagation (BEP) algorithm.[10-13] The BEP connection update rule can be derived by applying a gradient descent search in the space of all possible weights to minimize the summed squared error at the output of the network (Eqs. 1.12,1.13). Equation 1.15 is the update rule for a connection in the $l^{th}$ layer.

$$\varepsilon = \frac{1}{2}||\underline{y}_d^{(L)} - \underline{y}^{(L)}||^2 \tag{1.12}$$

$$\Delta\underline{w} \propto -\nabla_{\underline{w}}\varepsilon \tag{1.13}$$

$$\propto (\nabla_{\underline{w}}\underline{y}^{(L)T})(\underline{y}_d^{(L)} - \underline{y}^{(L)}) \tag{1.14}$$

$$\Delta w_{ij}^{(l)} = \delta_i^{(l)} y_j^{(l-1)} \tag{1.15}$$

$$\delta_i^{(l)} = \begin{cases} (y_i^d - y_i^{(L)})\frac{df_i^{(L)}}{dx}|_{x_i^{(L)}} & l = L, \\ \left[\sum_k \delta_k^{(l+1)} w_{ki}^{(l+1)}\right]\frac{df_i^{(l)}}{dx}|_{x_i^{(l)}} & \text{otherwise.} \end{cases} \tag{1.16}$$

BEP has become popular because of its simplicity and its proven ability. However, because it is a gradient descent algorithm, BEP will always change the connections in the direction of greatest reduction of error, and stops adjusting the weights when it reaches a local minimum–where every direction leads to increasing error. There is however no guarantee that the error at this local minimum will be satisfactory. Another problem with BEP is that in order to follow the error gradient precisely, it is often necessary to take "small" steps in adjusting the weights and thus, a large number of total steps before reaching the local minimum.

### 1.5.8 Learning by Choice of Internal Representations

Grossman, Meir, and Domani proposed an alternative algorithm[14] for training two-layer networks by extension of the single-layer perceptron algorithm. They empirically found faster convergence than BEP. Their algorithm can be described

as follows:

1) Assign random initial weights to connections in the first layer.
2) Tabulate the internal representation (response of the hidden layer units) for the input training vectors.
3) Use the perceptron algorithm (Subsec. 1.5.4) to properly classify the internal representation. If the perceptron algorithm succeeds, we are done.
4) Else randomly flip bits in the internal representation and apply the perceptron algorithm until an internal representation for the inputs is found which can be properly classified.
5) Use the perceptron to train the first layer to generate the desired internal representation from the actual inputs. If the perceptron succeeds for all hidden units, we are done, else we stop and go back to step 2), tabulating the internal representation generated by the new first layer weights.

### 1.5.9 Training Binary Multilayer Networks

An algorithm for training binary multilayer networks can be constructed by replacing every use of the standard perceptron in Grossman, Meir, and Domani's algorithm (Subsec. 1.5.8) with the use of Mok's binary perceptron (Subsec. 1.5.5). Snapp has demonstrated successful application of this algorithm to two-layer binary networks.[15]

### 1.5.10 Generalization and the VC-Dimension

After a network has successfully learned to map a set of training vectors to their associated outputs, we would like it to be able to map inputs it has not seen to "reasonable" outputs. Note that we assume that there exists a compact representation of the underlying mapping. If such a representation does not exist, there is no way to learn the entire mapping except to memorize each input and its associated output. If a compact representation exists, we would like to say that a

network has memorized the mapping if it can correctly map the training vectors but fails to provide reasonable responses for untrained inputs. On the other hand, we would like to say that it has generalized if it provides reasonable responses for untrained inputs.

We can define generalization more precisely in classification problems by considering the VC-dimension. If we train a network with $M$ random training vectors from two classes, we can measure the classification error rate $\hat{r}(s)$ over the training set for the resulting network $s$ where $s \in S$, a class of networks of fixed structure consisting of threshold elements connected in a feedforward fashion. The probability that the true classification error rate $P(s)$ differs from the measured one by more than $\epsilon$ is bounded by the Vapnik-Chervonenkis[16] inequality:

$$Pr(\sup_{s \in S} |\hat{r}(s) - P(s)| > \epsilon) \leq 4\Phi(2M, d)e^{-\frac{\epsilon^2 M}{8}} = h(2M, d, \epsilon) \tag{1.17}$$

$$\Phi(2M, d) = \sum_{i=1}^{d} \binom{2M}{i} \tag{1.18}$$

where $d$ is the VC-dimension of $S$.

It can be shown that $h$ has a sharp transition at $M'$ about $O(d)$ for large $d$ such that $h \approx 0$ for $M > M'$ and that $|\hat{r}(s) - P(s)| \leq \epsilon$ with probability approaching 1 for all $s$ and $\epsilon$. Thus, if a network is able to load more than $M'$ training samples with $\hat{r}(s)$ small, it will most likely demonstrate small error or generalization to samples it has not seen.

Baum and Haussler have shown that the VC-dimension $d_{2r}$ for two layer feedforward networks with $N$ inputs, $L$ hidden units, 1 output, and a total of $W$ weights can be bounded as follows:[17]

$$O(W) \leq d_{2r} \leq O(W \log W) \tag{1.19}$$

Ji has shown that the VC-dimension for two-layer binary networks $d_b$ with $2L$ units

in the hidden layer can be bounded as follows:[18]

$$O\left(\frac{W}{\log L}\right) \leq d_b \leq O(W) \tag{1.20}$$

Thus, we find that there is not a significant loss in network performance for classification problems when switching from analog to binary connections.

### 1.5.11 Resource Minimization and Algo-A1

Because the VC-dimension of a network depends on the number of neurons and connections, it is important to select a network of appropriate size and structure in order to achieve generalization. Rumelhart proposed a method[19] for accomplishing this goal by adding a term of the following form:

$$\varepsilon_w = \sum_i \sigma(w_i) \tag{1.21}$$

$$\sigma(w) = \frac{w^2}{1+w^2}, \tag{1.22}$$

to the energy function on which we apply gradient descent. This additional term acts to reduce the magnitude of the weights. Weights with small magnitudes may be pruned when learning is complete.

Ji has developed a number of algorithms designed to dynamically adjust the number of neurons and connections. Algo-A1, one such algorithm, proceeds as follows:

1) We begin with a two-layer network with a single neuron in the hidden layer.
2) We train the network using BEP until a local minimum is reached. If the error is low enough, we go to step 5, otherwise we proceed to step 3.
3) We add a neuron to the hidden layer, "freeze" connections to the other hidden units, and train using BEP until a local minimum is reached. By "freeze," we mean that those weights are not allowed to change during training.

4) We then "thaw" the frozen neurons and continue training with all weights free to change until a local minimum is reached. If the error is small enough, we proceed to step 5. Otherwise, we return to step 3.

5) We continue training the network, but apply gradient descent on the following energy function to reduce the number of neurons and connections:

$$\varepsilon = \varepsilon_0 + \lambda\varepsilon_1 + \mu\varepsilon_2 \tag{1.23}$$

$$\varepsilon_0 = \sum_n ||\underline{y}_d[n] - \underline{y}^{(L)}[n]||^2 \tag{1.24}$$

$$\varepsilon_1 = \sum_i S_i \tag{1.25}$$

$$S_i = \sum_{jk} \sigma(w_{ij}^{(1)})\sigma(w_{ki}^{(2)}) \tag{1.26}$$

$$\varepsilon_2 = \sum_i \int \tanh(w_i)dw_i \tag{1.27}$$

$\varepsilon_2$ acts to reduce the magnitude of weights; it has a stronger effect on smaller weights. $S_i$ represents the "significance" of the $i^{th}$ hidden unit. This function is more sensitive to changes in the weights for neurons connected to small weights. Thus, $\varepsilon_1$ acts to reduce the weights of connections to "less significant" neurons. The second and third terms should only play a role once the error (measured by $\varepsilon_0$) has been reduced sufficiently. The following dependence of the regularizers $\lambda$ and $\mu$ on $\varepsilon_0$ was found to be effective in simulation:[20]

$$\lambda \propto e^{-\beta\varepsilon_0} \tag{1.28}$$

$$\mu \propto \frac{d\varepsilon_0}{dt} \tag{1.29}$$

Step 5) is itself another of Ji's algorithms called the Network Reduction Algorithm.

### 1.5.12 Incremental Learning

Experimentally, Ji has found that training takes an inordinate amount of time as the size of the problem grows. Thus it may prove easier to divide a problem into

smaller subtasks and build a network by training on each subtask. This procedure is implemented in the following incremental learning algorithm, Algo-I2:

1) Divide the training set into related groups called subtasks.

2) A network is trained using Algo-A1 to learn the first subtask.

3) Freeze the resulting network and use Algo-A1 to learn the next subtask.

4) Thaw the network and train on both tasks using the network reduction algorithm.

5) Repeat 3) and 4) for each remaining subtask.

### 1.5.13 The Hopfield Model

A large number of neural networks have been proposed (for a review see Lippman).[21] Some have supervised learning like BEP where connections are learned given a training set of inputs and outputs. Some have feedback (recurrent) and some do not (feedforward). Some, like the Boltzmann machine[22] include random noise that allows the network to temporarily increase its error in order to find potentially better solutions than the local minimum. Connections can also be learned in an unsupervised fashion as in Kohonen's self-organizing feature maps.[23]

Hopfield has proposed a class of networks[24] which is quite different in flavor from BEP on multilayer feedforward neural networks. In a Hopfield style network, the *a priori* knowledge of the desired solution of a problem is used to generate an energy function whose minimum corresponds to the desired solution. This energy can then be used to calculate the connections between neurons. These networks can be fully interconnected and generally have feedback. Different initial conditions or inputs result in the network settling to different stable states, each corresponding to a solution. For a Hopfield autoassociative memory for example, we wish the stable states to correspond to the stored vectors. We then set the weight $T_{ij}$ of the

connection between neurons $i$ and $j$ as follows:

$$T_{ij} = \begin{cases} \sum_n (2y_i[n] - 1)(2y_j[n] - 1) & i \neq j \\ 0 & i = j \end{cases} \tag{1.30}$$

Hopfield has shown that this type of network will converge if it is updated asynchronously.[24] Convergence depends only on the fact that the connection matrix $T_{ij}$ is symmetric.

## 1 References

[1] W.D. Hillis, *The Connection Machine*, The MIT Press, Cambridge, MA, 1985.

[2] A.I. Selverston and M. Moulins, "Oscillatory Neural Networks," *Ann. Rev. Physiol.*, Vol. 47, pp. 29-48, 1985.

[3] W.S. McCulloch and W. Pitts, "A Logical Calculus of the Ideas Imminent in Nervous Activity," *Bulletin of Mathematical Biophysics*, Vol. 5, pp. 115-133, 1943.

[4] D.O. Hebb, *The Organization of Behavior*, John Wiley & Sons, New York, 1975.

[5] Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automaton," *Cornell Aeronautical Laboratory Report*, 1956.

[6] M. Minsky and S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, 1969.

[7] F.H. Mok, *Binary Correlators for Optical Computing and Pattern Recognition*, Ph.D. Thesis, California Institute of Technology, 1989.

[8] G. Cybenko, "Approximations by Superpositions of A Sigmoidal Function," *Math. Control, Signals, Systems*, Vol. 2, pp. 303-314, 1989.

[9] K. Hornkik, M. Stinchcombe, and H. White, "Multi-Layer Feedforward Networks are Universal Approximators," Neural Networks, Vol. 2, pp. 359-366, 1989.

[10] P.J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in Behavioral Sciences*, Ph.D. dissertation, Harvard University, Cambridge, MA, 1974.

[11] D.B. Parker, "Learning Logic," *Technical Report TR-47*, Massachusetts Institute of Technology, Cambridge, MA, 1985.

[12] Y. le Cun, "Une procedure d'apprentissage pour reseau a sequil assymetrique," *Proceedings of Cognitiva*, Vol. 85, pp. 599-604, 1985.

[13] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, Foundations, D.E. Rumelhart and J.L. McClelland, eds., Cambridge, MA, MIT Press, 1986.

[14] T. Grossman, R. Meir, and E. Domani, "Learning By Choice of Internal Representation," Weizman Inst. of Sci., Rehovot, Israel, 1989.

[15] R. Snapp, personal communication, 1990.

[16] V.N. Vapnik and A.Y. Chervonenkis, "On Uniform Convergence of Relative Frequency of Event to Their Probabilities," *Theor. Prob. Appl.*, Vol. 16, pp. 264-280, 1971.

[17] E. Baum and D. Haussler, "What Size Net Gives Valid Generalization?" *Neural Computation*, Vol. 1, No. 1, pp. 151-160, 1989.

[18] C. Ji, *Generalization Capability of Neural Networks*, Ph.D. Dissertation, California Institute of Technology, 1991.

[19] D.E. Rumelhart, seminar at the California Institute of Technology, Spring 1987.

[20] C. Ji, R. Snapp, and D. Psaltis, "Generalizing Smoothness Constraints for Discrete Samples", *Neural Computation*, Vol. 2, pp. 188-197, 1990.

[21] R.P. Lippmann, "An Introduction to Computing with Neural Nets," *IEEE ASSP Magazine*, pp. 4-22, April 1987.

[22] D.E. Rumelhart and J.L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, 1986.

[23] T. Kohonen, *Self-Organization and Associative Memory,* Springer-Verlag, Berlin, 1984.

[24] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Natl. Acad. Sci.*, Vol. 81, pp. 2554-2558, April 1982.

# 2 Neural Network Control

## 2.1 Introduction

According to Webster's, *control* means, "...to have under command; to regulate, to check; to restrain; to direct...."[1] In a real world process such as driving a car, control of it implies that by adjusting certain *inputs* such as the steering wheel, brakes, and accelerator (aspects of the process that we can directly manipulate), we regulate or direct the *outputs* such as the position, speed, and direction of the car (aspects of the process that we cannot directly manipulate). An *automatic control system* is designed to adjust the inputs of the process such that the actual outputs are close or equal to desired outputs that are specified. In this chapter, we consider several techniques for utilizing neural networks in control systems.

### 2.1.1 The Control Problem

We call the process that we wish to control, the *plant* (as in a chemical plant). The plant has inputs $u$, outputs $y$, and internal state $x$ as shown in Fig. 2.1. The inputs are aspects of the process that we can directly manipulate. The outputs are the aspects of the process that we can measure. The state consists of those aspects of the process that at any given time are necessary to predict the future outputs of the system given the future inputs. We assume that the inputs, outputs, and state of the plant can be fully characterized by real vectors $\underline{u}$, $\underline{y}$, and $\underline{x}$. The behavior of the plant can then be modeled by a set of dynamic equations that we can typically

cast into the following form:

$$\dot{\underline{x}} = \underline{F}(\underline{u}, \underline{x}) \tag{2.1}$$

$$\underline{y} = \underline{G}(\underline{x}, \underline{u}) \tag{2.2}$$

where Eq. 2.1 is a differential equation driven by the inputs $\underline{u}$ describing the time evolution of the state $\underline{x}$, and Eq. 2.2 describes the output $\underline{y}$ as a function of the input and current state.

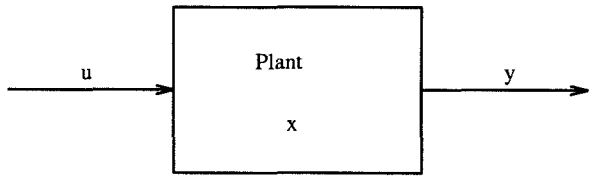



Fig. 2.1 - General Plant

A general control system takes as input the desired and actual outputs of the plant and generates a plant input designed to keep the output of the plant close or equal to the desired output. As shown in Fig. 2.2, a general control system may be divided into feedforward and feedback parts. The feedforward controller generates a component of the plant input $\underline{u}_{FF}$ based on the desired plant output only. This controller embodies the *a priori* understanding of the required plant input given the desired plant output. Ideally, (*i.e.*, given a perfect model of the plant, the absence of noise, *etc.*) feedforward control alone will suffice to keep the plant output exactly at its desired value. The feedback controller generates a component of the plant input $\underline{u}_{FB}$ using both the desired output and the actual output of the plant. Typically, the feedback controller takes as input only the difference between the desired and actual plant outputs (the plant output error). Given such an error, it contributes a correction term to the plant input that drives the plant output toward its desired value. Using feedback, it becomes possible to correct for a variety of nonidealities such as errors in the feedforward signal, noise, *etc.*.

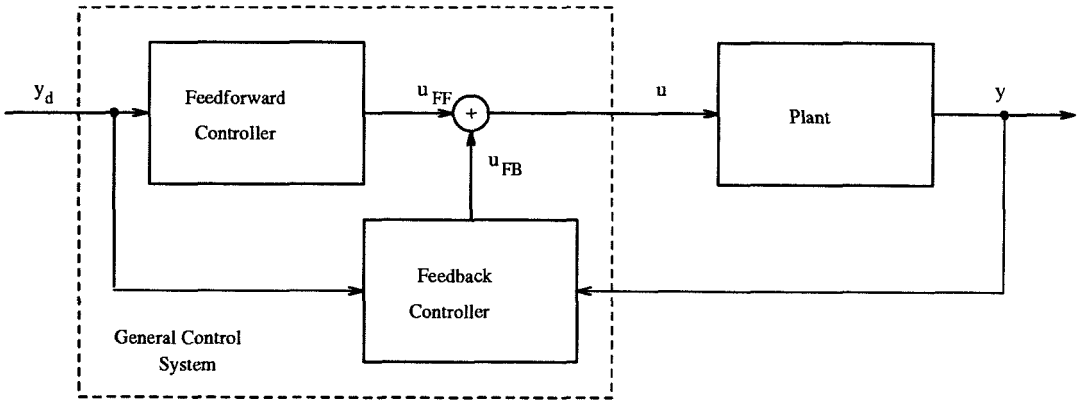



Fig. 2.2 - General Control System

When we use feedforward control alone without reference to the plant output, we call it *open-loop control.* As an example of open-loop control, consider the grilling of a steak. The desired output may be the doneness, *e.g.*, rare, well-done, *etc.*. The plant input may be the grilling time. If we simply throw the steak on the grill for a prescribed time, perhaps using our prior grilling experience, we are implementing a form of open-loop control. Open-loop control may be sufficient for some processes; however, in all but the simplest processes, incomplete knowledge about the process or the presence of noise results in unsatisfactory open-loop control of the output. Anytime we refer to the output of the plant in generating its input, we call it *closed-loop control.* Even in the steak-grilling example, one typically uses the appearance and smell of the steak, sometimes even cutting the steak open, to estimate the doneness of the steak–thus actually closing the control loop. For complex processes, control without reference to the output would be impossible.

The burden of control can be viewed as being shared by the feedforward and feedback controllers. If the feedforward controller is not well designed, the error at the output of the plant will tend to be larger and the feedback controller will be doing most of the work in reducing the output error. On the other hand, with a well-designed feedforward controller, the error at the output of the plant will tend to be smaller with the feedforward controller taking a greater share of the control

burden. Since the feedback controller is error-driven, one hopes that by better design of the feedforward controller, one can operate the plant with superior speed and accuracy.

In some cases, we may not know what plant outputs correspond to our desired goals and objectives. In these cases, we assume the presence of a *planner* as shown in Fig. 2.3. The planner takes the desired goals and objectives and translates them into a desired plant output. As shown in the figure, the planner may use the actual plant output to change or modify the desired plant output. Although planning is an important aspect of control, the focus of this chapter is on the control of the plant given a desired plant output.

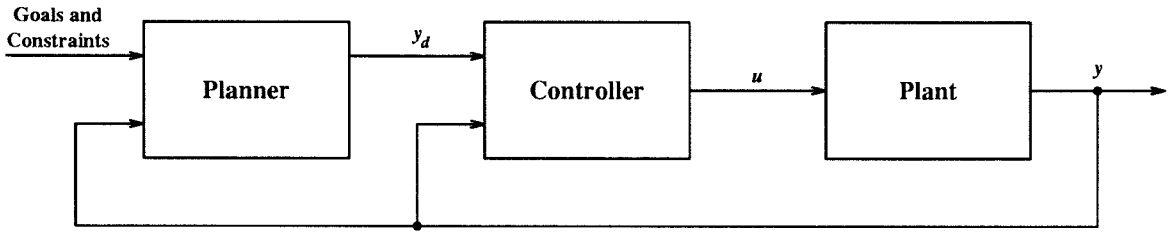



Fig. 2.3 - Trajectory Planning and Control

### 2.1.2 Conventional Control

Conventional control is a name applied to control theory prevalent in the United States and Western Europe through the 1950s. A conventional control system feeds back corrective terms, dependent on the plant output error, and adds them to the plant input.

The simplest form of linear feedback involves adding a term proportional to the plant output error to the input of the plant as follows:

$$u_{FB} = K(y_d - y) \tag{2.3}$$

A typical response to a step input for such a system might look like that shown in Fig. 2.4. One can also add a term to the plant input proportional to the time

integral of the output error. The magnitude of this integral feedback would grow in the presence of a steady-state error until the steady state error was reduced to zero. The addition of a term to the plant input proportional to the time derivative of the output error would act to damp the ringing in the response of the output to sudden changes. Different combinations of proportional, integral, and derivative feedback result in different control laws. The combination of all three is called *PID* control and has developed as an effective method for the control of many industrial processes. When using a PID controller, one must choose the coefficients weighting each term, shown in the following equation, so as to meet the specifications for the control system:

$$u_{FB} = K_P e + K_I \int_0^t e d\tau + K_D \dot{e} \tag{2.4}$$

$$e = y_d - y \tag{2.5}$$

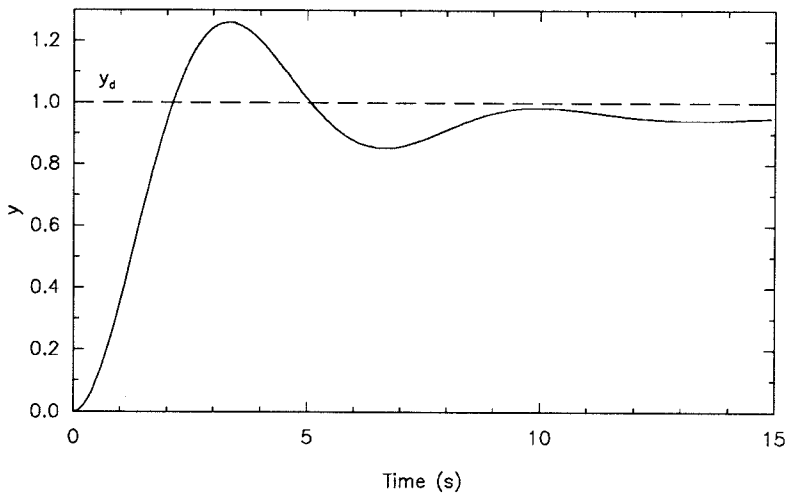



Fig. 2.4 - System Step Response

Conventional control emphasizes the use of frequency domain techniques to analyze the stability of control systems. The root-locus method, introduced in

1948 by W.R. Evans, became particularly important in the 1950s. In the root-locus method, a graph in the $s$-plane is produced tracing the path of the poles and zeroes of a system as a single parameter is varied. By observing the behavior of the poles and zeroes as a control parameter changes, one can then choose appropriate values for that parameter. Other frequency-domain techniques involve the use of Bode plots of the open-loop frequency response of the system in combination with the Nyquist stability criterion. The Nichols chart plotting the magnitude of the frequency response versus the phase is also useful in choosing appropriate values for parameters using the frequency domain.

Typically, one adjusts the behavior of the system by first adjusting proportional gain elements in the feedforward and/or feedback paths. However, the addition of dynamic elements (*compensation*) may be required to meet specified response characteristics. Compensation can be divided into *lead* and *lag* networks. Lead networks behave roughly like derivative feedback and act primarily to increase bandwidth and reduce rise time. Lag networks act roughly like integral feedback and raise low-frequency gain to reduce steady-state error.

Conventional control techniques are useful primarily in single-input single-output control systems because they utilize the frequency domain for parameter selection. Modern control techniques based on time domain formulation and analysis must be used for control of multivariable systems. There are many good references on conventional and modern control including textbooks by Franklin, Powell, and Emami-Naeini[2] and Ogata.[3]

### 2.1.3 Modern Control

While U.S. and Western European researchers concentrated on frequency domain techniques until the 1950s, Russian and Eastern European researchers concentrated on time domain techniques where system dynamics can be cast as a set of

ordinary differential equations and the behavior of systems analyzed in state-space. The ability to analyze multivariable control systems as well as the stability of non-linear systems are strong motivations for the development of modern state-space techniques. The desire to control systems optimally according to specified criteria also spurred the transition to state-space techniques since conditions for optimality are most often specified in the time-domain.

In a typical modern control system, the output of the controller is a linear combination of the desired plant output and the state of the plant. State feedback is used because the output of the plant depends on the input and state of the plant, so the required input should naturally depend on the desired output and state of the plant. If the state of the plant is unknown, an estimate of the state $\underline{\tilde{x}}$ is often used as shown in Eq. 2.6. This estimate often depends on the input, output, and current estimated state of the plant as shown in Eq. 2.7.

$$\underline{u} = \mathbf{A}\underline{y}_d + \mathbf{B}\underline{\tilde{x}} \tag{2.6}$$

$$\dot{\underline{\tilde{x}}} = \mathbf{C}\underline{\tilde{x}} + \mathbf{D}\underline{y} + \mathbf{E}\underline{y}_d \tag{2.7}$$

Lyapunov functions are often used in modern control for the stability analysis of control systems. A scalar function of system variables measuring a potential or energy, $V$, called a Lyapunov function is defined. Given a model of the system dynamics, one determines the stability of the system, by analyzing the behavior of $V$ and its time-derivative $\dot{V}$. Typically, the key to successful application of Lyapunov stability analysis is the choice of an appropriate Lyapunov function.

### 2.1.4 Nonlinear Control

Many of the real-world processes that we wish to control are nonlinear in nature. The techniques we have reviewed for controlling linear plants have generally been applied to nonlinear plants by linearizing the plant around an expected operating point. This technique has proven quite successful for processes with weak

nonlinearities or those in which the state does not deviate far from a given point during operation. However, nonlinear control laws must be used for processes with strong nonlinearities or where we wish to extend the region over which a controller can provide stable operation.

Fixed nonlinearities, whether intentional or otherwise, can be introduced into controllers for a variety of reasons. An example of an unintentional nonlinearity is a saturating transfer characteristic at the input or output. Intentional nonlinearities may be introduced to simplify or improve control. For example, if a nonlinear plant can be changed into a linear one using a nonlinear transformation, a linear control law can be designed using more conventional techniques in the transformed variables and changed back into a nonlinear one using the reverse transformation.

Another possibility for control of a nonlinear system is to change the parameters of a linear controller based on measurements of certain aspects of the process (typically the measured or estimated state). Although we might term any such system *adaptive*, we generally reserve this term for systems in which we modify controller parameters in a closed-loop fashion. Systems in which the parameters are dependent on state in an open-loop fashion are nonlinear but not "adaptive."

Gain scheduling is an example of the latter case. In gain scheduling, certain aspects of the process are used to delineate different operating regimes. Each operating regime is then assigned a set of controller gains designed to provided satisfactory operation in that regime. During operation, these aspects are monitored and the appropriate gains used in the controller.

Variable structure systems (VSS) are another form of a nonlinear control system where the controller changes are based on the state of the system being controlled. The name refers to the change in the structure of the controller as the state changes. Typically, a VSS controller is formulated as follows. We define a surface $\sigma(\underline{x}) = 0$ in state space on which we wish to operate. The control law switches at the surface

such that whether $\sigma(\underline{x})$ is greater or less than zero, the state is driven towards the surface $\sigma(\underline{x}) = 0$. Even if the state leaves the surface due to noise, deviations in the system, or intentional control inputs, the switching control laws drive it back to the surface. We then look for sliding modes–trajectories that remain on the surface. If these modes are stable, motion along the switching surface can be controlled and is insensitive to parameter variations of the plant.

The advantage of open-loop nonlinear control is that it is simpler to implement. The disadvantage is that we must precalculate the system gains, and if they prove to be wrong in operation for the given state, there is no way to adjust the input. For those nonlinear plants where it is difficult or impossible to precalculate desired gains, we must close the loop and provide full adaptive control. Once again, there are many good references for nonlinear and adaptive control including Astrom and Wittenmark.[4]

### 2.1.5 Adaptive Control

Any control system designed using only *a priori* knowledge may fail if the model is poor. However, in adaptive control, important parameters are adapted on-line while operating the actual plant. In model reference adaptive control, the controller is provided with a model of how we want the plant to respond. We then build a controller for the plant and modify the controller parameters continuously based on the measured or estimated state in order to reduce the difference between the model plant and the actual plant.

In self-tuning regulators, we measure or estimate plant parameters and at each stage redesign or update controller parameters to control the plant. In stochastic adaptive control, instead of using a single estimated state representing the maximum likelihood, we use the estimated probability distribution of the state as a *hyperstate* that is fed back in the control algorithm.

In many adaptive control algorithms, we must estimate plant parameters. There are a variety of standard techniques for making such an estimate given the inputs and outputs of the plant. The method of least squares can be used to adjust each estimated plant parameter such that the squared error between the plant model and the outputs is minimized. If we assume that we have a plant with inputs $\underline{u}$, outputs $\underline{y}$, and a plant model with parameters $\underline{\theta}$ and outputs $\underline{y}_p$, we find a set of plant parameters $\underline{\theta}_0$ such that the squared error $\varepsilon$ is minimized for either discrete (Eq. 2.9) or continuous (Eq. 2.11) time cases by setting to zero the gradient of the error with respect to the parameters yielding a set of equations in a like number of unknowns:

$$\nabla_{\underline{\theta}}\varepsilon|_{\underline{\theta}_0} = \underline{0} \tag{2.8}$$

$$\varepsilon_d = \sum_n^N ||\underline{y}[n] - \underline{y}_p[n]||^2 \tag{2.9}$$

$$\underline{0} = \sum_n^N \nabla_{\underline{\theta}} \underline{y}_m^T[n](\underline{y}[n] - \underline{y}_m[n]) \tag{2.10}$$

$$\varepsilon_c = \int_0^t ||\underline{y}(\tau) - \underline{y}_m(\tau)||^2 d\tau \tag{2.11}$$

$$\underline{0} = \int_0^t \nabla_{\underline{\theta}} \underline{y}^T(\tau)(\underline{y}(\tau) - \underline{y}_t au))d\tau \tag{2.12}$$

If the plant model is linear in the parameters, we can cast it in the following form:

$$\underline{y}_m = \Theta \underline{u} \tag{2.13}$$

Substituting back into Eqs. 2.10 and 2.12, we find the following expressions:

$$\Theta[N] = \left[\sum_n^N \underline{y}[n]\underline{u}^T[n]\right]\left[\sum_n^N \underline{u}[n]\underline{u}^T[n]\right]^{-1} \tag{2.14}$$

$$\Theta(t) = \left[\int_0^t \underline{y}(\tau)\underline{u}^T(\tau)d\tau\right]\left[\int_0^t \underline{u}(\tau)\underline{u}^T(\tau)d\tau\right]^{-1} \tag{2.15}$$

For the discrete case, we can reduce the number of calculations required by

recasting Eq. 2.14 in a recursive form as follows:

$$\mathbf{P}[N] \equiv \left[\sum_{n}^{N} \underline{u}[n]\underline{u}^T[n]\right]^{-1} \tag{2.16}$$

$$\sum_{n}^{N} \underline{y}[n]\underline{u}^T[n] = \Theta[N]\left[\sum_{n}^{N+1} \underline{u}[n]\underline{u}^T[n] - \underline{u}[N+1]\underline{u}^T[N+1]\right] \tag{2.17}$$

$$\Theta[N+1] = \left[\sum_{n}^{N} \underline{y}[n]\underline{u}^T[n] + \underline{y}[N+1]\underline{u}^T[N+1]\right]\mathbf{P}[N+1] \tag{2.18}$$

$$= \Theta[N] + \left(\underline{y}[N+1] - \Theta[N]\underline{u}[N+1]\right)\underline{u}^T[N+1]\mathbf{P}[N+1] \tag{2.19}$$

$$\mathbf{P}[N+1] = \left[\sum_{n}^{N} \underline{u}[n]\underline{u}^T[n] + \underline{u}[N+1]\underline{u}^T[N+1]\right]^{-1} \tag{2.20}$$

$$= \mathbf{P}[N] - \mathbf{P}[N]\underline{u}[N+1]\left[1 + \underline{u}^T[N+1]\mathbf{P}[N]\underline{u}[N+1]\right]^{-1}$$

$$\underline{u}^T[N+1]\mathbf{P}[N] \tag{2.21}$$

where Eq. 2.21 follows by inspection.

The nonlinear plant that we wish to control may be viewed as a linear plant with time-varying parameters. If the parameters change quickly, we can periodically reset our estimate of the plant parameters. For example in the recursive-least-squares algorithm, one would reset $\mathbf{P}$ to $\alpha\mathbf{I}$, with $\alpha$ a large constant, to "forget" input/output pairs corresponding to different operating regimes. If, however, the parameters change continuously but slowly, we may exponentially weight the importance of the input/output pairs using energy functions of the following form:

$$\epsilon_d = \sum_{n}^{N} \alpha^{(n-N)}||\underline{y}[n] - \underline{y}_m[n]||^2 \tag{2.22}$$

$$\epsilon_c = \int_0^t \alpha^{(\tau-t)}||\underline{y}(\tau) - \underline{y}_m(\tau)||^2 d\tau \tag{2.23}$$

yielding the following modifications to Eqs. 2.14, 2.15, and 2.21 respectively:

$$\Theta[N] = \left[\sum_{n}^{N} \alpha^{n-N}\underline{y}[n]\underline{u}^T[n]\right]\left[\sum_{n}^{N} \alpha^{n-N}\underline{u}[n]\underline{u}^T[n]\right]^{-1} \tag{2.24}$$

$$\Theta(t) = \left[\int_0^t \alpha^{\tau-t}\underline{y}(\tau)\underline{u}^T(\tau)d\tau\right]\left[\int_0^t \alpha^{\tau-t}\underline{u}(\tau)\underline{u}^T(\tau)d\tau\right]^{-1} \tag{2.25}$$

$$\mathbf{P}[N+1] = \alpha\Big\{\mathbf{P}[N] - \mathbf{P}[N]\underline{u}[N+1]\left[I/\alpha + \underline{u}^T[N+1]\mathbf{P}[N]\underline{u}[N+1]\right]^{-1} \underline{u}[N+1]\mathbf{P}[N]\Big\} \tag{2.26}$$

Even using the recursive least squares algorithm, we require a large number of calculations to maintain our estimates of plant parameters. Projection algorithms can be used to provide estimates of plant parameters using fewer calculations. For the discrete case, we add components of the plant inputs to the plant model parameters to move the model outputs toward the actual outputs:

$$\Theta[N+1] = \Theta[N] + \underline{\alpha u}^T[N+1] \tag{2.27}$$

If we choose $\alpha$ such that the most recent plant model output matches the plant output, we get Kaczmarz's projection algorithm:

$$\underline{y}[N+1] = \Theta[N+1]\underline{u}[N+1] \tag{2.28}$$

$$= \underline{y}_p[N+1] + \underline{\alpha}||\underline{u}||^2 \tag{2.29}$$

$$\underline{\alpha} = \frac{\underline{y}[N+1] - \underline{y}_p[N+1]}{||\underline{u}||^2} \tag{2.30}$$

$$\Theta[N+1] = \Theta[N] + \left(\underline{y}[N+1] - \Theta[N]\underline{u}[N+1]\right)\frac{\underline{u}^T}{||\underline{u}||^2} \tag{2.31}$$

We often modify Kaczmarz's projection algorithm as follows to prevent it from diverging for small $\underline{u}$:

$$\Theta[N+1] = \Theta[N] + \left(\underline{y}[N+1] - \Theta[N]\underline{u}[N+1]\right)\frac{\gamma\underline{u}^T}{\beta + ||\underline{u}||^2} \tag{2.32}$$

where $\beta > 0$ and $0 < \gamma < 2$.

The plant input must be chosen properly for any of these parameter estimation techniques to work properly. In adaptive control, we say that we want the input to

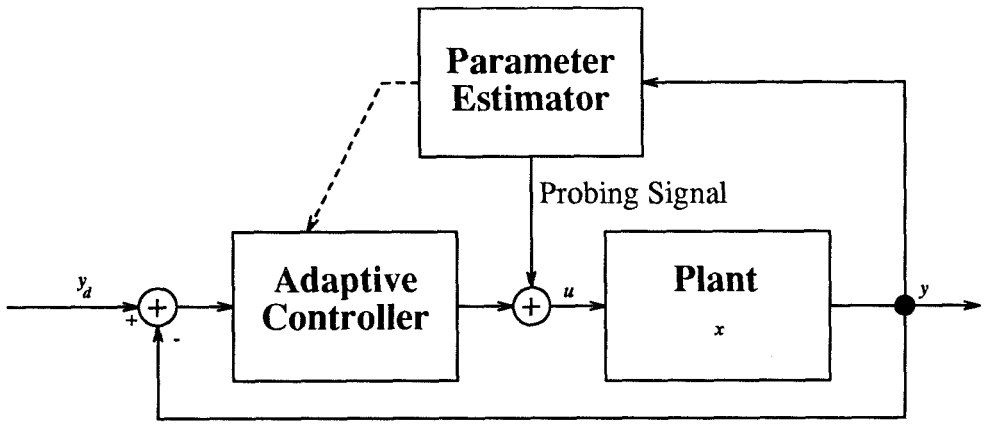



Fig. 2.5 - Use of Probing Signals for Parameter Estimation

be *persistently exciting.* It is also possible to impose probing signals on the input as in Fig. 2.5 and look for correlations at the output to measure plant parameters. Sinusoids and white-noise are often used as probing signals.

### 2.1.6 Neural Networks and Control

In Chapter 1, we defined the term *neural network* as it applies to this thesis and reviewed some important historical developments in the theory of neural networks. Of particular interest to this chapter was the fact that it is possible to implement any analog mapping using a two-layer feedforward network (Subsec. 1.5.6). In this chapter, we develop a method for training a neural network feedforward controller. We are motivated to do so for a number of reasons. Many systems that we wish to control are nonlinear in nature, and, as we saw, neural networks such as the multi-layer feedforward neural network, are capable of implementing any static mapping. Because many of these systems may be difficult to model, the ability of neural networks to learn mappings from examples may prove quite powerful for allowing us to control previously unmanageable systems or control systems with greater speed and accuracy than previously possible. Finally, by developing on-line learning techniques, we may design a network capable of controlling time-varying systems and adapting to changing system parameters that are difficult to quantify, such as those

due to wear-and-tear of machinery.

## 2.2 Feedforward Control and Functional Inversion

As described in Sec. 2.1, a general controller can be divided into feedforward and feedback components. The function implemented by the feedforward controller should be as close as possible to the inverse of the plant. Several proposed training methods were mentioned for finding both the connection pattern and weights between neurons and the network architecture itself. In these methods, sample ordered pairs of inputs and associated outputs (a training set) are used to train the neural networks. By simply switching inputs and outputs, one might expect that one could just as easily learn the inverse of the function described by the training set. However, when trying to control a plant which either is difficult to accurately model or about which we have little knowledge, a training set may not be available. Thus, it may be desirable to learn to control the plant through actual operation of the plant itself.

In this section, we consider a number of techniques, each based on extension of the standard BEP algorithm, for training multilayer feedforward neural networks to implement the inverse of a function through actual experimentation with the plant itself. The extensions to BEP described in this section could also be used to extend other algorithms using different architectures to learn inverse functions.

### 2.2.1 Generalized Learning

The most straightforward method for trying to train a feedforward controller would be based on the idea that the feedforward controller should be something like the inverse of the plant function. That is, given a plant performing the mapping of Eq. 2.33, we wish the feedforward controller to perform the mapping of Eq. 2.34.

$$\underline{y} = \underline{P}(\underline{u}; \underline{x}) \tag{2.33}$$

$$\underline{u} = \underline{P}^{-1}(\underline{y}; \underline{x}) \tag{2.34}$$

One way to accomplish this task would be to place the neural network controller after the plant during training as shown in Fig. 2.6. This is called the generalized learning architecture. We present training inputs $\underline{u}_b$ to the plant, feed the plant output $\underline{y}$ to the network, and measure the resulting network output $\underline{u}$. If we treat the plant inputs as the desired output of the network, each step in the training provides us with a network input $\underline{y}$, a network output $\underline{u}$, and a desired output $\underline{u}_b$. Back-error propagation, or another desired training algorithm, can then be used to modify the network to drive the output of the network towards the desired output. The goal of the generalized learning architecture is to train the network in as direct a manner as possible to implement the "inverse" of the plant. Generalized learning, also called inverse system identification or direct inverse learning, has been discussed by a number of authors.[5−10]

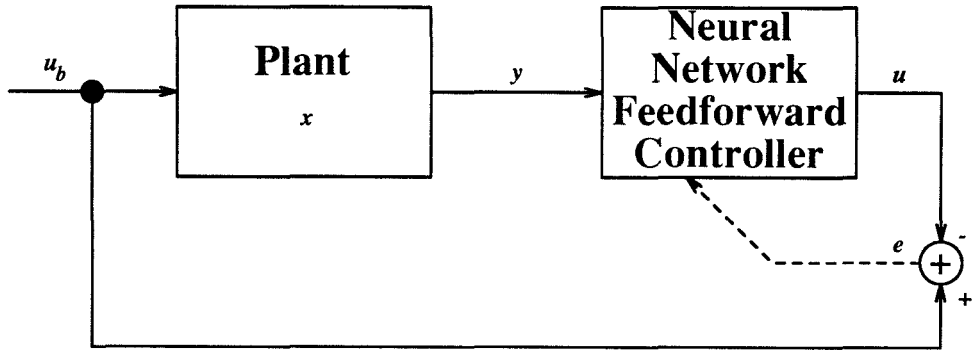



Fig. 2.6 - Generalized Learning Architecture

The advantage of the generalized learning architecture is that we can train the network directly without the use of a preliminary learning step or an auxiliary controller. The disadvantage is that we do not know precisely which set of training inputs $\underline{u}_b$ will provide network inputs $\underline{y}$ in the desired region $Y_d$ of plant operation. Instead, just as a child might try random combinations of exhalation, tongue placement, *etc.* resulting in babbling as it tried to learn to speak; we might provide the plant with random inputs $u_b$ which will produce "babbling" of the plant. Hopefully, the babble input $u_b$ will be sufficiently rich that the resulting plant outputs

will train the network over a region of plant operation $Y_b$ that contains the desired region of plant operation $Y_d$. Any *a priori* knowledge of the relationship between $\underline{y}$ and $\underline{u}$ could be used to more carefully select the set of plant inputs $\underline{u}_b$ used in training. Note that if we know the precise correspondence between $\underline{y}$ and $\underline{u}$, we could use this relationship between plant output and input to train the network directly off-line. We could even use our knowledge of this relationship to implement a nonneural feedforward controller.

### 2.2.2 Specialized Learning and BEPing Through The Plant

In the specialized learning architecture, we place the neural network controller before the plant–in the same configuration as we would when we actually use the network (Fig. 2.7). We present training inputs $\underline{y}_d$ to the network, feed the network output $\underline{u}$ to the plant, and measure the resulting output of the plant $\underline{y}$. Note that we cannot use the standard back-error propagation algorithm to train the network since we do not know $\underline{u}_d$. Several techniques exist for estimating $\underline{u}_d$ as described below.

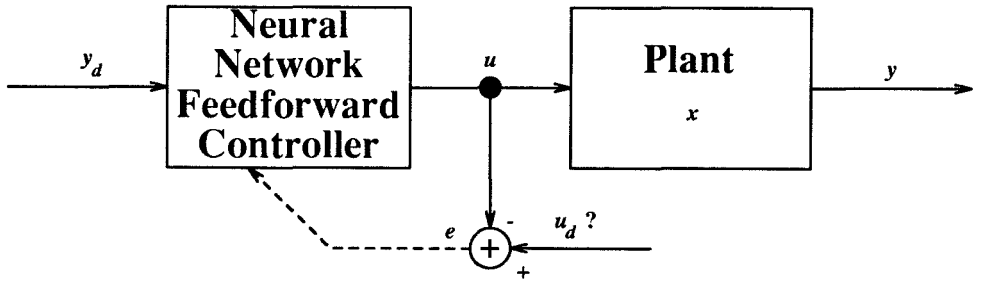



Fig. 2.7 - Specialized Learning: Unknown Target

#### 2.2.2.1 Feedback Error Learning

Any effective control system for all but the most simple processes, requires a feedback path. Thus, practically all control systems employing a neural network in the feedforward path will also have a feedback controller. Assuming that the feedback control signal acts to correct the feedforward control signal, the sum of

the feedforward and feedback control signals may be viewed as the desired output of the network (Eq. 2.35), and thus the feedback control signal can be used as an estimate of the output error of the feedforward neural network used to train the neural network (Eq. 2.36)

$$\underline{u}_d \approx \underline{u} = \underline{u}_{FF} + \underline{u}_{FB} \tag{2.35}$$

$$\underline{\varepsilon}_d = \underline{u}_d - \underline{u}_{FF} \approx \underline{u}_{FB} \tag{2.36}$$

Figure 2.8 shows the basic architecture for using the output of the feedback controller to train the neural network. Both Kawato[11] and Hosogi[12] have proposed using proportional feedback for the feedback controller.

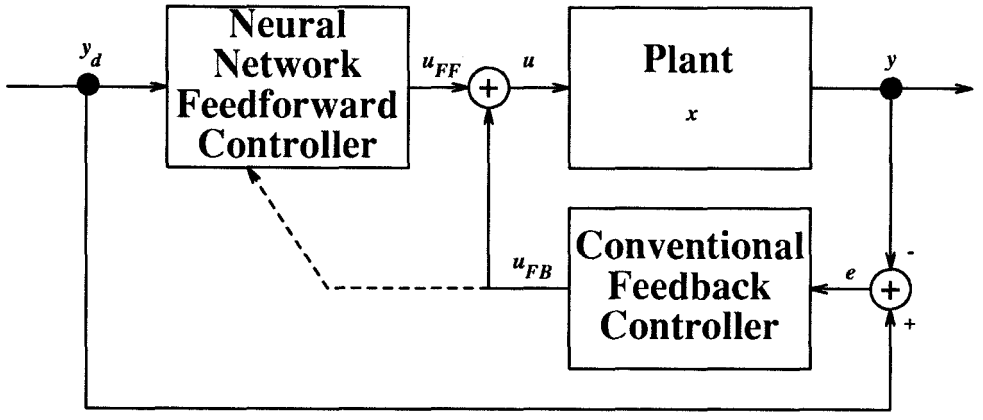



Fig. 2.8 - Specialized Learning with Feedback Controller

Because a feedback controller must be present in any case for effective control, an advantage of using the feedback controller to generate the error signal at the output of the network is that no additional hardware is required. This feedback controller may be designed using techniques such as those described in Sec. 2.1. A disadvantage of feedback error learning is that we must have enough *a priori* knowledge about the plant to design a feedback controller. The feedback controller need not provide completely effective control, however, as long as it acts to correct the feedforward control signal. With an iterative training scheme, an estimate of the

direction of parameter change is more important than an estimate of the magnitude of that change. Given the ability to estimate the direction of parameter change yielding maximal error reduction and a restriction to small parameter adjustments, we can implement a descent algorithm that will reduce the error in the plant output signal.

#### 2.2.2.2 Neural Model of the Plant

In this method, a neural network forward model of the plant is trained as shown in Fig. 2.9 using observed plant inputs and associated plant outputs as a training set.

$$\underline{y} \approx \underline{y}_m(\underline{u}; \underline{W}_m) \tag{2.37}$$

$$\Delta \underline{W}_m \propto -\nabla_{\underline{W}_m} ||\underline{y} - \underline{y}_m||^2 \tag{2.38}$$

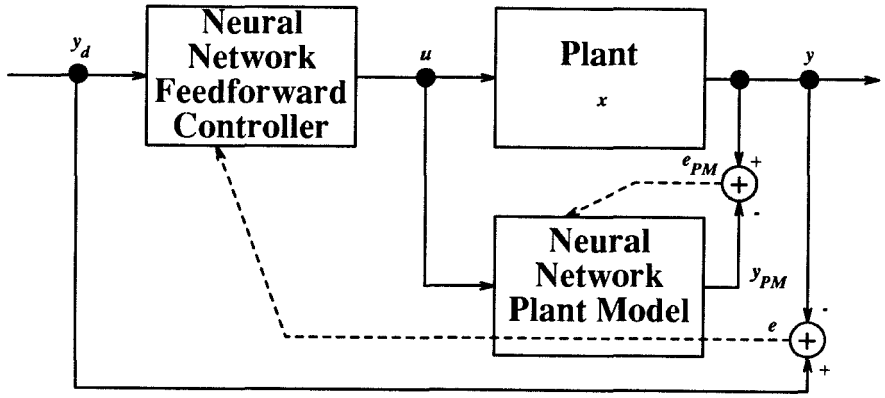



Fig. 2.9 - Neural Model of the Plant

Because the structure and parameters of the neural model of the plant are known, once it has been trained, it can be used to estimate the error at the output of the feedforward network by linearizing the network about its operating point as follows:

$$\underline{\epsilon}_d \approx (\underline{\nabla}_{\underline{u}} \underline{y}^T)(\underline{y}_d - \underline{y}) \tag{2.39}$$

$$\approx (\underline{\nabla}_{\underline{u}} \underline{y}_m^T)(\underline{y}_d - \underline{y}) \tag{2.40}$$

For a multilayer feedforward neural network plant model, the estimated error corresponds to the back-propagated error at the input of the neural plant model:

$$\underline{\epsilon}_d = \underline{\delta}_0 \tag{2.41}$$

$$\underline{\delta}_l = \begin{bmatrix} f_l'(x_1^l) & \underline{0}^T \\ \underline{0} & \ddots \end{bmatrix} [\mathbf{W}_{F,l+1}]\, \underline{\delta}_{l+1} \tag{2.42}$$

$$\underline{\delta}_L = \begin{bmatrix} f_l'(x_1^l) & \underline{0}^T \\ \underline{0} & \ddots \end{bmatrix} (\underline{y}_d - \underline{y}_N) \tag{2.43}$$

The advantage of using a forward model of the plant in training is that a good model of the plant should provide greater accuracy in choosing both the direction and magnitude of the correction required at the output of the feedforward controller.

The disadvantage, of course, is that greater resources are required, and the feedforward model must be learned. Training in this case may occur on- or off-line. A feedback controller is required in the former case. The neural plant modeling technique has been used by Rumelhart,[13] Jordan,[14–16] Widrow,[17–19] and others.

#### 2.2.2.3 BEPing Through The Plant

In BEPing through the plant, we consider the plant to be an extra, unmodifiable layer of our network. The input of the controller is then the desired output of the total network. We define an energy, $E$, equal to the squared error at the output of the enlarged network, and modify the weights in the neural feedforward controller using gradient descent in the parameter space of the feedforward controller:

$$E = ||\underline{y}_d - \underline{y}||^2 \tag{2.44}$$

$$\Delta \underline{w}_F = \nabla_{W_F} E \tag{2.45}$$

$$= 2(\nabla_{W_F} \underline{y}^T)(\underline{y}_d - \underline{y}) \tag{2.46}$$

$$= 2 \left[\nabla_{W_F} \underline{u}^T(\underline{y}_d, \underline{W}_F)\right] \left|\frac{\partial y_i}{\partial u_j}\right|_{ji} (\underline{y}_d - \underline{y}) \tag{2.47}$$

In order to successfully implement this algorithm, we need to estimate $|\partial y_i / \partial u_j|_{ij}$, the plant Jacobian. In the plant model method, the Jacobian is estimated

using a neural model of the plant. The spirit of BEPing through the plant, however, is that we generate this estimate by shorter term observation of the plant inputs and outputs. In effect, we are trying to identify the plant. In modern control, this can be accomplished by perturbing the input with a small signal sinusoid or white noise. In our original neural control paper,[5] we used small perturbations on the input and estimated the elements of the Jacobian using the following equation:

$$\frac{\partial y_i}{\partial u_j} \approx \frac{y_i(\underline{u} + \delta \hat{u}_j) - y_i(\underline{u})}{\delta} \tag{2.48}$$

where $\delta$ is a small constant and $\hat{u}_j$ a unit perturbation of the $j^{th}$ plant input. In Subsec. 2.2.4, other techniques for estimating the plant Jacobian are discussed.

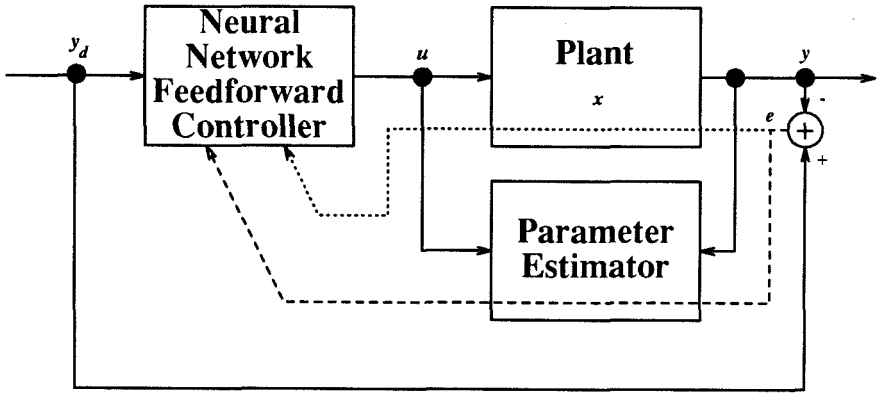



Figure 2.10 - Back Error Propagation Through the Plant

### 2.2.3 Dynamic Inputs

Thus far, we have only considered the inversion of plants with static input-output mappings. In most real world applications; however, the dynamics of the input and output signals are important. We can provide our network with information about these dynamics using a tapped delay line at the input, as shown in Fig. 2.11, feeding an additional dynamic input layer of linear neurons. For example, by using just three closely spaced taps for each input, we could configure the new dynamic input layer to provide the old static input layer with the input signals

and their first two time derivatives. In fact, the output of each input neuron can be viewed as an approximate convolution of a multivariate signal with a Finite Impluse Response (FIR) filter. We can then view the responsibility of the dynamic input layer as providing the rest of the network with an input representation, generated by filtering the controller inputs, appropriate for calculating control inputs using a static nonlinear mapping. Using *a priori* knowledge about the form of the equations governing the behavior of the plant, we can hardwire the dynamic input layer to convert the representation at the output of the tapped delay line into a representation appropriate for generation of the plant input. If we have no *a priori* knowledge with which to choose a good input representation, we do not lose any functionality by eliminating the dynamic input layer and connecting the outputs of the tapped delay line directly to the original input units because the dynamic input neurons are linear.

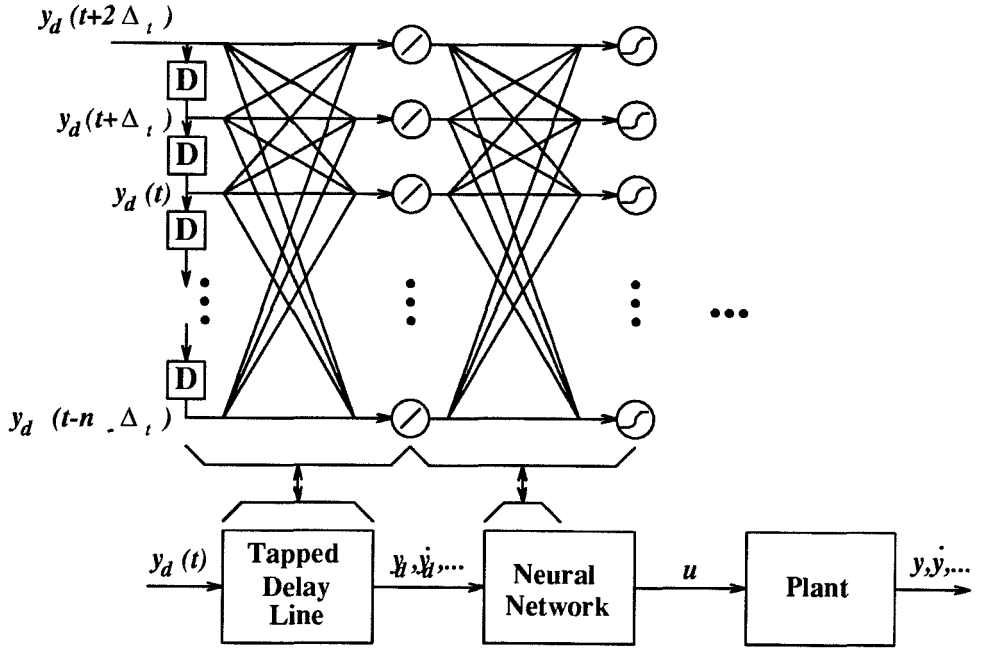



Fig. 2.11 - Providing Input Dynamics Using a Tapped Delay Line

### 2.2.4 Alternative Methods for Estimating the Plant Jacobian

Perturbation methods such as the one we discussed in Subsubsec. 2.2.2.3 for use with static systems can be used for estimating the plant Jacobian of dynamic systems as well. These techniques may be implemented for dynamic systems in one of two ways: first, the same control input may be fed to the plant many times with a single input at a single timestep perturbed each time the control input is fed to the plant followed by comparison of the resulting plant outputs; or second, small signal sinusoids or white noise may be added to each plant input and the magnitude of correlated signals at the plant output measured to estimate the plant Jacobian. The perturbation method discussed previously corresponds to introducing a small step- or delta-function at the input. Alternative methods we have proposed for dynamic systems include using a linear model for the plant and estimating the coefficients by observation of plant inputs and outputs:

$$\underline{y}_L = \mathbf{A}\underline{u} + \mathbf{B}\underline{x} + \underline{c} \tag{2.49}$$

$$= \tilde{\mathbf{A}}\tilde{\underline{u}} \tag{2.50}$$

$$\tilde{\mathbf{A}} = [\,\mathbf{A} \quad \mathbf{B} \quad \underline{c}\,] \tag{2.51}$$

$$\tilde{\underline{u}} = \begin{bmatrix} \underline{u} \\ \underline{x} \\ 1 \end{bmatrix} \tag{2.52}$$

We then wish to identify the elements of the matrix $\mathbf{A}$, $|\frac{\partial y_i}{\partial u_j}|_{ij} \approx |\frac{\partial y_{Li}}{\partial u_j}|_{ij} = \mathbf{A}_{ij} = \tilde{\mathbf{A}}_{ij}$.

Given $n = n_u + n_x + 1$ input and output samples $(\tilde{\underline{u}}^1, \ldots, \tilde{\underline{u}}^n)$ and $(\underline{y}^1, \ldots, \underline{y}^n)$, where $n_u$ is the dimension of the plant input and $n_x$ is the dimension of the plant output, we can attempt to find $\tilde{\mathbf{A}}$ by solving the following matrix equation:

$$[\,\underline{y}^1 \quad \ldots \quad \underline{y}^n\,] = \tilde{\mathbf{A}}\,[\,\tilde{\underline{u}}^1 \quad \ldots \quad \tilde{\underline{u}}^n\,] \tag{2.53}$$

$$\tilde{\mathbf{A}} = [\,\underline{y}^1 \quad \ldots \quad \underline{y}^n\,]\,[\,\tilde{\underline{u}}^1 \quad \ldots \quad \tilde{\underline{u}}^n\,]^{-1} \tag{2.54}$$

In order to reduce the effects of measurement noise, we might instead use a least squares estimate of $\tilde{\mathbf{A}}$. Let us minimize an exponentially weighted sum of the

squared error $E$ of our linear plant model with respect to model parameters $\tilde{\mathbf{A}}$ by solving the following system of equations:

$$E = \sum_n \alpha^{-n} ||\underline{y}^{-n} - \underline{y}_L^{-n}||^2 \tag{2.55}$$

$$0 = \frac{\partial E}{\partial \tilde{A}_{ij}} \quad \forall i,j \tag{2.56}$$

$$= \sum_n \alpha^{-n} \tilde{u}_j^{-n} (y_i^{-n} - \sum_k \tilde{A}_{ik} \tilde{u}_k^{-n}) \quad \forall i,j \tag{2.57}$$

$$\underline{0} = \sum_n \alpha^{-n} \tilde{u}_j^{-n} (\underline{y}^{-n} - \tilde{\mathbf{A}} \underline{\tilde{u}}^{-n}) \quad \forall j \tag{2.58}$$

$$\mathbf{0} = \sum_n \alpha^{-n} (\underline{y}^{-n} - \tilde{\mathbf{A}} \underline{\tilde{u}}^{-n}) \underline{\tilde{u}}^{-nT} \tag{2.59}$$

$$\tilde{\mathbf{A}} = \left[ \sum_n \alpha^{-n} \underline{y}^{-n} \underline{\tilde{u}}^{-nT} \right] \left[ \sum_n \alpha^{-n} \underline{\tilde{u}}^{-n} \underline{\tilde{u}}^{-nT} \right]^{-1} \tag{2.60}$$

By comparison with Eq. 2.24, we see that this is exactly the same formula we found for exponentially weighted least squares parameter estimation in a self-tuning adaptive regulator using a linear plant model. In fact, another way to view BEPing through the plant is that we use adaptive control type techniques to estimate the parameters for a linear model of the plant. Instead of updating a more conventionally designed controller using these estimated parameters, we use them to estimate the error at the output of the feedforward network. We then see BEPing through the plant as a bridge between the use of a conventional feedback controller and the use of a neural plant model since BEPing through the plant essentially uses a nonneural adaptive plant model to estimate the error at the input of the plant.

Using established adaptive control parameter estimation techniques as inspiration, we might also try projection algorithms to estimate the plant parameters. Comparison of Eqs. 1.19 and 2.31 show that the projection algorithms may in fact be viewed as an array of adalines used to model the plant.

## 2.3 Experimental Results

In this section, we present experimental results using various training algorithms paralleling the chronlogical order of their development within our group. We develop a number of simulated tasks, each of which is used to compare the performance of some of the different training algorithms described above.

### 2.3.1 Coordinate Transformation

The first task involves the "control" of a two-input, two-output static plant that converts polar coordinates $(r, \theta)$ to Cartesian coordinates $(x, y)$. Because the dynamics of the inputs are irrelevant, we are simply comparing the ability of the different algorithms to learn, given a function, the inverse of that function. In all cases, we use a two-layer network with no tapped-delay-line at the input. Note that the desired ideal feedforward controller, the inverse of the plant, is not unique in that the plant implements a many-to-one mapping. Assume that we are particularly interested in operating the plant in two disjoint rectangular regions defined in the output space of the plant and labeled "2" and "3" in Fig. 2.12.

#### 2.3.1.1 Generalized Learning

With generalized learning, recall that our training inputs consist of a set of plant inputs and that we may have little or no *a priori* knowledge about which plant inputs correspond to the desired plant outputs. In this case, assume that we know that our desired region of operation is contained in a region of the plant input space defined as follows:

$$\left\{(r,\theta)|r \in [0,10], \theta \in [0,\frac{\pi}{2}]\right\}.$$

We then choose a training set defined essentially as a grid over this region of the input space (circles in Fig. 2.12) as follows:

$$(r,\theta) \in \left\{(1,0), \left(1,\frac{\pi}{4}\right), \left(1,\frac{\pi}{2}\right), (5,0), \left(5,\frac{\pi}{4}\right), \left(5,\frac{\pi}{2}\right), \right.$$
$$\left. (10,0), \left(10,\frac{\pi}{6}\right), \left(10,\frac{\pi}{3}\right), \left(10,\frac{\pi}{2}\right)\right\}$$

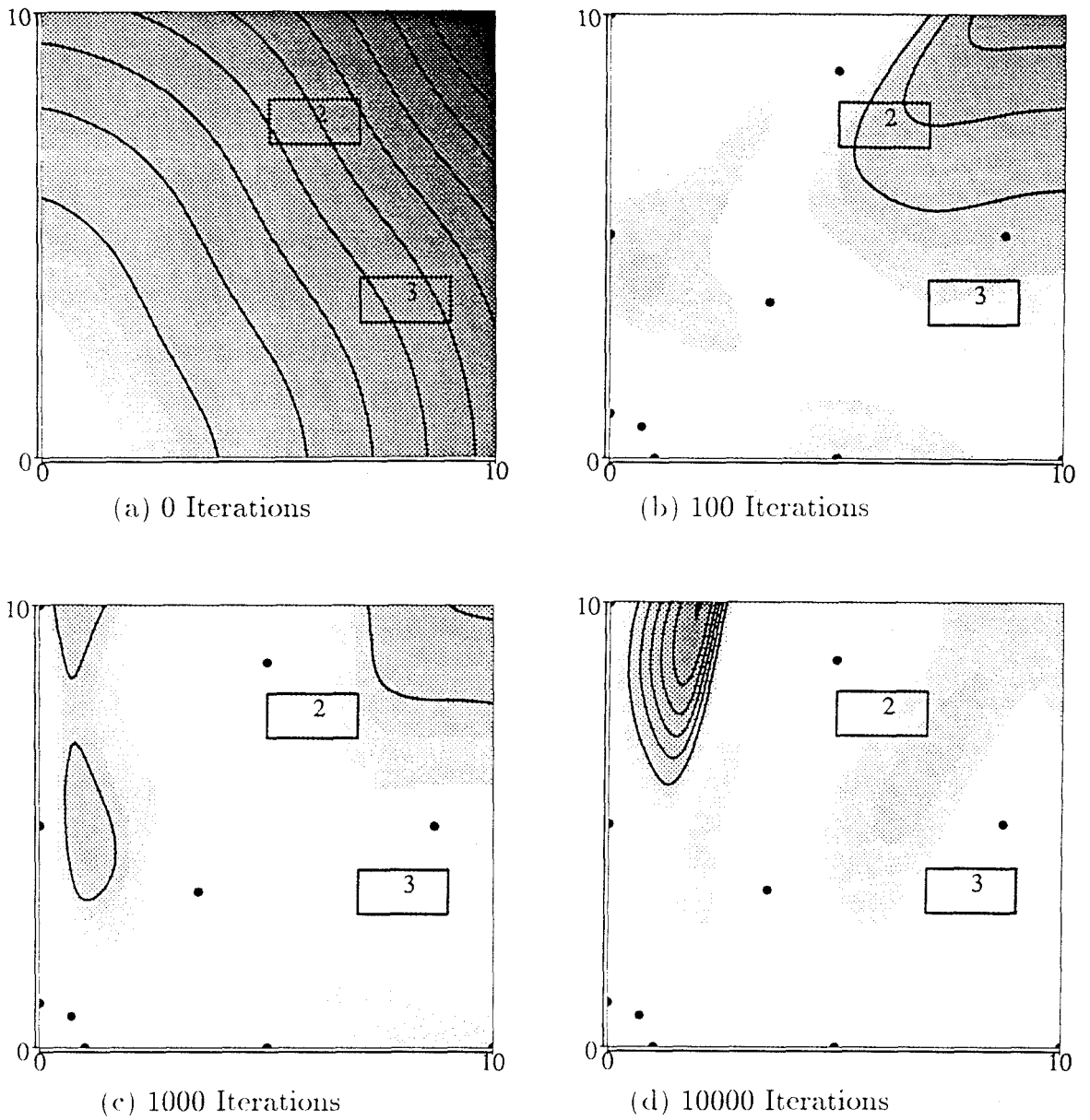
We choose a two-layer network with 10 hidden units. Weights are initialized to random values $w_i \in [-.5, .5]$. The acceleration constant $\rho = .01$. Each iteration consists of the presentation of every point in the training set, in order, with modification of weights after the presentation of each input.

Figure 2.12 shows the squared error at the output of the plant versus the input of the neural network when the output of the network is fed directly to the plant. Dark areas indicate high error, light areas low. Each contour in the plot represents lines of constant squared error, with a squared error differential of 20 between contours. Fig. 2.12 shows the error after 0, 100, 1000, and 10000 iterations. The circles in the figures represent the points of generalized training. Note that the error rapidly decreases over the entire region, but then begins to climb in between the training points as we continue training.

A disadvantage of generalized learning is that one cannot specifically choose training points that lie in and around the desired operating region. With generalized learning, we might have to sample a large region of the input space in hopes of sufficiently sampling the desired region of operation, rectangles 2 and 3. This will of course require greater resources and training time than otherwise required for satisfactory performance.

#### 2.3.1.2 BEPing Through the Plant

With specialized learning, we may concentrate our learning efforts in the desired region of operation. For this static problem, we choose to use BEPing through the plant with perturbation of plant inputs to estimate the plant Jacobian. We present a point in the training set to the neural network, either unperturbed or with a small (10%) perturbation of each input separately. Comparison of the resulting plant outputs with the unperturbed input and the perturbed inputs allow us to estimate the plant Jacobian. This can then be used to propagate the error at the




(a) 0 Iterations

(b) 100 Iterations

(c) 1000 Iterations

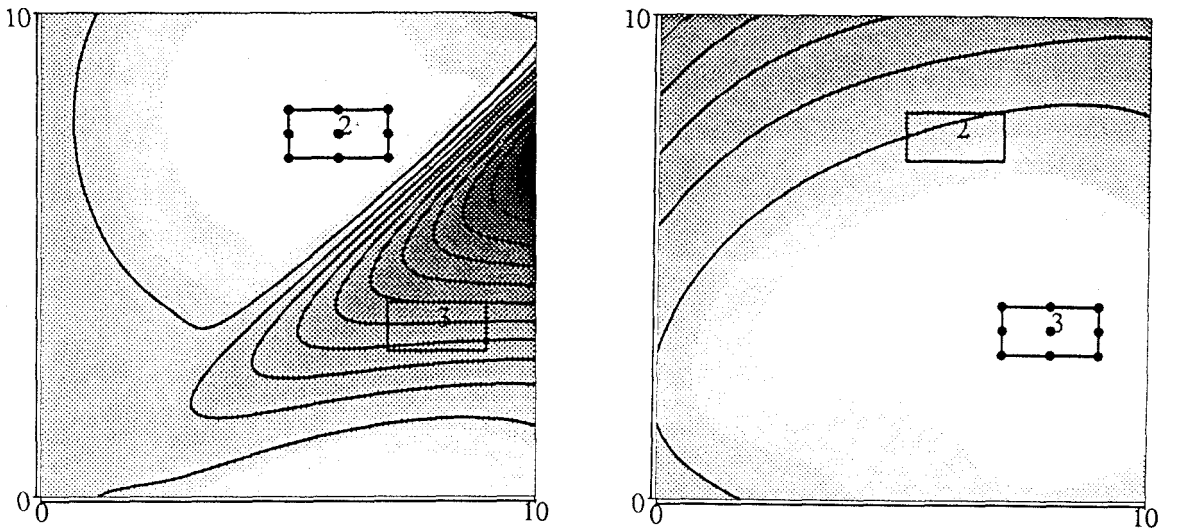
(d) 10000 Iterations

Fig. 2.12 - Squared Error for Coordinate Transformation with Generalized Learning

output of the plant back to the output of the network. We use back propagation with the same parameters as before to modify the weights in the network.

Figure 2.13a and b show the results after 1000 iterations in regions 2 and 3 respectively. Note that in both cases, the network can learn effectively with

specialized learning. Note also that *interpolation* in the desired regions of operation is better in the specialized learning case. This is most likely because the training points are sufficiently close together to sample the mapping.




(a) Rectangle 2 (b) Rectangle 3

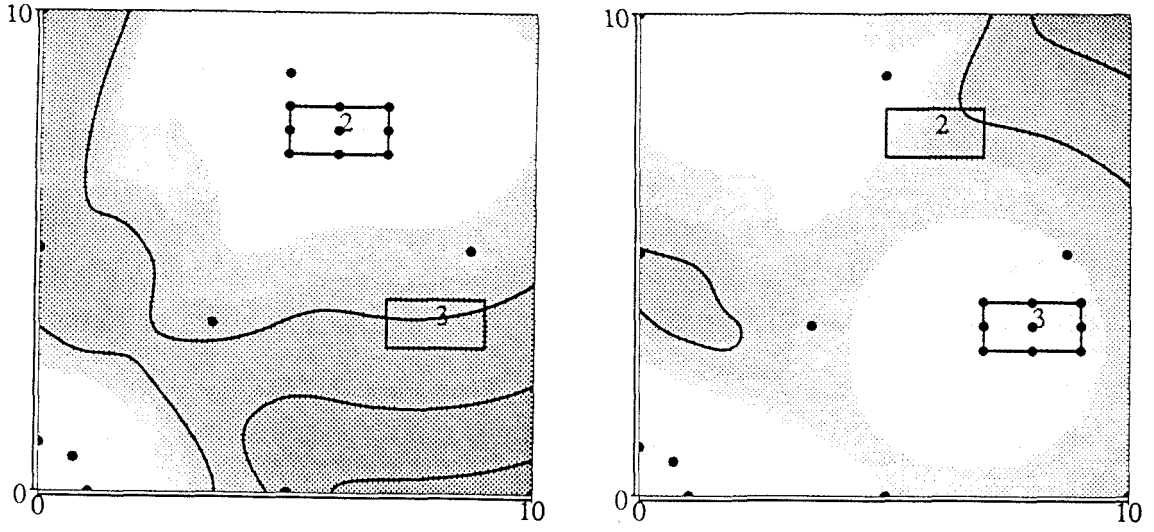
Fig. 2.13 - Squared Error for Coordinate Transformation
After 1000 Iterations of BEPing Through the Plant

Figure 2.14 shows the error after 100 iterations of generalized learning followed by 1000 iterations of specialized learning in any region of interest. We note that a little generalized learning before specialized learning can lead to better overall generalization.

Figure 2.15 shows that when switching from one specialized learning region to the other, there is a tendency to forget about the first error region. Thus it is important to develop techniques to learn in multiple regions.

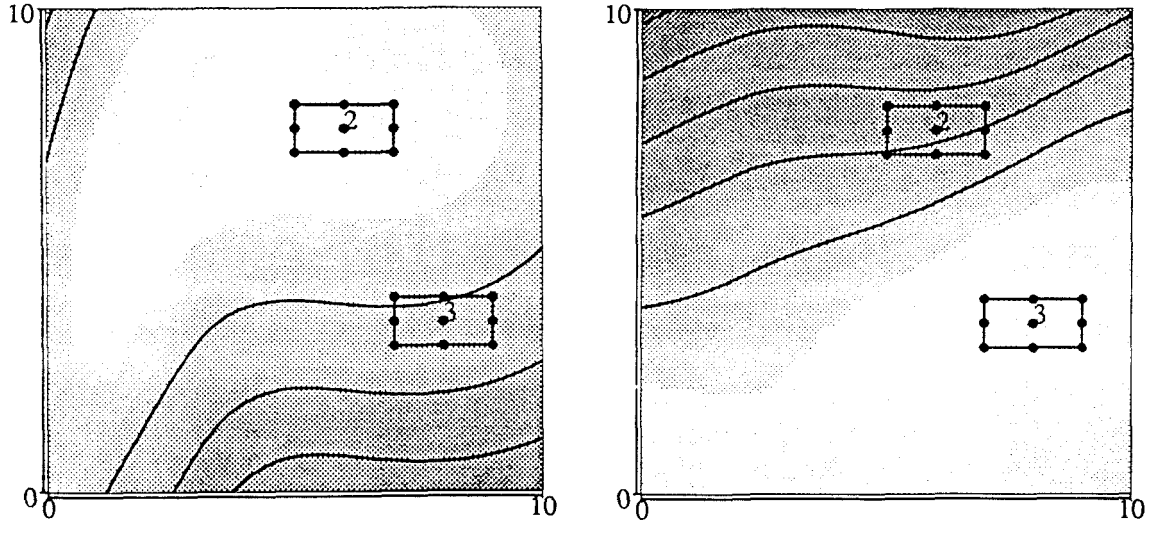
### 2.3.2 Robot Manipulators

The rest of our tasks consist of simulations involving robot manipulators. Robot manipulators or robot arms, such as those shown in Fig. 2.16 are mechanical




(a) Rectangle 2 (b) Rectangle 3

Fig. 2.14 - Squared Error for Coordinate Transformation
After 100 Iterations of Generalized Learning
and 1000 Iterations of BEPing Through the Plant




(a) 3 then 2 (b) 2 then 3

Fig. 2.15 - Squared Error for Coordinate Transformation
After Switching Regions of Specialization

devices designed to move objects through space. Robot manipulators are currently

used to perform a wide variety of tasks from welding automobiles to deploying satellites.

We often model manipulators as a series of rigid *links* connected at *joints*. Joints may either be rotational or translational. Each joint is presumed to have a single degree of rotational or translational freedom. If two links meet at a joint having multiple degrees of freedom, we assume the presence of links of zero length between each "joint." Given rigid links, a set of variables describing the "position" of each joint uniquely determines the configuration of the manipulator. Often, useful work is performed only by the end of the last joint or *end-effector*. Given a physical model of the system, it is possible to derive (kinematic) equations that map each set of joint coordinates to a position and orientation for the end-effector. Each joint typically has some means by which the position of the joint may be adjusted. We assume that mechanical devices governing the position, linear or angular velocity, or force or torque at each joint are available and are the inputs of the manipulator. Various sensors may also be integrated into the manipulator to either directly or indirectly measure the state of the manipulator and its interaction with the environment.

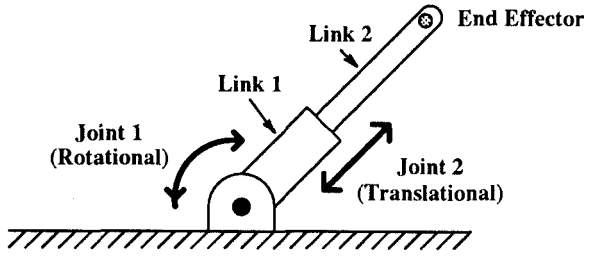



Fig. 2.16 - Links and Joints

Robot manipulators are an exciting application for neural networks. Their dynamics are often nonlinear in nature, and can often become quite complex as the number of joints increases. Furthermore, factors such as flexibility in the links and friction in the joints can make modeling very difficult. Finally, the objects that the manipulator move or operate on may be nonuniform in size, weight, position, *etc.*

The ability of neural networks to learn and adapt is very attractive in both planning and control phases of robot manipulator operation. In the tasks that follow, we consider utilizing neural networks for control of one- and two-link manipulators.

### 2.3.3 One-Link Manipulator

The first task demonstrated that both generalized and specialized learning can be used to train a network to invert a static mapping at a set of training points. In the second task, we test whether the tapped delay line provides adequate information for control of a dynamic system. This next task involves the control of a one-link manipulator as shown in Fig. 2.17.

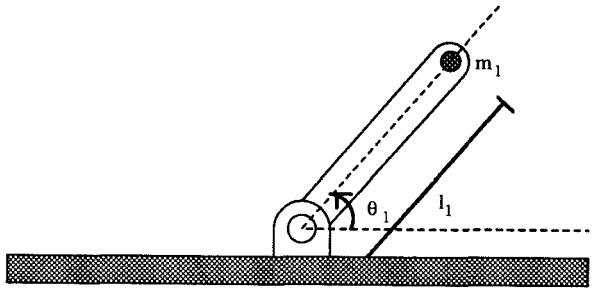



Fig. 2.17 - One-Link Manipulator

The manipulator consists of a single rigid body, anchored through a rotational joint to the ground, and constrained to move in a vertical plane. We model the single-link as a massless rigid body, one meter long with a one kilogram load at the end-effector. We assume a gravitational acceleration of 10 $m/s^2$. The dynamic equation for such a single-link manipulator takes the following form:

$$\tau = m_1 l_1^2 \ddot{\theta} + m_1 l_1 g \cos(\theta) \tag{2.61}$$

Notice that even with a single link, the dynamics of the one-link manipulator are nonlinear in the joint position, $\theta$. We assume that we wish to operate the manipulator on trajectories with the following limits on joint position and acceleration: $0 \leq \theta \leq \pi$ *radians* and $|\ddot{\theta}| \leq 15$ *radians*$/s^2$.

We simulate the network and manipulator in discrete time. Each step in discrete time represents $dt = .1\ s$ of real time. We integrate Eq. (2.61) using forward Euler integration as follows:

$$x[n+1] = x[n] + \dot{x}[n]dt \tag{2.62}$$

We use a three-layer network for the feedforward controller. Because we need joint angle and desired acceleration information only to calculate the required torque, we use a delay line with three taps at the input and hardwire the first layer to estimate the joint acceleration:

$$\underline{y}^{(0)}[n] = \begin{bmatrix} \theta_d[n] \\ \theta_d[n+1] \\ \theta_d[n+2] \end{bmatrix} \tag{2.63}$$

$$\underline{y}^{(1)}[n] = \begin{bmatrix} y_0^{(0)} \\ \frac{y_2^{(0)} - 2y_1^{(0)} + y_0^{(0)}}{dt^2} \end{bmatrix} \approx \begin{bmatrix} \theta[n] \\ \ddot{\theta}[n] \end{bmatrix} \tag{2.64}$$

The weights for the second and third layer are initialized to random values in $w_i \in [-.5, .5]$ and determined through learning. We arbitrarily choose to us ten neurons in the second layer, each of which implements a sigmoidal nonlinearity $f(x) = \frac{1}{1+e^{-x}}$.

#### 2.3.3.1 Fixed Training Points

We use off-line specialized learning to train the network. During off-line training, we bypass the first, hardwired layer and initially choose to train the network on a fixed training set of 100 points chosen on either a 5×5 or a 10×10 uniformly spaced Cartesian grid over the desired operating region. The plant Jacobian is estimated using perturbation of plant inputs. We used an acceleration constant of $\rho = .01$ during training.

Tables 2.1 and 2.2 show the squared error at the output, averaged over the training points and over the entire input space after various numbers of iterations for the two grids. They also show the maximum squared error over the input space.

Table 2.1 - One-Link Error after Learning on 5×5 Grid

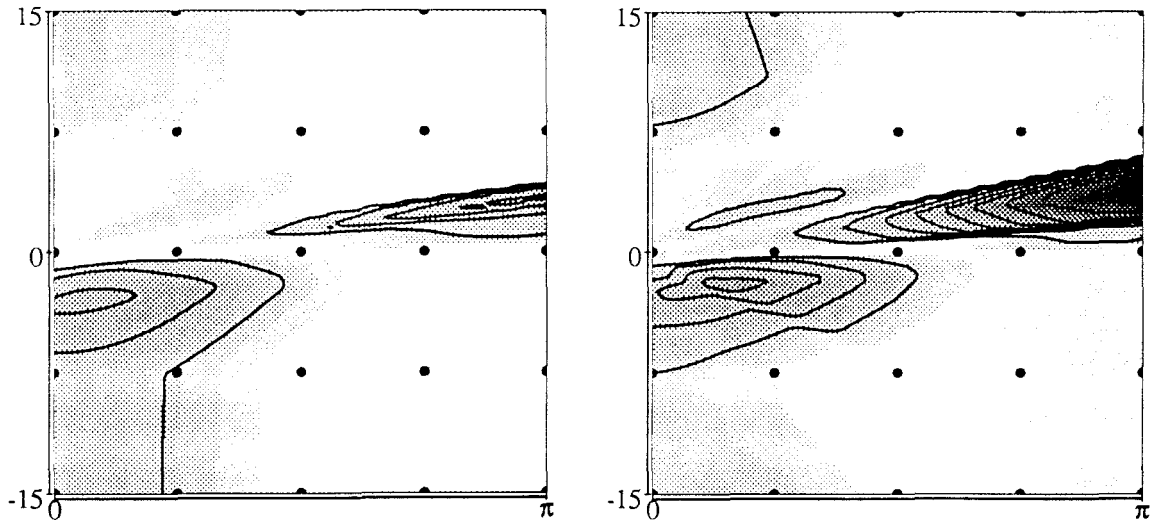
| Iterations | Av. Sq. Err. | Max. Err. | Av. Sq. Tr. Err. |
|---|---|---|---|
| 112000 | 1047 | 8731 | 651 |
| 317800 | 1571 | 19238 | 528 |

Table 2.2 - One-Link Error after Learning on 10×10 Grid

| Iterations | Av. Sq. Err. | Max. Err. | Av. Sq. Tr. Err. |
|---|---|---|---|
| 24200 | 436 | 3639 | 466 |
| 37800 | 332 | 3399 | 324 |
| 39400 | 220 | 3224 | 252 |

We find that the squared error is quite large in comparison to the maximum torque of squared output of 625.

Figure 2.18 shows the squared error at the output over the operating regime after (a) 112 000 and (b) 317 800 iterations of specialized learning. The contour spacing is 2000. The black circles mark the location of the training points. Notice that just as we saw for the coordinate transformation task, the error tends to be lower at the training points, but can grow quite large between training points.




(a) 112000 Iterations (b) 317800 Iterations

Fig. 2.18 - Squared Error for One-Link Manipulator Using Fixed Training Points

#### 2.3.3.2 Random Training Points

As we saw in the coordinate transformation task, as well as the one-link manipulator with fixed training points, we must adequately sample the input space to generate a network that achieves good generalization. The problem that arises when we use too many resources or too few training points is that the error can become quite large between training points.

When using random training points, we randomly select a training point in the desired operating region, feed it to the network followed by the plant, and modify the weights such that the plant output moves towards the desired output to complete each iteration. That training point is then discarded, and another one chosen from the continuum of training points available in the desired operating region. The idea behind the use of random training points is that instead of minimizing the error at a fixed set of points, we minimize a Monte-Carlo integration of the error over the entire operating region. In effect, as the number of iterations grows without bound, the number of training points around any fixed neighborhood of a point in the operating regime also grows without bound.

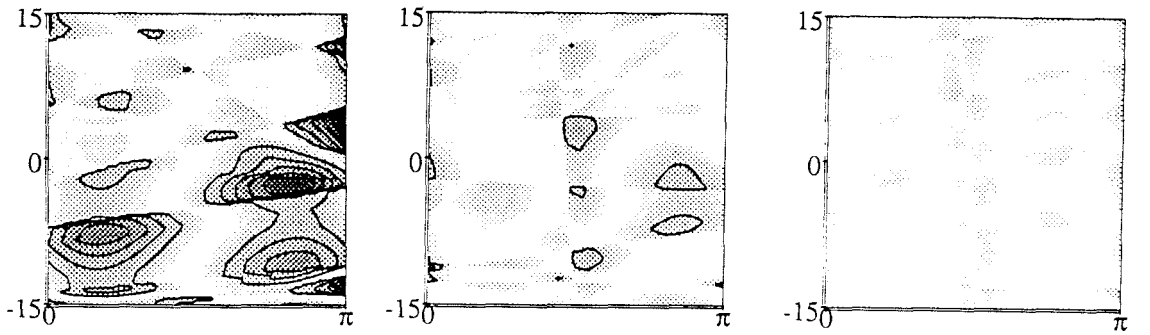
Table 2.3 shows the average and maximum errors after specialized learning using random training points chosen from the region of operation after 34100, 89100, 142100, and 163300 iterations. We notice that in this case, both the average and maximum errors decrease as we increase the number of iterations.

Table 2.3 - One-Link Error after Random Learning

| Iterations | Av. Sq. Err. | Max. Err. |
|---|---|---|
| 34100 | 89 | 663 |
| 89100 | 24 | 278 |
| 142100 | 12 | 108 |
| 163300 | 10 | 99 |

Figure 2.19 shows the squared error at the output over the operating regime after 34100, 89100, and 163300 iterations of specialized learning with random train-

ing points. The contour spacing is 100. Notice that unlike the case with the fixed training set, the error with random learning tends to decrease uniformly at all points in the desired operating regime.




(a) 34100 Its. (b) 89100 Its. (c) 163300 Its.

Fig. 2.19 - Squared Error for One-Link Manipulator Using Random Training Points

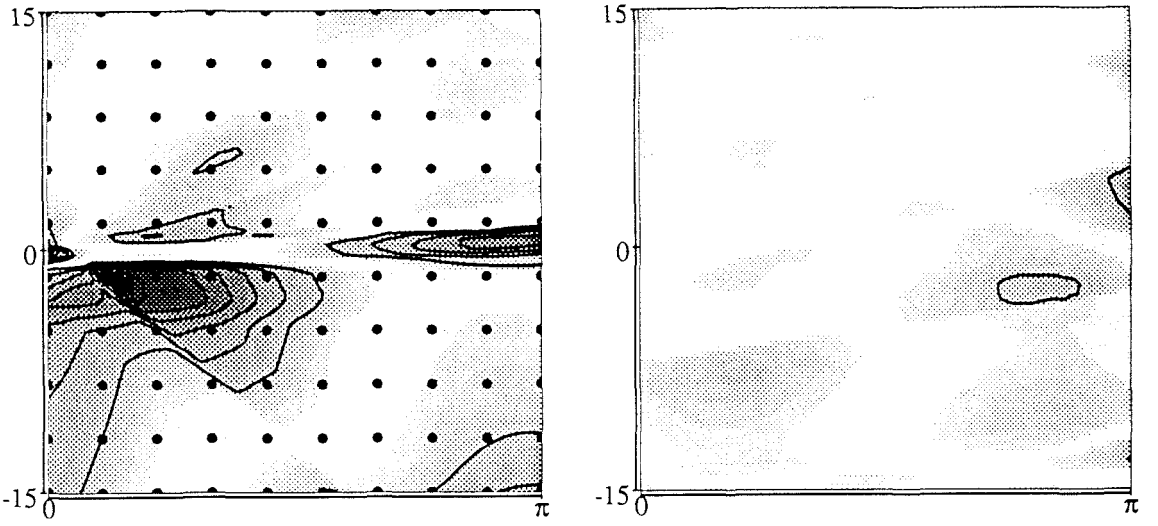
Figure 2.20 compares the squared error after (a) 39400 iterations with a $10 \times 10$ grid over the input and (b) 34100 iterations of random learning. The contour spacing is 500. We see that random training tends to provide better generalization than the use of a fixed training set for a similar number of iterations.

#### 2.3.3.3 Random Training Points and Generalization

In Chapter 1, we found that if a network effectively learns a mapping at a number of training points greater than the VC-dimension or capacity of the network, it must have discovered a means of representing the problem that achieves some measure of generalization. In this context, the use of random training points with a network of fixed size may be viewed as an attempt to force the network to generalize by increasing the number of training points while holding the capacity of the network constant.

### 2.3.4 Two-Link Manipulator

The next task involves the control of a two-link manipulator as shown in




(a) Fixed (b) Random

Fig. 2.20 - Comparison of Squared Error for One-Link Manipulator using Fixed and Random Training Points

Fig. 2.21. This manipulator consists of a massless rigid body, two meters in length and anchored to the ground through a rotary joint at one end with a two kilogram load at the other. A second massless rigid body, one meter long, connects through a rotary joint to the first link and carries a one kilogram load at the end-effector. Once again, we assume a gravitational acceleration of 10 $m/s^2$. The Newton-Euler equations for the dynamics of such a manipulator may be expressed in the following form where $\mathbf{M}$ is the mass matrix, $\underline{v}$ the vector of centrifugal and Coriolis terms, and $\underline{g}$ the vector of gravitational terms:

$$\underline{\tau} = \mathbf{M}(\underline{\theta})\ddot{\underline{\theta}} + \underline{v}(\underline{\theta}, \dot{\underline{\theta}}) + \underline{g}(\underline{\theta}) \tag{2.65}$$

$$\mathbf{M}(\underline{\theta}) = \begin{bmatrix} 2m_2 l_1 l_2 cos(\theta_2) + m_1 l_1^2 + m_2 l_1^2 + m_2 l_2^2 & m_2 l_1 l_2 cos(\theta_2) + m_2 l_2^2 \\ m_2 l_1 l_2 cos(\theta_2) + m_2 l_2^2 & m_2 l_2^2 \end{bmatrix} \tag{2.66}$$

$$\underline{v}(\underline{\theta}, \dot{\underline{\theta}}) = \begin{bmatrix} -m_2 l_1 l_2 sin(\theta_2)\dot{\theta}_2^2 - 2m_2 l_1 l_2 sin(\theta_2)\dot{\theta}_1\dot{\theta}_2 \\ m_2 l_1 l_2 sin(\theta_2)\dot{\theta}_1^2 \end{bmatrix} \tag{2.67}$$

$$\underline{g}(\theta) = \begin{bmatrix} m_2 l_2 g cos(\theta_1 + \theta_2) + m_1 l_1 g cos(\theta_1) + m_2 l_1 g cos(\theta_1) \\ m_2 l_2 g cos(\theta_1 + \theta_2) \end{bmatrix} \tag{2.68}$$

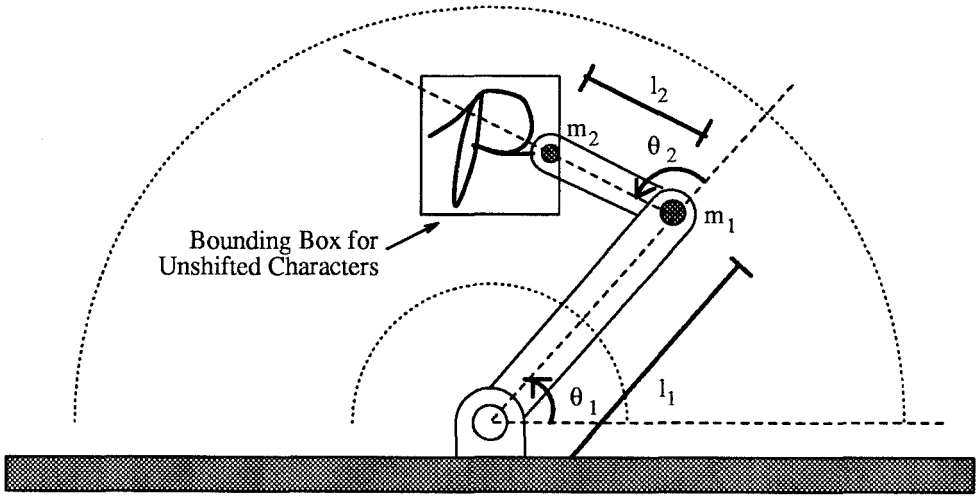



Fig. 2.21 - Two Link Manipulator

Notice that in comparison with the case of a single-link manipulator (Eq. 2.61), the addition of a link significantly increases the complexity of dynamic equations. Again, we assume joint angle, velocity, and acceleration limits, as follows, for the operation of the manipulator: $0 \leq \theta_1 \leq \pi$ *radians*, $|\theta_2| \leq \pi$ *radians*, $|\dot{\theta}_1| \leq 5$ *radians/s*, $|\dot{\theta}_2| \leq 10$ *radians/s*, $|\ddot{\theta}_1| \leq 15$ *radians/s*$^2$, $|\ddot{\theta}_2| \leq 75$ *radians/s*$^2$.

We simulated the network and manipulator in discrete time with $dt = .1$ *s*. We employed forward Euler integration (Eq. 2.62) of the differential equations. We generally use three-layer networks for the feedforward controller. With joint angle, velocity, and desired acceleration information required to calculate torque, we again use a delay line with three taps at each input and hardwire the first layer to estimate the joint velocities and accelerations:

$$\underline{y}^{(0)}[n] = \begin{bmatrix} \theta_1 d[n] \\ \theta_1 d[n+1] \\ \theta_1 d[n+2] \\ \theta_2 d[n] \\ \theta_2 d[n+1] \\ \theta_2 d[n+2] \end{bmatrix} \tag{2.69}$$

$$\underline{y}^{(1)}[n] = \begin{bmatrix} y_0^{(0)} \\ \frac{y_1^{(0)} - y_0^{(0)}}{dt} \\ \frac{y_2^{(0)} - 2y_1^{(0)} + y_0^{(0)}}{dt^2} \\ y_3^{(0)} \\ \frac{y_4^{(0)} - y_3^{(0)}}{dt} \\ \frac{y_5^{(0)} - 2y_4^{(0)} + y_3^{(0)}}{dt^2} \end{bmatrix} \approx \begin{bmatrix} \theta_1[n] \\ \dot{\theta}_1[n] \\ \ddot{\theta}_1[n] \\ \theta_2[n] \\ \dot{\theta}_2[n] \\ \ddot{\theta}_2[n] \end{bmatrix} \tag{2.70}$$

The weights for the second and third layer were initialized to random values in $w_i \in [-.5, .5]$ and determined through learning. Each neuron in the second layer implemented a sigmoidal nonlinearity $f(x) = \frac{1}{1+e^{-x}}$.

We initially trained the networks at fixed points along a specific trajectory using 5, 10, 20, and 30 neurons. Fig. 2.22 compares the output of the neural networks versus the actual torque along the trajectory, at or in between the training points for the 5 and 30 neuron cases. We find that there are regions of both high and low error. In addition, the behavior of the networks does not seem to change appreciably as we vary the number of neurons.

We also trained networks of 30, 50, 100, and 200 neurons using random learning over the entire operating region. Fig. 2.23 compares the desired torque with the torque produced by the random sample trained network. As shown in the figure, although the random learning algorithm provides a smoother response and is able to learn rudimentary features, it is far from being able to learn how to control the two-link manipulator.

Because we have two outputs to our network, corrective actions for one of the outputs may lead to degradation of the other. Thus, we also tried "splitting" the network by providing separate hidden units for each output. Fig. 2.24 shows the response after random training when each output had 100 hidden units of its own. As in regular random learning, the response is smooth and shares the general shape of the desired response, however it also falls short of acceptable error levels. We also tried a network with three variable layers and one fixed. Again, the response

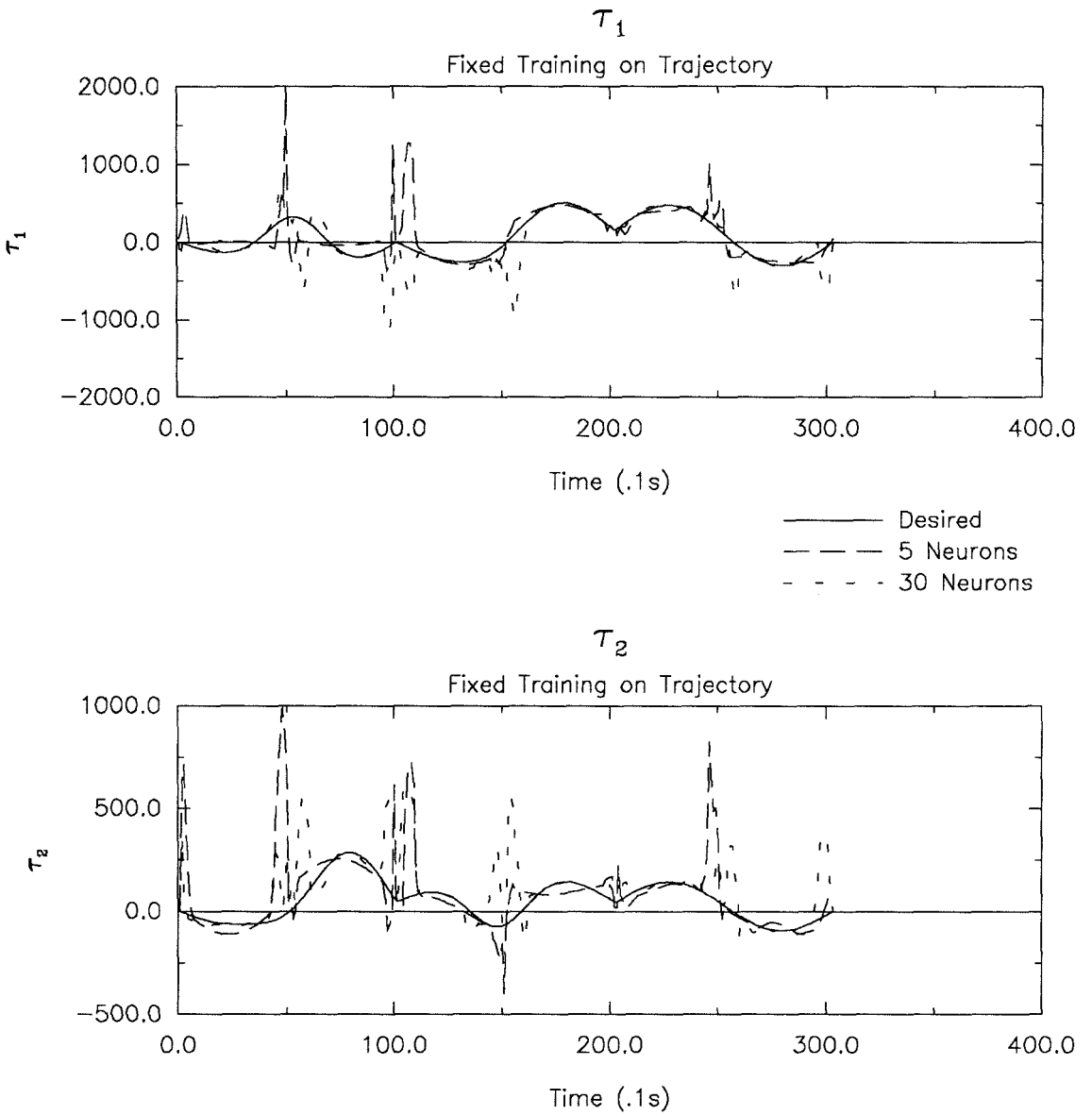



Fig. 2.22 - Learning on Specific Trajectory

was smooth, but the error was unsatisfactory. Fig. 2.25 shows the response when we had 80 neurons in the first hidden layer and 20 neurons in the second hidden layer.

Although one could speculate endlessly on the reason why we were unable to successfully train the network, I believe that the nonlinear mapping of (Eq. 2.65)

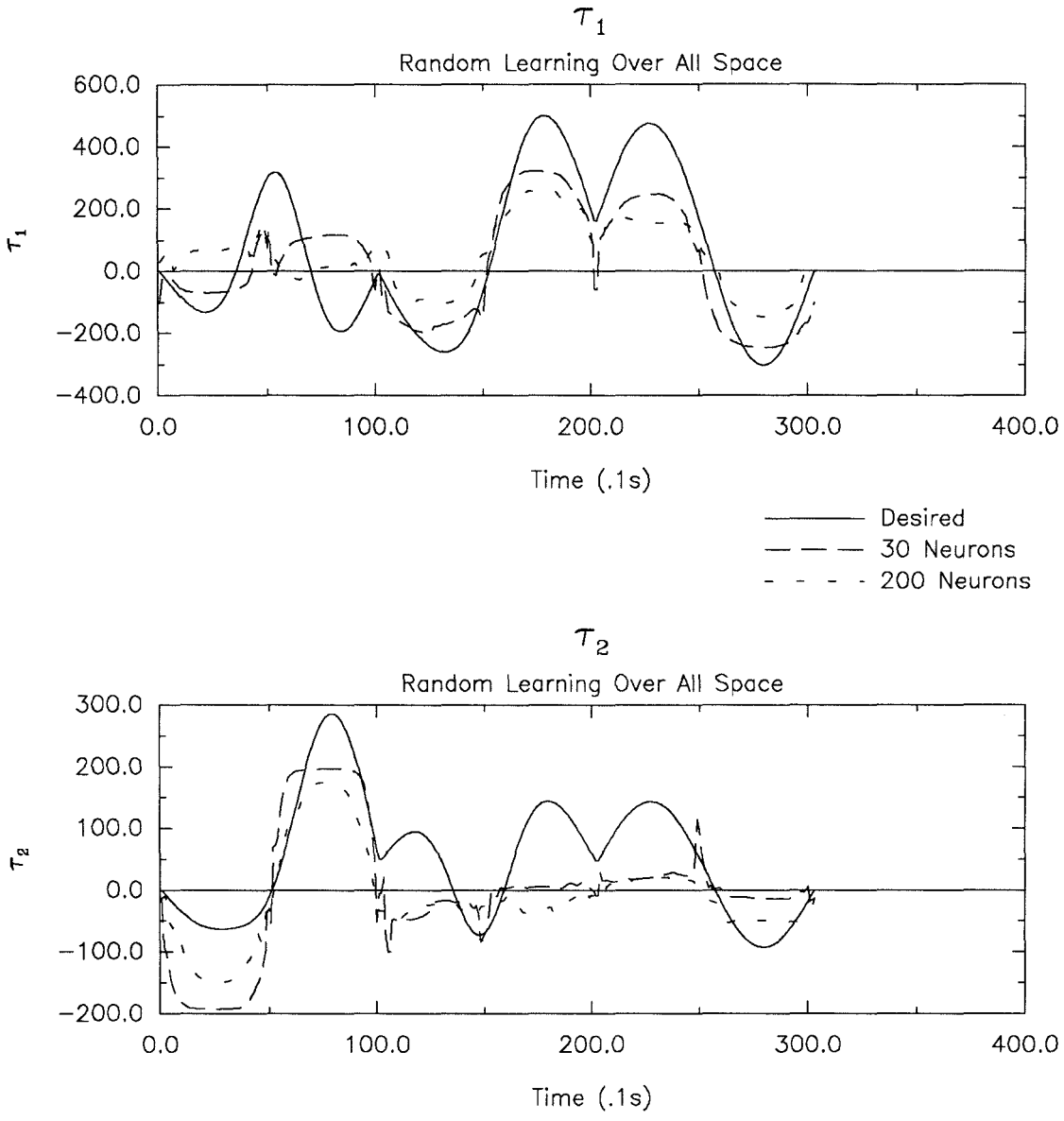



Fig. 2.23 - Random Learning Over All Space

was too difficult to learn over large operating regions using standard BEP with *ad hoc* choice of network architecture.

### 2.3.5 The Two-Link Manipulator Revisited

We look once again to controlling the two-link manipulator, but with a num-

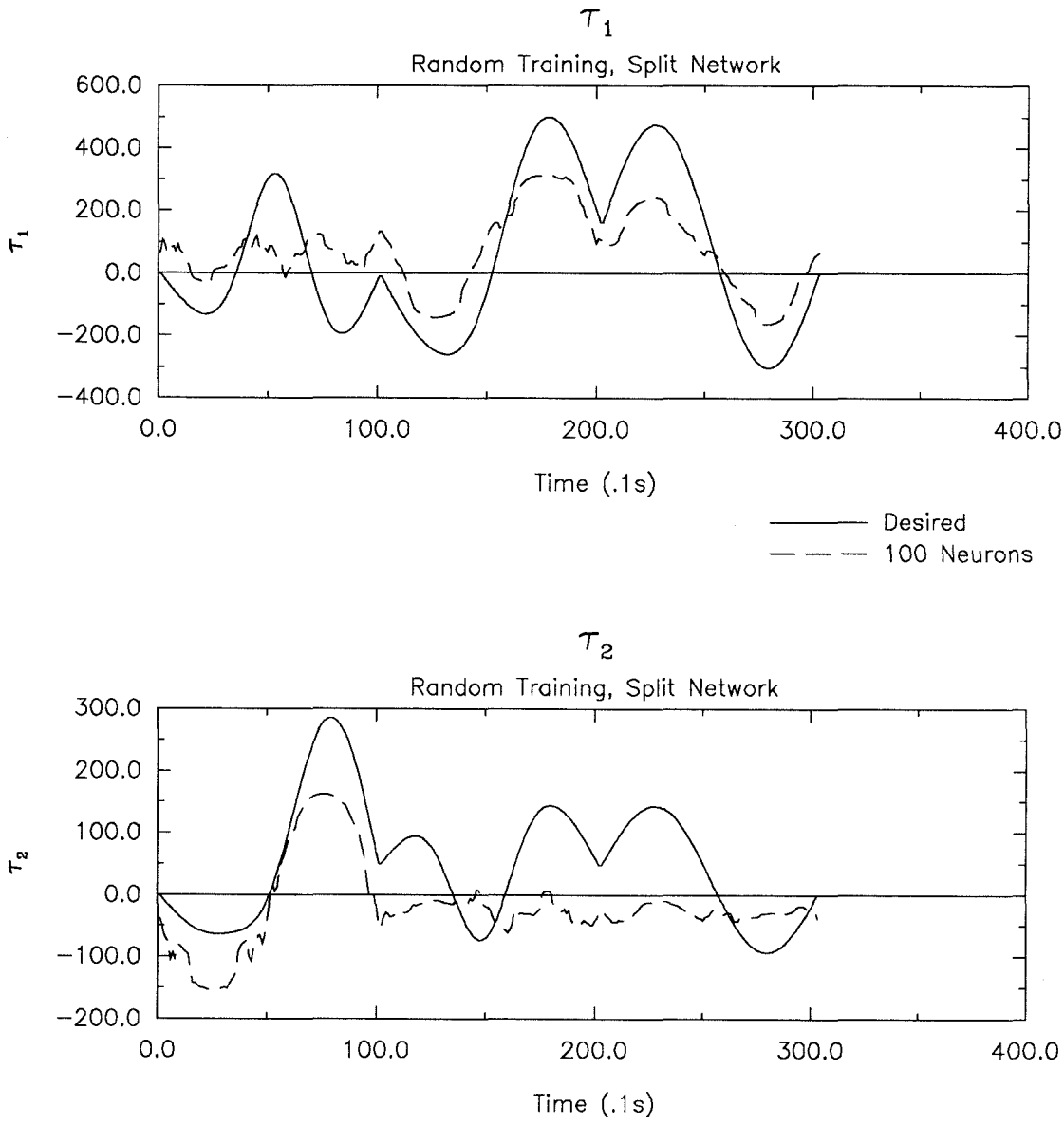



Fig. 2.24 - Random Learning Over All Space With a Split Network

ber of modifications. First, we refine the task itself to focus on a specific application, thus effectively reducing the region over which we must learn to operate the plant. Second, we switch from standard BEP to Ji's algorithms decsribed in Subsections 1.5.11 and 1.5.12.

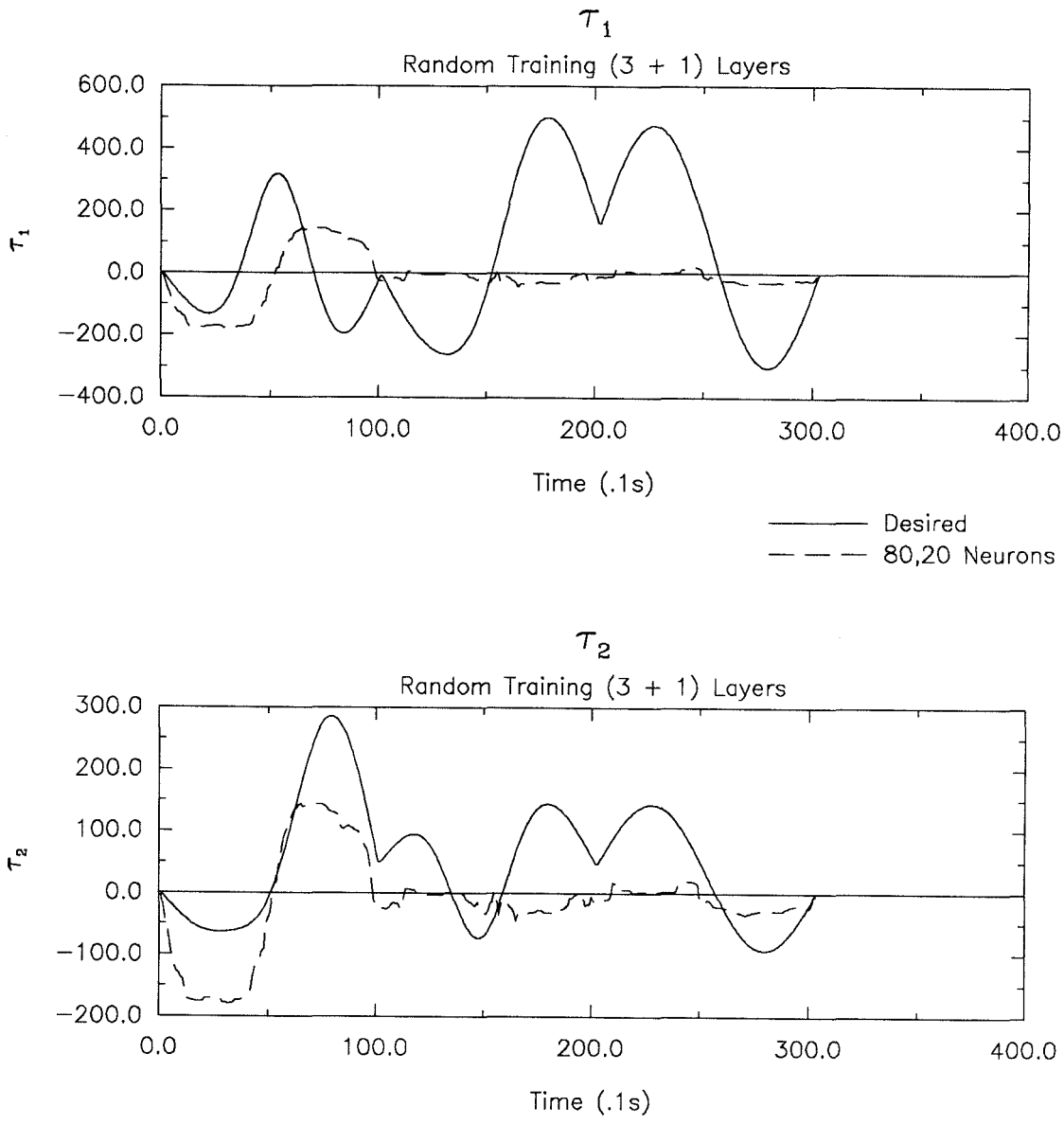



Fig. 2.25 - Random Learning with Three Layer (+1 Hardwired)

### 2.3.5.1 Drawing Handwritten Characters

In this subsection, we focus the training on achieving a specific task–drawing handwritten characters. In other words, the trajectories we attempt to follow are specifically designed to trace out handwritten characters with the end-effector.

We begin by defining a square region (Fig. 2.21), one meter on a side within

which we will train the network. First, a handwritten character is drawn in a square region on a piece of paper (Fig. 2.26a). Second, a human operator chooses points along the character that "define" the general shape of the character (Fig. 2.26b). Third, a 2-D cubic spline is fitted to the chosen points to generate a parametric version of the character. The parametric version is compared to the original to determine whether it is a good approximation. If not, the sample points may be adjusted. Fourth, the parametrized character is sampled at uniform intervals corresponding to the position of the end-effector at each discrete time step. The resulting sequence of positions represents a constant speed trajectory. The points at the start and end of the trajectory are chosen such that the initial and final

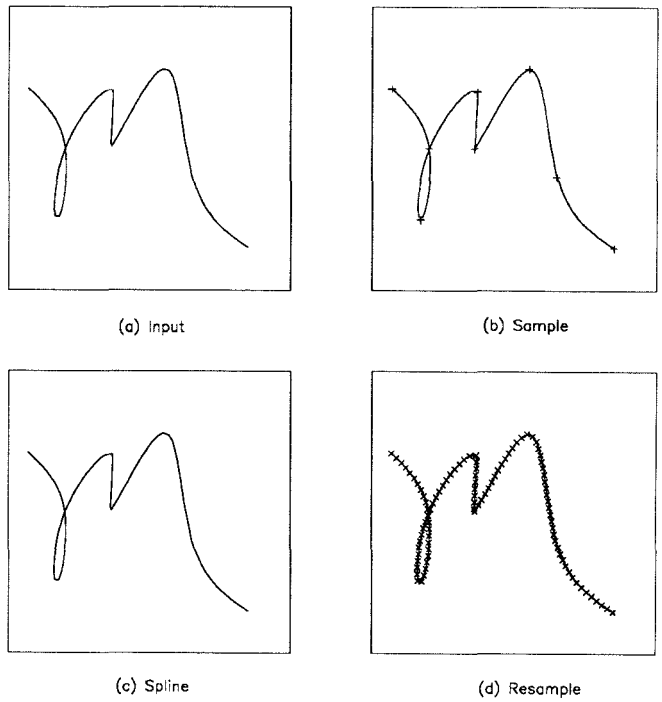



Fig. 2.26 - Character Generation (a) Input Handwritten Character, (b) Select Salient Points, (c) Fit With Cubic Spline, and (d) Resample at Uniform Intervals for Constant Speed Trajectory

velocities are zero.

All the operations above are performed in Cartesian coordinates. We assume, however, that the sensors in the manipulator measure its position in joint coordinates. We must thus convert between Cartesian and joint coordinates. The equations for conversion from joint to Cartesian coordinates are as follows:

$$x = l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2) \tag{2.71}$$

$$y = l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2) \tag{2.72}$$

However, this is not a one-to-one mapping. Thus, there exist degenerate positions in joint coordinates that correspond to the same position for the end-effector in Cartesian coordinates. For the two-link manipulator, the degenerate arm configurations,

if they exist, correspond to whether the angle between the first link and the second link $\theta_2$ is less than or greater than zero (Fig. 2.27). For those cases where $\theta_2$ is positive, the "elbow" or second joint will be to the right of an imaginary line between the "shoulder" or first joint and the end-effector or "hand." This configuration will be called "right-handed" while the corresponding degenerate configuration will be called "left-handed." The shaded areas of Fig. 2.27 show the regions in Cartesian space that the end-effector may access in a right-handed configuration of the arm. We assume all characters are drawn in a right-handed configuration. The equations for conversion from Cartesian coordinates to right-handed joint coordinates are as follows:

$$\theta_1 = \tan^{-1}\left(\frac{y}{x}\right) - \tan^{-1}\left(\frac{\sqrt{(1-\rho^2)}}{\rho + l_1/l_2}\right) \tag{2.73}$$

$$\theta_2 = \cos^{-1}(\rho) \tag{2.74}$$

$$\rho = \frac{(x^2+y^2) - (l_1^2 + l_2^2)}{l_1 l_2} \tag{2.75}$$

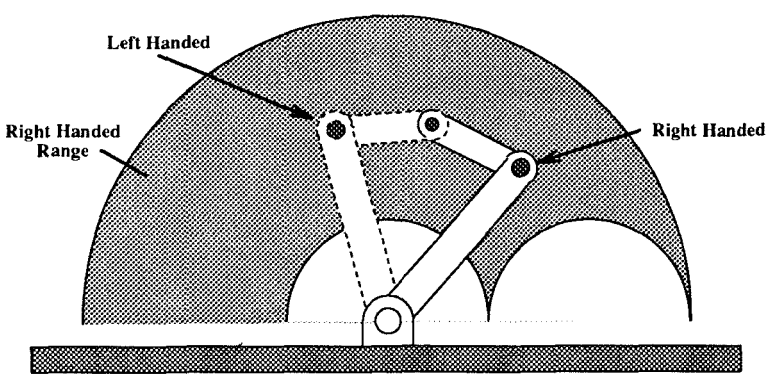



Fig. 2.27 - Right- and Left-Hand Degenerate Configurations

This refinement of the task acts to reduce the operating regime over which the network must effectively learn to control the manipulator. By reducing the amount of information the network must learn, we may simplify the learning process.

#### 2.3.5.2 Feedback Through Trajectory Modification

It became clear in simulation that neural network feedforward control alone is not sufficient. Because of the inherent instability of feedforward control, even the exact equations governing the simulated motions of the arm could not be used effectively in simulation over long periods of time. Clearly then, feedback is required to maintain adequate control.

Although we could have introduced more conventional linear or adaptive feedback elements to correct for errors in operating the plant, we used feedback through trajectory modification. Let the solid line in Fig. 2.28 represent the desired trajectory through state-space, and the dashed line the actual trajectory. If we assume that the feedforward controller will adjust the plant inputs in the proper direction given changes to the desired outputs, then by modifying the desired trajectory (dotted-line) to lead from the actual current position in state-space to the desired trajectory, we introduce corrective action to move back towards the desired trajectory.

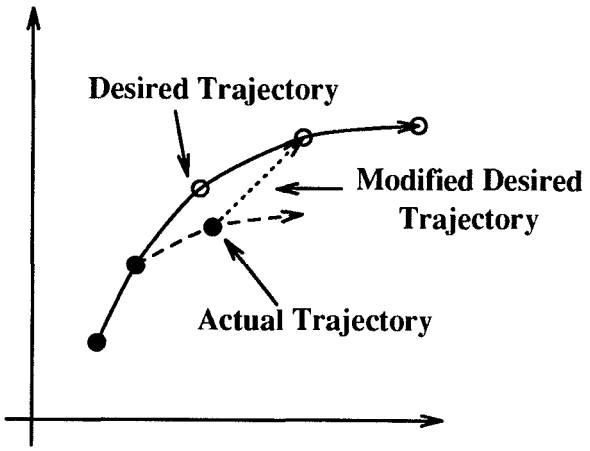



Fig. 2.28 - Trajectory Modification

We modify the trajectory by splitting the tapped delay line of Fig. 2.11 at the position representing the current time. Actual measurements of the manipulator output are provided to the network as shown in Fig. 2.29 at taps representing the output at points up to and including the present. Points representing future

points are still taken from the desired trajectory. This procedure, in effect, presents a modified desired trajectory $\hat{y}_d$ to the network based on the desired and actual trajectories $y_d$ and $y$. For the current task, this procedure feeds the actual joint angles and velocities directly back from the plant into the network and generates a modified desired angular acceleration based on the future desired angle and the current actual angle and velocity as follows:

$$\hat{y}_d(t) = y(t) \tag{2.76}$$

$$\hat{\dot{y}}_d(t) = \dot{y}(t) \tag{2.77}$$

$$\hat{\ddot{y}}_d(t) = \Delta_t^{-2}(y_d(t + 2\Delta_t) - y(t)) - 2\Delta_t^{-1}\dot{y}(t) \tag{2.78}$$

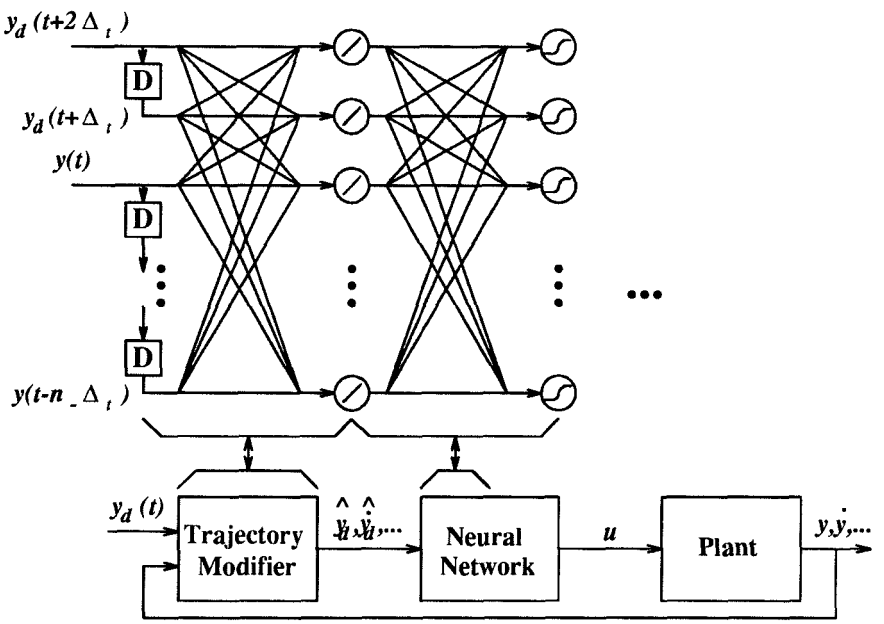



Fig. 2.29 - Trajectory Modification Using Tapped Delay Line

The sudden change from desired to actual joint angles in the tapped delay line could lead to rather abrupt changes in joint angles, velocities, and/or accelerations, this was not observed during simulation. Nevertheless, in order to reduce the abruptness of any corrective action, we may modify the trajectory more gradually over a number of timesteps. The following equations show how we can modify the

trajectory with a desired linear reduction of the error in time:

$$\tilde{\underline{x}}_d[n] = \begin{cases} \underline{x}[n] & n \leq n_0 \\ \underline{x}_d[n] + \frac{n-n_0}{N}(\underline{x}[n_0] - \underline{x}_d[n]) & n_0 < n < n_0 + N \\ \underline{x}_d[n] & n \geq n_0 + N \end{cases} \tag{2.79}$$

Third, we utilize a more advanced training algorithm to dynamically choose the network architecture as it learns to minimize the number of resources required to learn mappings within certain error bounds. This algorithm allows us to avoid costly trial-and-error runs to determine which network architecture from a given class provides near-optimal performance.

#### 2.3.5.3 Dynamic Choice of Architecture

In addition to reducing the operating regime over which the network must learn to control the plant, we utilize a more advanced training algorithm, Algo-A1, in this task. This algorithm is described in Subsec. 1.5.11 and dynamically adjusts the number of neurons and connections in order to minimize the number of resources required to control the plant within specified error bounds. Algo-A1 allows us to avoid costly trial-and-error runs to determine which network architecture chosen from a given class provides near-optimal performance.

We performed off-line training on a trajectory tracing out a handwritten "p." The resulting network had three neurons in 8 of the 10 runs. Figure 2.30 demonstrates the ability of the network to draw the "p" for which it was trained, as well as to draw an "m", an "o", and a line. The resulting network was compared with BEP networks with fixed sizes of 2, 6, 10, and 20 units. The networks with large numbers of units failed with consistently higher frequency.

The networks had the most difficulty adjusting to drawing characters at different positions. In order to expand the range of the neural controller, the incremental learning technique, Algo-I2, was used. The network learned five separate subtasks–drawing the character "p" at five locations as shown in Fig. 2.31a. Fig. 2.31b

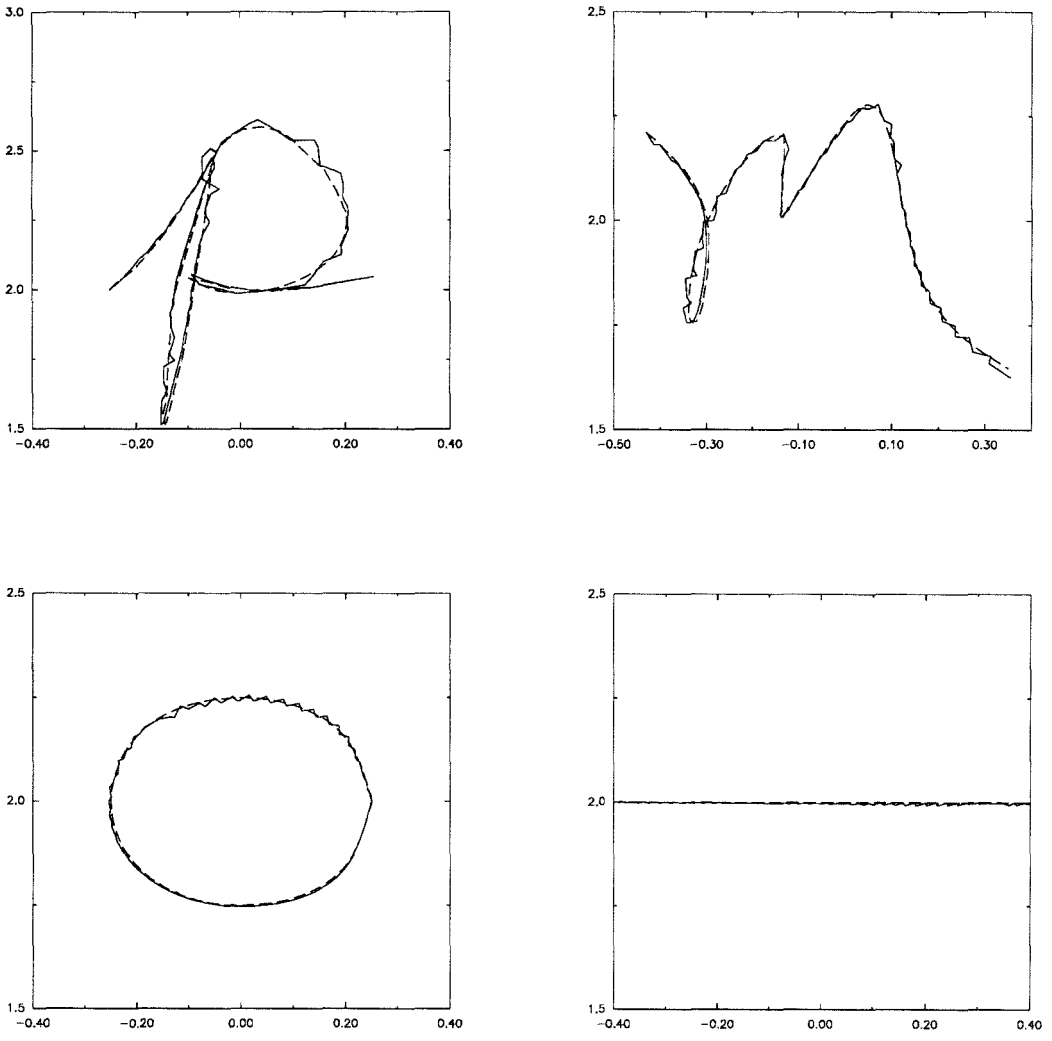



Fig. 2.30 - Feedforward Trained Handwritten Characters

demonstrates the ability of the resulting network to draw other shifted characters as well.

#### 2.3.5.4 On-Line Learning and Feedback Training

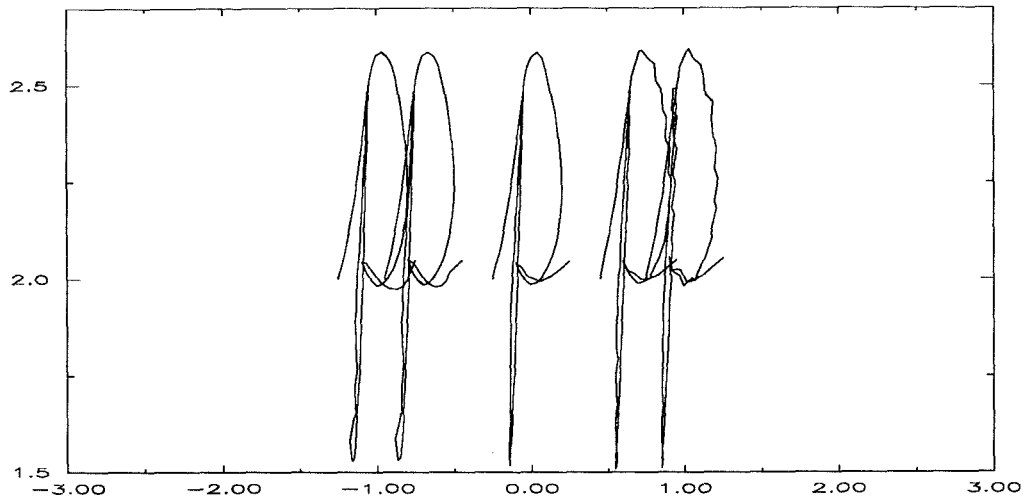
In this task, we also experiment with making the training more "realistic" by simulating on-line learning and adaptation. We claim that one of the strengths of neural network control is that instead of requiring accurate plant models to derive control laws, the networks can be trained to control plants by experimenting with

Table 2.4 - Generalization vs. Number of Hidden Units

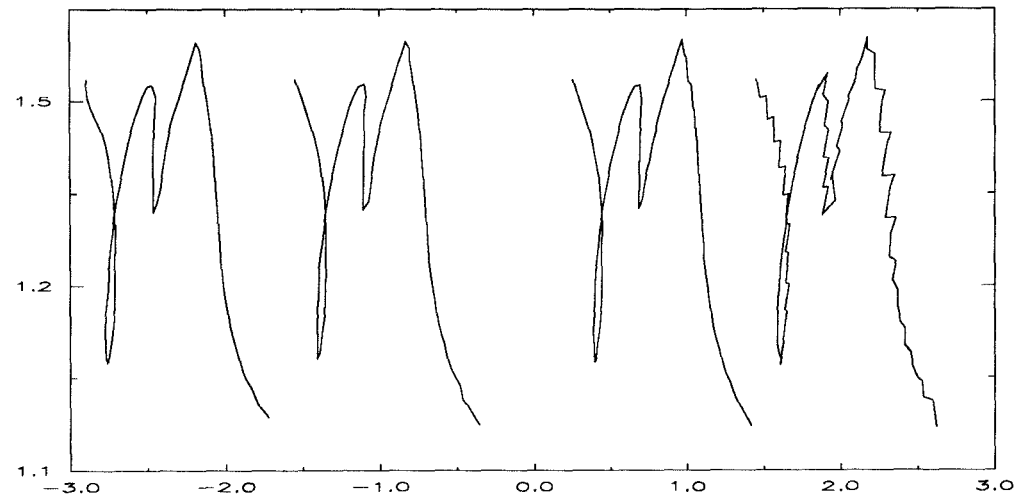
| Character | # Units (Algorithm) | # Failures/10 | Av. Sq. Err. |
|---|---|---|---|
| P (trained) | 3 (Algo-A1) | 0 | .005 |
| | 2 (BEP) | 0 | .004 |
| | 6 (BEP) | 2 | .011 |
| | 10 (BEP) | 0 | .046 |
| | 20 (BEP) | 9 | |
| M | 3 (Algo-A1) | 0 | .005 |
| | 2 (BEP) | 0 | .003 |
| | 6 (BEP) | 2 | .010 |
| | 10 (BEP) | 0 | .009 |
| | 20 (BEP) | 7 | |
| Circle | 3 (Algo-A1) | 0 | .004 |
| | 2 (BEP) | 0 | .002 |
| | 6 (BEP) | 0 | .010 |
| | 10 (BEP) | 1 | .011 |
| | 20 (BEP) | 7 | |
| Line | 3 (Algo-A1) | 0 | .004 |
| | 2 (BEP) | 0 | .003 |
| | 6 (BEP) | 2 | .012 |
| | 10 (BEP) | 1 | .013 |
| | 20 (BEP) | 6 | |

the plants, observing the effect on the outputs of changing the inputs, to learn how to operate the plants. With off-line learning in the preceding tasks, we assumed that we could somehow maneuver the state of the plant, if it existed, to the operating points corresponding to the individual training points. However, this may not be possible without an effective means of controlling the plant. Here, we consider on-line training techniques that enable us to effectively learn to control certain plants without the ability to arbitrarily set their state.

We will assume that some method exists for resetting the plant to a known state, and that the desired trajectory begins at that point in state-space. As in the previous subsubsection, we feed back the actual plant outputs to the network input. Although we choose to split the delay line as before, we could in general provide the network with parallel delay lines containing both desired and actual




(a) Training




(b) Testing

Fig. 2.31 - Incrementally Trained Handwritten Characters

trajectories. Because the actual trajectories can deviate from the desired, training may occur outside the desired operating region. Further, on-line training may not occur around a given point on the desired trajectory unless we have learned enough of the preceding part of the trajectory to move the manipulator to that point. If we

use algorithms such as BEP, we are using gradient descent algorithms that reduce the error at the training points. With continuous neurons (*e.g.*, sigmoidal instead of step) and a continuous desired control law, the change in error for a given change in weights will be continuous as well; thus, the error will be reduced in a neighborhood of each training point. Heuristically speaking, as we train on-line using feedback training, the actual trajectory will deviate from the desired one, thus the actual training points will differ from the desired ones. However, if each training point is within a sufficiently small neighborhood of the desired one, the error reduction at the actual training point will carry over to the desired one. Thus, we hope that on a subsequent feedback learning attempt, the actual trajectory will move closer to the desired trajectory. In feedback learning, we set an error threshold such that if an actual training point deviates from a desired training point by an amount exceeding the bound, the system is reset and learning continues from the beginning of the trajectory. This reset serves two purposes: one, it hopefully prevents the network from training at points too far from the desired training points to do any good at the desired points, and two, it hopefully prevents the system from "going out of control" or becoming damaged. In some sense, the system might always be considered "stable" since it resets before becoming "unstable." However, the system is not guaranteed to converge (*i.e.*, learn the entire trajectory within a given error bound).

We trained neural networks using feedback learning with the same parameters as in the previous subsubsection (*e.g.*, dynamic modification of network architecture, *etc.*). Instead of Algo-A1, we used BEP with seven hidden units. We found that although the number of iterations required for convergence was typically a factor of three to five larger, on-line feedback learning worked as well as off-line learning. Fig. 2.32 shows the characters traced by the manipulator when operated by the network trained on-line.

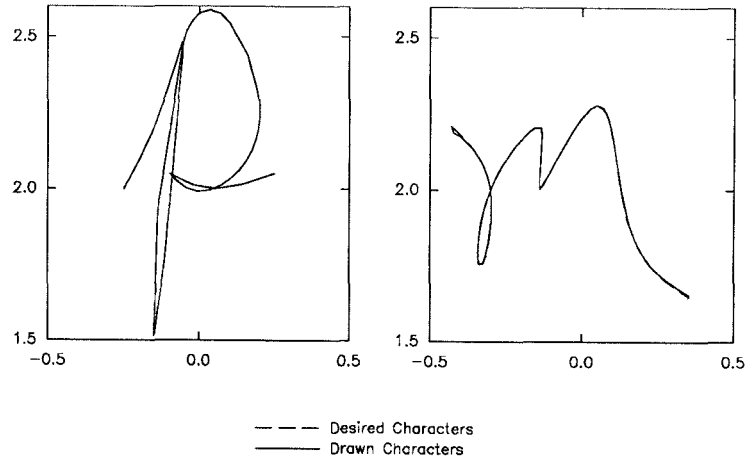



Fig. 2.32 - Test of Neural Network Controller Trained On-Line

2.3.5.5 GENERALIZATION

Figure 2.32 also shows the ability of a neural network, trained to draw the letter "p," to draw the letter "m." Note that the characters are drawn very closely to the desired characters. Figure 2.33 shows the ability of the neural network to generalize to drawing the same characters at different speeds. Here, a network trained to draw the character "p" at one speed is tested in its ability to draw the letter "o" drawn at four times that speed. We see that once again, the network demonstrates a good ability to generalize. Figure 2.34 shows the relative sizes of the angular acceleration $\tau_M$, centrifugal and Coriolis $\tau_v$, and gravitational $\tau_g$ contribution to the total desired output torque. We see in fact that the dynamic terms are indeed significant in comparison to the gravitational term.

Figure 2.35 shows the ability (or inability) of the network to generalize well to shifts in the position of the character drawn. Specifically, a network trained to draw the letter "p" is tested in its ability to draw shifted "p"s and "m"s. Although we have not established a good metric for measuring the closeness or similarity of two trajectories, we believe that the trajectories tracing out different characters drawn

at different speeds at the same location as the original must somehow be closer to the original trained trajectory than the same character drawn at the same speed at a different location. When such a metric has been established, it may provide help in selecting training sets and estimating how well a new character will be drawn based on its similarity to previously learned characters.

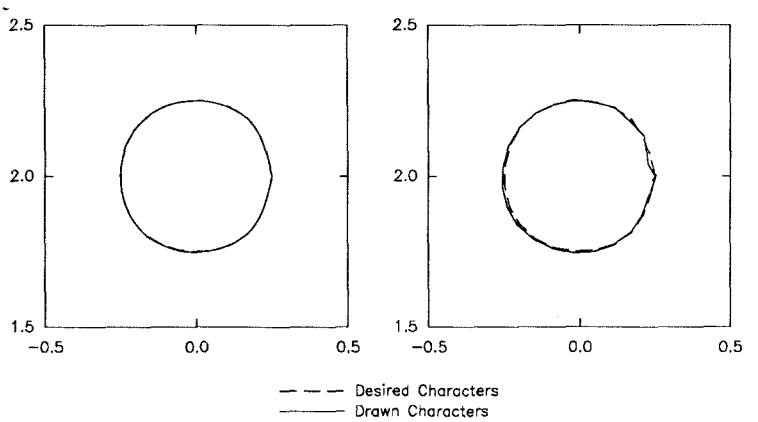



Fig. 2.33 - Generalization to Different Speeds

#### 2.3.5.6 Incremental Learning

We were also able to demonstrate on-line incremental learning. Once again, we trained a network to draw the letter "p" in a restricted region of the robot's range. Then, by concurrently extending the training set and increasing the resources available, we were able to train it to draw letters in other regions of space as shown in Fig. 2.36

## 2.4 Future Directions

In this chapter, we demonstrated that BEPing through the plant does work when using perturbation of the inputs to measure the plant Jacobian. However,

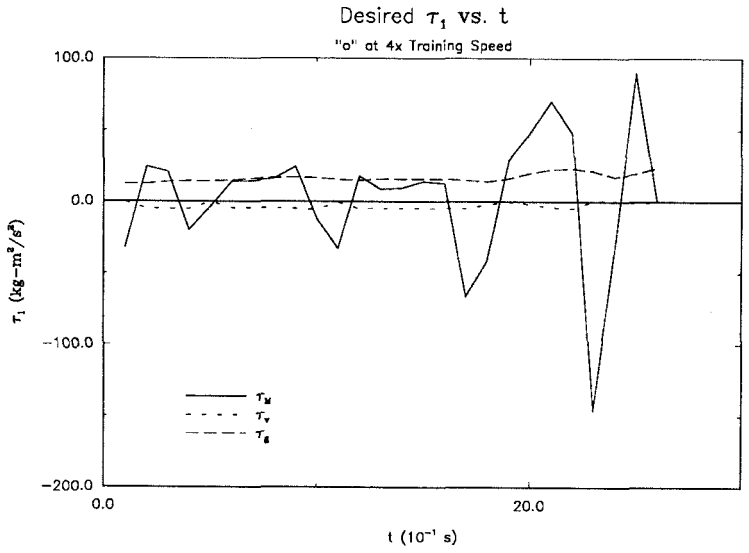



Fig. 2.34 - Contribution of Dynamic Terms to Total Torque.

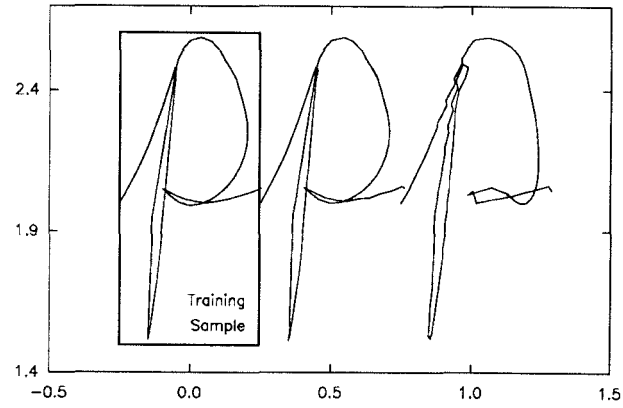



Fig. 2.35 - Generalization to Different Positions

such perturbation is not as attractive a technique for measuring the Jacobian as some of the other parameter estimation techniques such as the least squares and

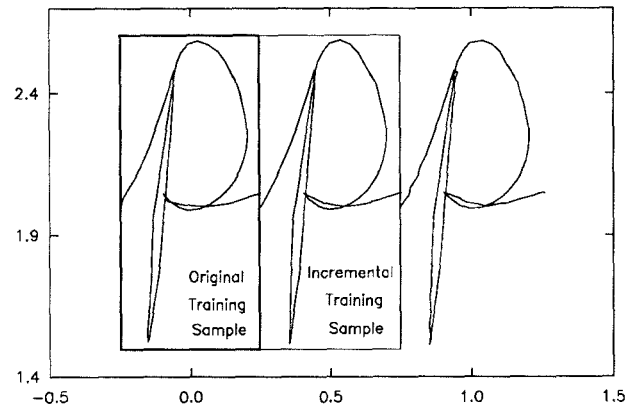



Fig. 2.36 - Incremental Learning

projection algorithms decribed in the chapter. It would be useful to test these methods for estimating the plant Jacobian as well.

If the convergence time of the plant Jacobian estimates is longer than the time constants of the time-varying Jacobian itself, then BEPing through the plant may no longer work. Thus comparison of different parameter estimation techniques in terms of convergence time versus accuracy will be important in determining which technique to use for controlling a given plant.

Feedback is an important part of any control system. The use of adaptation and learning in the feedback path needs to be explored as well. Some researchers are already beginning to do this.[20]

Many researchers are already studying the application of neural networks to other aspects of the control problem such as sensory processing and planning, *etc.*. Finally, more actual implementations of neural control systems are required to find and focus attention on any problems with current neural control techniques.

## 2 References

[1] *Webster's Dictionary*, J.G. Allee, ed., Ottenheimer Publishers, USA, p. 86, 1975.

[2] G.F. Franklin, J.D. Powell, and A. Emami-Naeni, *Feedback Control of Dynamic Systems*, Addison-Wesley Publishing Company, Menlo Park, CA, 1986.

[3] K. Ogata, *Modern Control Engineering*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1970.

[4] K.J. Astrom and B. Wittenmark, *Adaptive Control*, Addison-Wesley Publishing Company, Menlo Park, CA, 1989.

[5] D. Psaltis, A. Sideris, and A. Yamamura, "A Multilayered Neural Network Controller," *IEEE Control Systems Magazine*, Vol. 8, pp. 17-21, 1988.

[6] J.S. Albus, "A new approach to manipulator control: The cerebellar model articulation controller (CMAC)," *Transactions of the ASME Journal of Dynamic Systems, Measurement, and Control*, Vol. 97, pp. 220-227, 1975.

[7] B. Widrow, J. McCool, and B. Medoff, "Adaptive Control by Inverse Modeling," Twelfth Asilomar Conference on Circuits, Systems, and Computers, 1978.

[8] C.G. Atkeson and D.J. Reinkensmeyer, "Using associative content-addressable memories to control robots," *Proceedings of the IEEE Conference on Decision and Control*, pp. 791-797, 1988.

[9] M. Kuperstein, "Neural model of adaptive hand-eye coordination for single postures," *Science*, Vol. 239, pp. 1308-1311, 1988.

[10] W.T. Miller, F.H. Glanz, and L.G. Kraft, "Applications of a general learning algorithm to control of robotic manipulators," *International Journal of Robotics Research*, Vol. 6, pp. 84-98, 1987.

[11] Kawato, Furukawa, and Suzuki, p. 226, 1987.

[12] S. Hosogi, "Manipulator Control Using Layered Neural Networks," 1988.

[13] D. Rumelhart, seminar at the California Institute of Technology, March, 1987.

[14] M.I. Jordan and D.A. Rosenbaum, *COINS Technical Report 88-26*, Department of Computer and Information Science, University of Massachusetts, Amherst, 1988.

[15] M.I. Jordan, "Sequential Dependencies and Systems with Excess Degrees of Freedom," *COINS Technical Report 88-27*, Department of Computer and Information Science, University of Massachusetts, Amherst, 1988.

[16] M.I. Jordan, "Generic Constraints on Underspecified Target Trajectories," *Proc. IJCNN*, San Diego, June 1989.

[17] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, 1985.

[18] B. Widrow, "Adaptive Inverse Control," *Adaptive Control in Control and Signal Processing*, Inst. of Feder. of Autom. Control, Pergammon Press, Lund, Sweden, July 1986.

[19] Nguyen and Widrow, "The Truck Backer-Upper: An Example of Self Learning in Neural Networks," *Proc. IJCNN*, San Diego, pp. 357-363, June 1989.

[20] Kawato, *Proceedings of the IJCNN*, 1991.

# 3 Parallel Readout from Optical Disks

## 3.1 Introduction

Optical disks have developed into a mature commercialized technology for information storage. Magnetic storage technology presents the main source of competition for optical disk technology in those applications where optical disks are now utilized. Although optical disk technology has established itself as a viable technology by exploiting many of its advantages over magnetic storage, none of the commercially available optical disk systems exploits one of the strongest potential advantages of optical storage technology–namely optical parallel access.

In this chapter, we begin by briefly reviewing optical disk storage technology and applications. We then characterize the SONY sampled-format optical disks including those aspects relevant to the parallel readout of information from the disks. Next, we discuss techniques for storing information on these disks and reading it out in parallel and present experimental results. Finally, we describe a number of potential applications for the parallel readout of information from optical disks.

### 3.1.1 Commercial Optical Disk Applications

Surprisingly, optical disks have a long history of commercialization. In 1935, Baird Radiovision offered a wax disc displayed by an optical scanner.[1] Today, optical disks come in a variety of forms, the most ubiquitous of which is the digital audio compact disc or CD. Together with the consumer video LaserDisc introduced in 1978, these two products represent the most well-known forms of optical storage

available today. For the storage of computer accessible archival information, the CD-ROM (an adaptation of the digital audio CD) has been developed. These three forms are all read-only disks–that is, the information is prerecorded on the disk and cannot be altered by the user. Write-once read-many or WORM disks have been available for a number of years, but have not been as popular as the erasable optical disks available today for the storage of computer information.

All these commercial applications of optical disks have achieved some degree of success in the marketplace by exploiting some of the advantages of optical storage *vis-a-vis* competing technologies in the niche of high-density archival storage. Currently, the main competition in these applications are the phonograph, compact audio cassette tape, and digital audio tape, in consumer audio; videotape in consumer video; and magnetic disk and tape in computer storage applications.

Optical disks now hold an edge over magnetic disks in terms of storage density. The resolution of any optical system is essentially limited by the wavelength of the light used in the system. The shorter the wavelength, the higher the resolution. In current systems, this is on the order of 1 $\mu m^2$ per pixel of information. The resolution in current magnetic systems is limited by the "flying height"–the distance between the head and the disk. Commercially available systems have flying heights of about 200 $nm$, although magnetic storage with a flying height of less than 50 $nm$[2] yielding storage areas of 4 $\mu m \times .16\ \mu m$ has been demonstrated in lab. This storage density is comparable to that of available optical disk systems.

Although a number of paths exist for increasing the storage density of optical disks, they are still many years from commercialization. The most obvious path–reduction of wavelength–is limited to gains of a factor of four or so. Wavelength multiplexing is also being investigated.[3] Superresolution techniques have been proposed and are being studied as well.[4] Perhaps the most promising path is volume recording with layers of pixels or volume holograms.[5] Reduction of flying height is

still the primary means of increasing storage density of magnetic disks. However, to achieve smaller flying heights, the study of disk tribology (the effects of friction and methods for obviating them) becomes of primary importance.

With subsubmicrometer flying heights, contaminants and mechanical vibration can cause serious problems, and high-density magnetic media must remain in fixed locations. However, because optical lenses exist that can be used to transfer information between the storage medium and an optical detector(/modulator), an optical read(/write) head can "fly" high above the disk, typically 1 $mm$, and still access information at the highest available resolution. Because information can be accessed at a distance, it is also possible to include a protective layer between recording medium and the outside world, and the optical disk is not as sensitive to contamination and vibration. Thus, optical disks have a decided advantage in applications requiring removable or transportable storage media.

On the other hand, a disadvantage of current optical disks is their slower access times and data rates relative to magnetic disks. This is a direct result of the larger size and weight of optical heads (with laser sources, beamsplitters, lenses, and detectors) which are more difficult to position. Thus work on reduction of the size, weight, and number of components and servo improvement is the direction in which the most effort has been focused to decrease access times.

Because magnetic storage is not diffraction limited, it will likely eventually surpass wavelength-limited optical storage through continuing incremental improvements in storage density. Electronic storage in the form of DRAMs is also approaching similar resolutions with 1.28 $\mu m^2$ capacitors for storage in 64 $Mbit$ DRAMs.[6] Thus for practical applications, research on optical disks must concentrate on those areas in which optics has potential advantages in the long term: rugged, transportable, archival storage; high density through superresolution or volume holography; and optical parallel access as discussed in this chapter.

### 3.1.2 Optical Disk Recordability

The different types of optical disks can be categorized in a number of ways– three important such ways being the *recordability* or whether information can be recorded and/or erased on the disk, the *media* or the structure of the material which makes up the disk, and the *format* or the way in which data is stored on the disk.

One way to classify optical disks is according to "recordability" or whether and how many times data can be recorded and rerecorded *in situ* at the same location on the disk. If the data is recorded once at the "factory" and never recorded in the field, it is a *ROM-Disk* (Read-Only Memory). If the data can be recorded just once at each disk location, it is a *WORM-Disk* (Write-Once Read-Many). Finally, if the data can be recorded and rerecorded multiple times, it is an *Erasable Disk*. Although the erasable disk provides the greatest flexibility, there are other applications such as database record storage for WORM disks or storage of prerecorded audio or visual information for ROM disks where erasability is not required or even not desired.

### 3.1.3 Optical Disk Storage Media

A large number of materials have been considered for optical recording. For ROM disks, injection molded stamped plastic substrates with a sputtered layer of metal for reflection probably best satisfy the requirement for low cost, mass producibility, and stable long-term storage. Disks of this type are currently used for both the CD and LaserDisc.

Both WORM and erasable disks require media whose physical properties can be altered using light. Bartolini, *et al.*,[7] have divided such media into 11 classes. Media for WORM disks typically fall into one of the following four classes: photographic, photoresist, photopolymer, or ablative. Photographic materials consist of emulsions in which a photochemical reaction changes the optical density (transmissivity) of the material upon development; chemical bleaching can convert this

into an index change. Photoresists are organic materials sensitive to light such that either exposed or unexposed areas are removed upon development depending on whether the medium is a positive or negative photoresist, resulting in thickness (phase) variations in the medium. Photopolymers are organic materials whose index can be changed upon exposure to light, possibly after a separate developing step. Ablative media are either metal or organic and rely on localized heating by a focused laser beam to melt or ablate the medium to vary the reflectivity and/or transmittivity.

Erasable disks typically fall into one of the following seven classes: thermoplastic, photochromic, chalcogenide, magnetooptic, photoferroelectric, photoconductive-electrooptic, and electrooptic. Thermoplastic consists of a conductor, a photoconductor, and a thermoplastic; after establishing a charge across the photoconductor and thermoplastic, illumination selectively discharges regions via the photoconductor, and upon heating, the plastic deforms according to static force variations. Photochromic materials are those which have two stable states with different optical properties with optical switching from one state to the other. Chalcogenides materials can be switched optically between crystalline and amorphous states having different optical properties. Magnetooptic media rotate the polarization of a transmitted or reflected readout beam through the Faraday or Kerr effect respectively; the angle of rotation is determined by the magnetization which is determined by an applied field upon heating using a focused laser beam. Photoferroelectric media consist of ferroelectric crystalline and photoconductive layers; an incident light pattern striking the photoconductor modulates the field across the ferroelectric thus resulting in modification of the polarization of transmitted or reflected light. Photoconductive-electrooptic media are sandwiched between charged insulators that are selectively discharged through illumination; the resulting charge distribution produces variations in the retardation of the medium. Electrooptic media are those such that

carriers are generated in illuminated areas and diffuse into dark regions establishing a charge distribution producing variations in the index of the medium. These media are also called photorefractive.

### 3.1.4 Optical Disk Storage Formats

There are many ways to classify disk storage formats. For instance, if the disk is spun at a constant velocity and data accessed with a constant bandwidth, the amount of information stored on the disk will be constant per unit angle resulting in a constant angular velocity (CAV) disk. If the angular velocity of the disk or the data bandwidth depend on the radius of the stored information on the disk such that the amount of information stored on the disk is constant per unit length, we have a constant linear velocity (CLV) disk. A CAV disk results in a simpler mechanism since the disk spins at a uniform velocity but CAV disks waste storage space since they pack information less densely at larger disk radii.

Information recorded on the disk can be encoded in different ways as well. Each bit of information can correspond to a pixel on the disk, or the information can be recorded by pulse code modulation as in the CD.

### 3.1.5 Desirable Characteristics for Parallel Access

Considerations for optimal characteristics of disks for parallel access are very much the same as those for serial access; high storage density, the stability of the medium, *etc.*, are important concerns whether we wish to access information serially or in parallel. However, in a world where laser power limitations will often play a significant role, the recording sensitivity of the disk medium and the readout reflectivity or transmissivity, contrast, and fill-factor take on greater importance where parallel access is concerned.

Data recorded on the disk for parallel readout may be organized in one of three possible natural formats as shown in Fig. 3.1. In the along-track format (Fig. 3.1a),

data is recorded azimuthally on the disk in the direction of disk rotation. This format allows a parallel readout head aligned in the same direction to view shifted versions of the same data in time as the disk rotates. This form of parallel readout may be useful for applications where the correlation or convolution of data recorded on the disk with a sample pattern is desired (see Sec. 3.4).

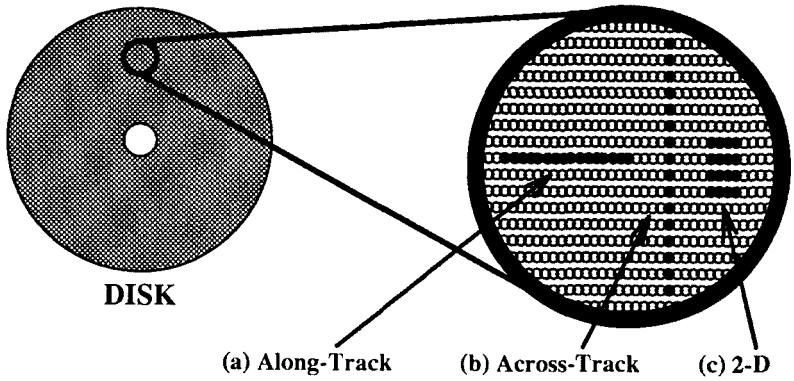



Fig. 3.1 - Data Organization for Parallel Readout

In the across-track format (Fig. 3.1b), data is recorded radially on the disk in the direction perpendicular to that of disk rotation. This format allows each element of a parallel readout head aligned in the same direction to view a different set of data in time as the disk rotates. This form of parallel readout maximizes potential bandwidth since each element in the readout head can read out an independent stream of data.

In the 2-D format (Fig. 3.1c), data is recorded in a two-dimensional array with one axis of the data along-track and the other across-track. This format is best-suited for data which is naturally two-dimensional in nature, such as images, matrices, *etc.*.

It is necessary for data recorded in the across-track format and highly desirable for data recorded in the 2-D format that recorded pixels can be aligned radially on the disk–a property called *across-track coherence*. Across-track coherence is not a requirement for serial recording and readout, and is not guaranteed in commercially

available disk recorders which are designed for serial readout applications. Fig. 3.2a shows pixels recorded by a typical disk system. Note that pixels are not aligned across tracks. In fact, the drift between corresponding pixels in different tracks can easily reach several pixels in size. Fig. 3.2b shows that pixels recorded on the Sony sampled-format disk system are in fact aligned across track. This across-track coherence is not designed for parallel readout purposes; rather, it is a byproduct of the data format required to maintain tracking in a sampled-format system as described in Sec. 3.2.3.

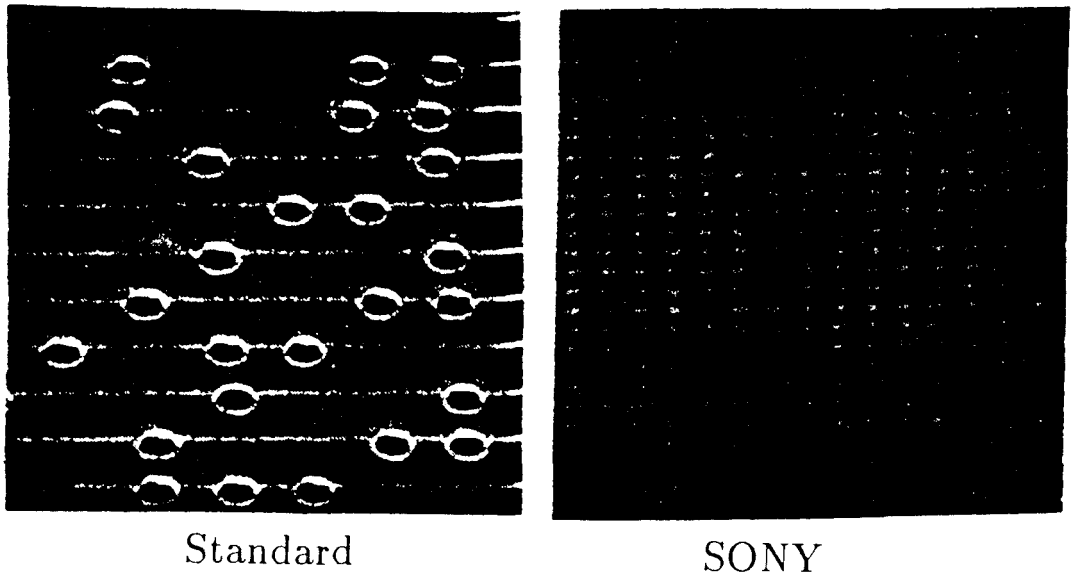



Fig. 3.2 - Across-Track Coherence

### 3.1.6 Organization of Data for Parallel Readout

Given a large memory storage capacity and a number of channels with which to access it in parallel, we could conceivably allow each channel to independently and arbitrarily access any of the memory storage locations. With $N$ locations on the disk and $M$ channels, arbitrary access would require a readout system with $N^M$ distinct configurations. With $N = 10^{10}$ bits on the disk, we see that arbitrary access is only possible with small-scale parallel readout (small $M$) before the sheer number of configurations becomes unmanageable.

In order to achieve large-scale parallel readout with $M \tilde{} 1000$, we must restrict the allowed mappings between memory storage locations and readout channels. In parallel block access, we impose a fixed relationship between storage locations that can be accessed in parallel at a given time. For example in 2-D block access, we only allow access to a rectangular neighborhood of fixed dimensions in the 2-D array of pixels recorded on disk. In block access, by specifying the correspondence between a single storage and detection element, we specify the correspondence between all others as well. Thus, we require a readout system with $N$ distinct configurations. In fact, if each storage location can only be accessed by a single specific detector element, we require a readout system with only $N/M$ distinct configurations. This scheme is also called paged readout. A large number of 2-D holographic paged readout schemes proposed in the past.[8,9]

### 3.1.7 Mechanical Scanning

Although non-mechanical optical scanning may provide the fastest possible access time, it requires the use of optical components with space-bandwidth product large enough to view the entire optical disk. In addition, optical scanning techniques often require the consumption of space-bandwidth product on the disk to encode diffractive carriers required to keep the readout light within the optical system. Mechanical scanning by either spinning the disk or radially translating the head simply provides a method for drastically reducing the requirements on optical system and the diversion of disk storage area for readout purposes.

### 3.1.8 Parallel Readout

A number of authors have proposed using various parallel readout techniques with optical disks. Many of these have proposed 1- or 2-D holographic recording and readout using disks with photographic film.[10–13] Because of the limitations of optical disk technology at the time, most of these proposed systems were intended

for limited applications mainly involving the readout of prerecorded images. More recently, other authors have proposed both holographic[14] and imaging[15] readout from pixellated disks. However, none have discussed in detail, imaging and holographic encoding techniques appropriate for use with the high-density, pixellated, binary optical disks available today; nor have they discussed the alignment of the optical disk and detector. These are the areas that will be discussed in the remainder of this chapter.

## 3.2 Characterization of the SONY Sampled-Format Disk

We use a prototype Sony sampled-format disk recording and playback system (Fig. 3.3) to write information on the disks used in our experiments. The disk system itself is interfaced to a PC which provides serial read/write access to the disks. It can utilize both write-once and erasable storage media which are described in greater detail in Subsec. 3.2.2. The disks are spun at 2400 *rpm* and can read or write data at 15 *Mbps*.

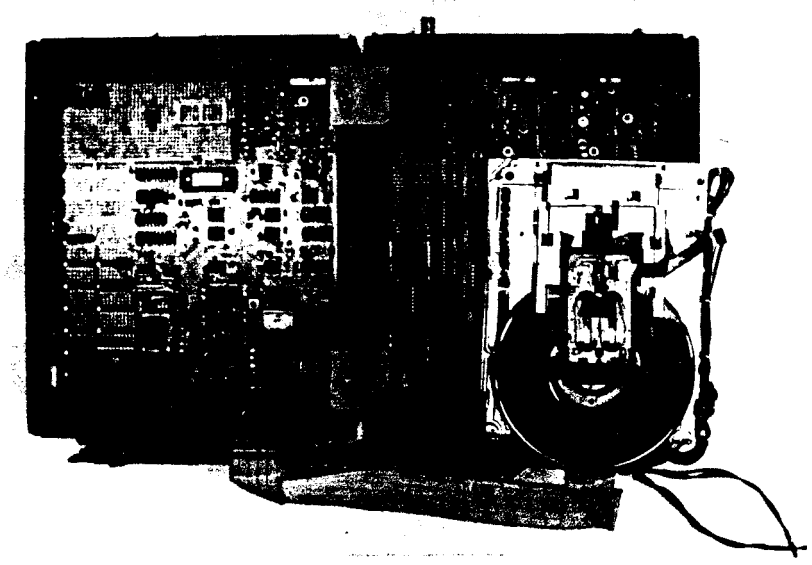


Fig. 3.3 - Prototype Sony Sampled-Format Disk System

### 3.2.1 Physical Characteristics of the SONY Disks

The disks themselves are $5\frac{1}{4}$" (12 $cm$) in diameter. The disks can be single- or double-sided because both the write-once and erasable media operate in a reflective mode. Both types of disk are formed on a protective substrate of glass or polycarbonate (PC).

### 3.2.2 Storage Media of the SONY Disks

The Sony sampled-format disk system is capable of serial recording on and readout from both write-once and erasable disks. The actual storage media for the two disk types are described below. Because of the relatively high energies required for recording compared to lower energies for readout, parallel recording of millions of pixels is at present infeasible with the types of disks used in our experiments; thus the experiments in this chapter are restricted to those concerning only parallel readout of data.

#### 3.2.2.1 SONY Write-Once Storage Medium

The storage medium of the SONY write-once disks that we use in our experiments is rather unusual. These disks are the same as those described in detail in Ref. [16] except that the disks that we use do not have a guide-groove. The medium consists of a layered structure as shown in Fig. 3.4. The bottom layer is 1000 Å thick and consists of a reflective aluminum backing. The next layer is 1400 Å thick and consists of $Sb_2Se_3$. Next comes a 150 Å layer of $Bi_2Te_3$ followed by another layer of $Sb_2Se_3$ 300 Å thick. Finally comes a 1.2 $mm$ layer of protective polycarbonate or glass.

Table 3.1 shows the refractive index and absorption of each of the above materials at the laser diode wavelength of 860 $nm$ for which the disk is optimized. When a read beam strikes an unwritten portion of the disk from above, part of the beam is reflected at each interface due to discontinuities in the refractive index.

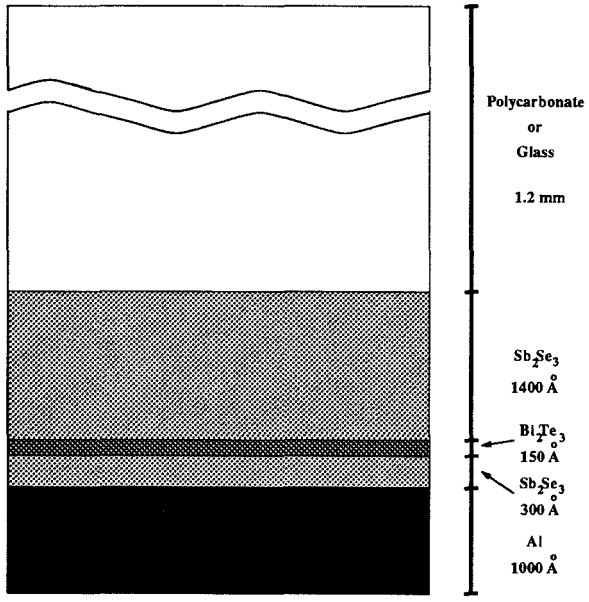



Fig. 3.4 - Sony Write-Once Recording Medium

The thickness of each layer is chosen to create a destructive interference filter at 860 $nm$ thus resulting in a low reflectivity between 12 and 14%.

Table 3.1 - Index and Absorption of Write-Once Disk Materials[16]

| | $Sb_2Se_3$ | $Bi_2Te_3$ | Al | PC |
|---|---|---|---|---|
| n | 3.7 | 4.6 | 2.0 | 1.56 |
| k | .06 | 3.7 | 6.0 | 0 |

Because the $Bi_2Te_3$ layer has a much higher absorptivity than the surrounding layers, as shown in Table 3.1, it experiences the greatest temperature increase upon illumination. At a laser power of 0.8 $nJ/\mu m^2$, a sharp threshold in the material characteristics of the medium is reached where the $Bi_2Te_3$ diffuses into the surrounding layers of $Sb_2Se_3$ sandwiching it. The medium is write-once because this process of diffusion is irreversible. Figure 3.5 shows the simulated change in concentration of $Bi_2Te_3$ and $Sb_2Se_3$ as a function of depth in the medium for various pulsewidths of a 6 $mW$ laser. Figure 3.6 shows the simulated change in index and absorptivity for the same conditions. The sharp index variations in the medium present before recording are no longer present after recording thus effectively de-

stroying the intereference filter resulting in two to four times higher reflectivity after recording.

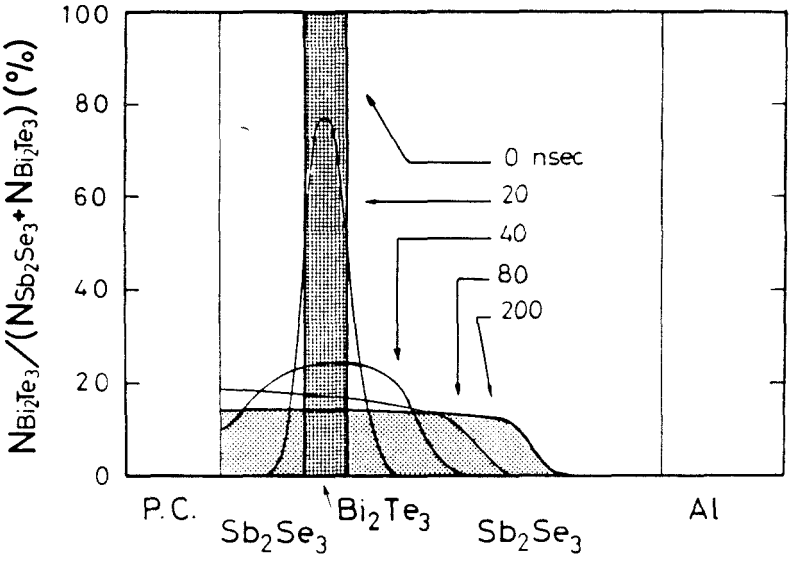



Fig. 3.5 - Metal Alloy Composition After Exposure [16]

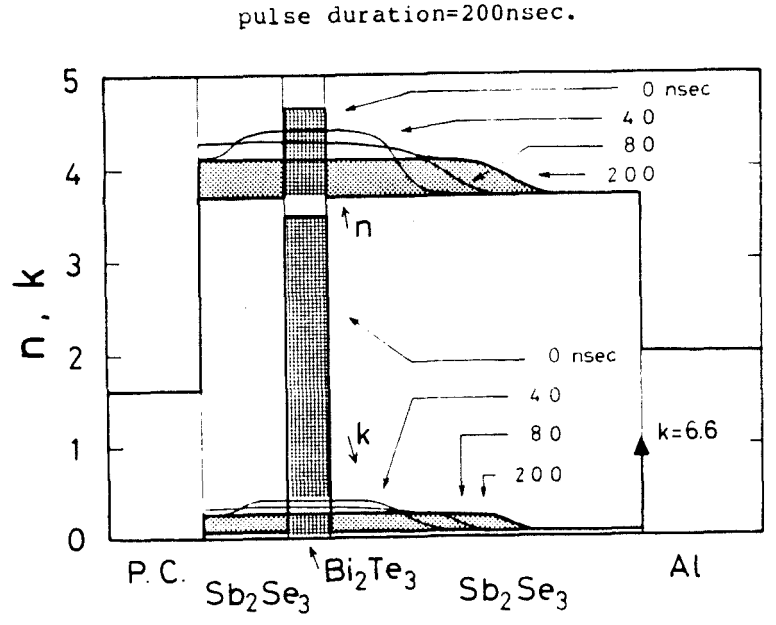



Fig. 3.6 - Index and Absorptivity Variations After Exposure [16]

The thicknesses of the three metal alloy layers are chosen to achieve high reflectivity and sensitivity at 860 *nm*. High sensitivity is achieved with high absorption in the $Bi_2Te_3$ layer, low thermal diffusion, and low heat capacity.

Many of our experiments were carried out using illumination from a He-Ne laser source at 633 $nm$. Because the reflectivity of an interference filter is wavelength dependent, we measured the reflectivity using the optical setup of Fig. 3.7. We collimate a He-Ne laser beam and focus it on the disk using a microscope objective. The size of the focused spot can be estimated by calculating the distance from the center of the first zero in the radially symmetric intensity pattern. Because we are focusing a planewave, we will be approximately one focal length $f = 4.34\ mm$ behind the objective. We estimate the aperture to be $l = 6\ mm$ in diameter. The resulting amplitude distribution will be a Bessel function given by the following equation:

$$A(\rho) \propto \mathrm{J}_1(\pi r l/2\lambda f)/r \tag{3.1}$$

This distribution is known as the Airy pattern and has its first zero at $r = 1.22\lambda f/l_0$ $= .56\mu m$.

When the collimated beam is focused properly on the disk, the reflected beam, passing back through the objective, will also be collimated. We then focus a portion of the reflected beam using a beam-splitter and lens, onto a detector. We calibrate the system using a mirror of known reflectivity, $R_m = .94$. We then measure the reflectivity of recorded and unrecorded portions of the write-once disk. We found the reflectivity of unrecorded portions of the disk to be $R_0 = .049$. Because recorded portions of the disk have a fill-factor less than unity, we perform our measurement indirectly by recording the reflectivity at many spots in a recorded region near the inner radius and calculating the average $R_1'$. We then calculate the fill factor $\alpha$ and solve for the reflectivity $R_1$ as follows:

$$\alpha = \frac{\int_{-\Delta_\theta/2}^{\Delta_\theta/2} \sqrt{(d/2)^2 - x^2}\,dx}{\Delta_\theta \Delta_r/2} \tag{3.2}$$

$$R_1 = R_0 + \frac{1}{\alpha}(R_1' - R_0) \tag{3.3}$$

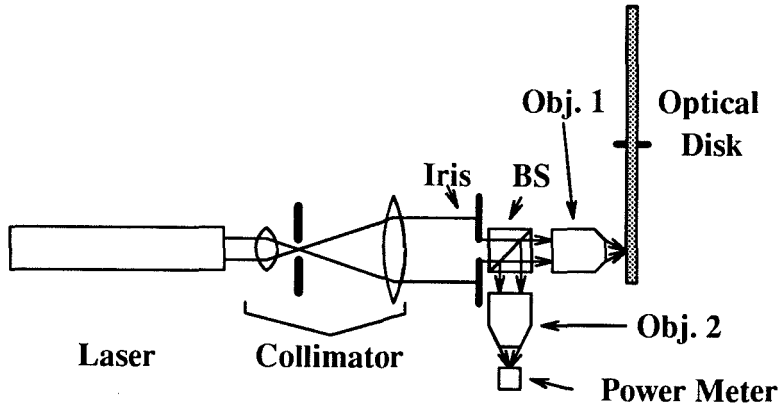



Fig. 3.7 - Experimental Setup to Measure Optical Disk Reflectivity

We found the average reflectivity of recorded portions at the inner radius to be $R_1' = .075$ yielding a reflectivity $R_1 = .090$.

#### 3.2.2.2 SONY Erasable Storage Medium

The erasable medium consists of an aluminum backing with an alloy of TbFeCo as the recording medium covered with a 1.2 $mm$ layer of glass or polycarbonate for protection. The erasable disk, like the write-once disk, operates in a reflective mode. Using the same setup for measuring reflectivity as for the write-once disk, we measured the reflectivity of the magnetooptic disk to be 17% at 633 $nm$. The polarization of light incident upon the disk surface is rotated $\pm.15°$ due to the magnetooptic Kerr effect. The direction of this rotation depends on the magnetization recorded at the spot from which the light is reflected. This polarization rotation is detected through a linear polarizer set such that the amplitudes of the light corresponding to the two states are either (a) plus/minus or (b) on/off[17] as shown in Fig. 3.8.

### 3.2.3 Data Format of the Sony Sampled-Format Disks

Data is recorded on the Sony sampled format disks as circular pixels, 1 $\mu m$ in diameter. As shown in Fig. 3.9, the pixels are recorded along a single spiral on the disk, much like that on a conventional LP (record), extending from an inner radius

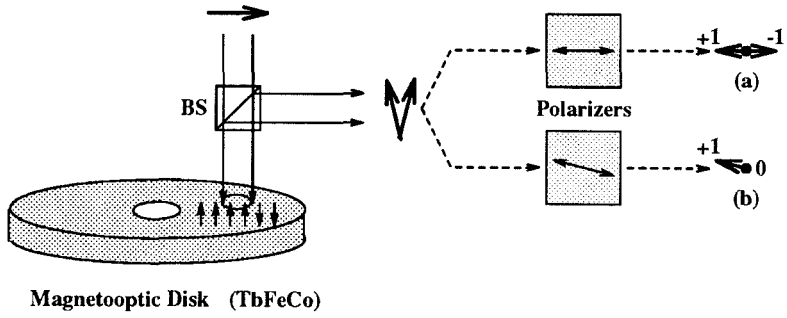



Fig. 3.8 - Magnetooptic Disk

of 3 $cm$ to an outer radius of 6 $cm$. Each turn of the spiral is called a *track* with a track-to-track spacing of 1.5 $\mu m$. Because of the low pitch of the spiral, the tracks can be modeled as concentric circles separated by 1.5 $\mu m$. There are 20 000 circular tracks on each side of the disk. Because the sampled-format disk is intended for CAV systems, the angle between adjacent pixel locations remains a constant $1/1000^{th}$ of a degree, but the linear distance varies from track to track with 1 $\mu m$ between adjacent pixel locations at the outer radius of 6 $cm$ and .5 $\mu m$ between adjacent pixel locations at the inner radius of 3 $cm$. Thus adjacent pixels are tangential at the outer recording radius and overlap by half at the inner recording radius.

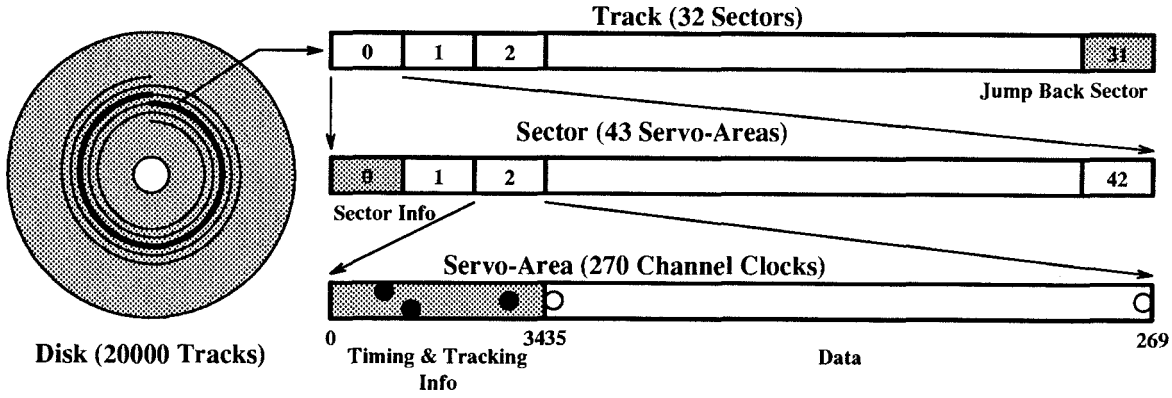



Fig. 3.9 - Pixel Locations on Sony Sampled-Format Disk

The disk is divided, in the azimuthal direction, like a pizza into 32 pie-shaped wedges called *sectors*. Information about the track and sector location is recorded in each track at the beginning of every sector. Each sector is further azimuthally

subdivided into 43 servo-blocks. Each servo-block contains 270 channel clocks or pixel positions. The first 35 pixel positions of every servo-block contains timing and tracking information used by the sampled-format system to maintain alignment between the read/write head and the data on the disk. With these specifications for recording area and pixel separation, the Sony disks provide 6.5 billion pixels per side of the disk. Thus the optical disk provides a high density, high capacity optical storage medium.

The timing and tracking information recorded at the beginning of each servo block consists of three pixels as shown in Fig. 3.10. The first two pixels are tracking pixels and are separated by 4-5 pixels along track and offset by half a track from the track center in opposite directions. If the head is offset outside a track, the readout signal from the first tracking pixel will be larger than that from the second as shown in Fig. 3.10a. If the head is offset inside a track, the readout signal from the second tracking pixel will be larger than that from the first as shown in Fig. 3.10b. If the head is aligned with a track, the readout signal from the tracking pixels will be as shown in Fig. 3.10c.

The timing pixel is offset from the tracking pixels by a fixed angular distance of 19 pixel locations. By measuring the elapsed time between the readout signals from the tracking pixels and the timing pixel, one can determine the angular velocity of the disk. This time is used to generate a clock signal in the disk system locked to the rotation of the disk. Pixels recorded on the disk are recorded at fixed time intervals and thus fixed angles beyond the timing and tracking information. Because the timing and tracking information in each track is embossed in a radially aligned fashion on the disk, the information recorded on the disk will also be aligned radially, thus providing the desired across-track coherence.

### 3.2.4 Modulation Transfer Function

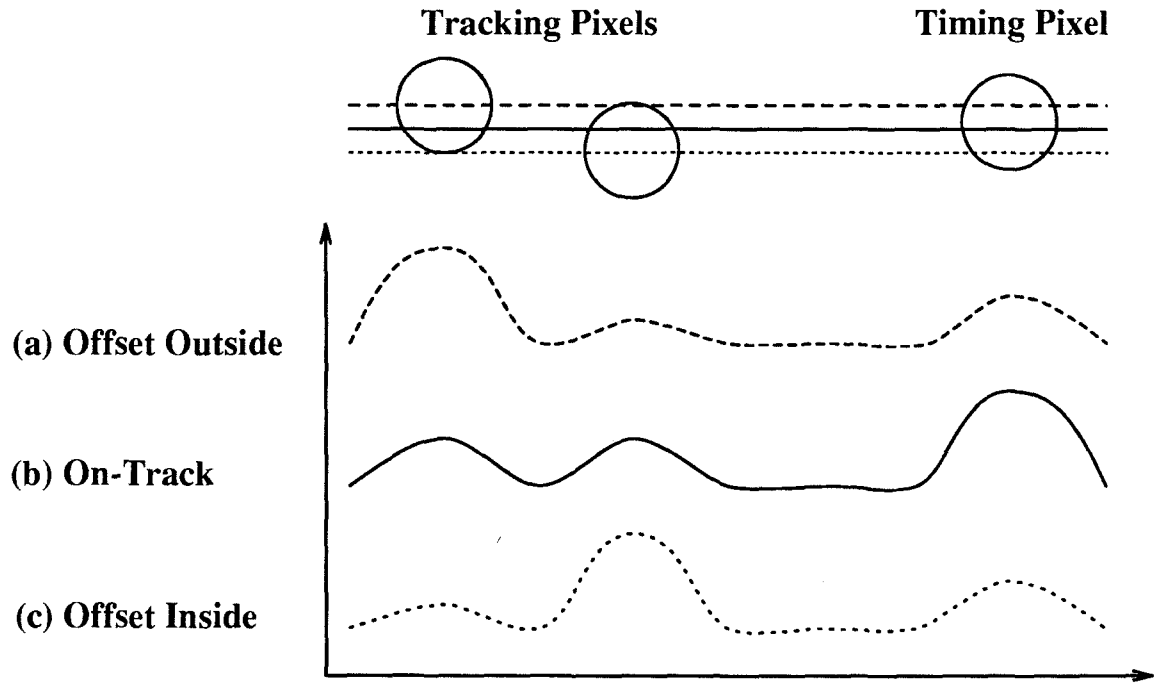



Fig. 3.10 - Timing and Tracking Information

The modulation transfer function is one important way to characterize an optical system. Modulation is defined as follows,

$$M = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \tag{3.4}$$

and always lies between 0 and 1. For a sinusoidal intensity pattern, the modulation is defined as follows:

$$I = I_0 + I_1 \cos[2\pi(u_1 x + v_1 y)] \tag{3.5}$$

$$M = I_1 / I_0 \tag{3.6}$$

The modulation transfer function, which measures the frequency response of an optical system, is defined as the output modulation over the input modulation as a function of frequency:

$$MTF(f) = \frac{M_o(f)}{M_i(f)} \tag{3.7}$$

The optical disk, like film, is a nonlinear medium. As Goodman points out about film,[18] it does not make sense to talk about a transfer function of a nonlinear

system, but it is often possible to separate the nonlinear system into a cascade of a linear system followed by a memoryless nonlinearity; one can then talk about the MTF of the linear part of the system. The same holds true for optical disks.

In Subsubsec. 3.3.1.4, we model the pixel pattern recorded on the disk given a desired binary image. The recorded reflectivity pattern is a linear function of the binary image. We can extend this model to an analog image by applying a hard-limiting nonlinearity at the output of the linear system as shown in Fig. 3.11. The frequency domain representation of Eq. 3.5 is as follows:

$$I = I_0\delta(u,v) + I_1\frac{\delta(u-u_1,v-v_1)+\delta(u+u_1,v+v_1)}{2} \tag{3.8}$$

If we record this pattern on the disk, we find the following response in the frequency domain before hard limiting:

$$\tilde{A}(u,v) = S(u,v)\sum_{nm}\left[I_0\delta\left(u-\frac{n}{\Delta_\theta},v-\frac{m}{\Delta_r}\right)+\frac{I_1}{2}\delta\left(u-u_1-\frac{n}{\Delta\theta},v-v_1-\frac{m}{\Delta_r}\right)\right]$$
$$+\frac{I_1}{2}\delta\left(u+u_1-\frac{n}{\Delta_\theta},v+v_1-\frac{m}{\Delta_r}\right) \tag{3.9}$$

$$S(u,v) = \Delta_\theta \mathrm{sinc}(\Delta_\theta u)\otimes\frac{d}{2}\frac{\mathrm{J}_1(\pi\rho d)}{\rho} \tag{3.10}$$

If we restrict our attention to the zero-order, we find the following expression for the recorded pattern and MTF:

$$I = S(0,0)I_0 + S(u_1,v_1)I_1\cos[2\pi(u_1x+v_1y)] \tag{3.11}$$

$$MTF(f) = \frac{S(u_1,v_1)}{S(0,0)} \tag{3.12}$$

The amplitude plot of Figure 3.12 shows the MTF for information recorded at the inner and outer radii of the disk.

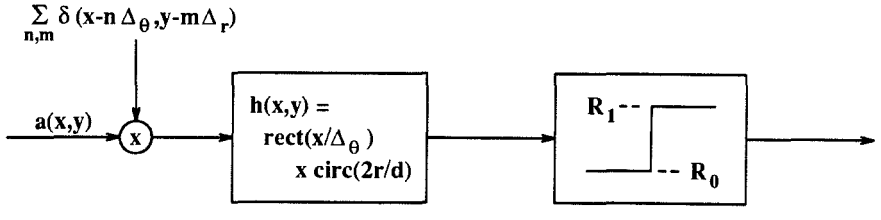
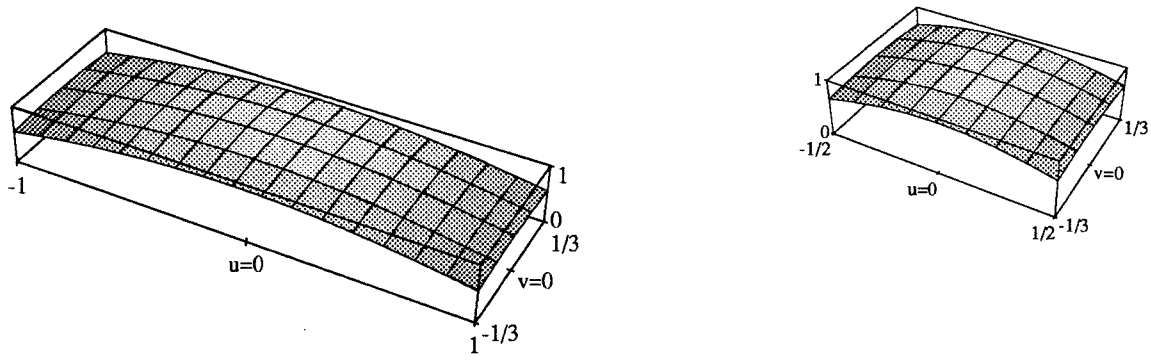



Fig. 3.11 - Optical Disk as Linear System Followed By Hard Limiter




(a) $u, v(\mu m^{-1})$ (b)

Fig. 3.12 - MTF of Optical Disk for Recording at (a) $R$=3$cm$ and (b) $R$=6$cm$

### 3.2.5 Optical Flatness of Disks

Many of the parallel readout techniques that will be described are diffractive or holographic in nature. Thus the optical flatness of the disks are an important consideration. Any variations in the optical flatness can be viewed as phase variations in the recorded data that if too large can effectively destroy parallel readout.

Possible sources of phase variations include actual imperfect flatness of the recording medium, and nonuniformities in the thickness and index of the protective layer. The recording medium itself is essentially flat optically to within less than a wavelength. Thus thickness and index variations in the protective layer are the main source of phase variation across the disk.

A Fizeau interferometer[19] was used to measure phase variations across large areas of the disk. In the interferometer (Fig. 3.13), a collimated laser beam strikes the disk. Part of the beam is reflected at the air-PC or air-glass interface at the surface of the protective layer. The rest of the beam is reflected by the storage medium. The two reflected beams interfere creating a fringe pattern where adjacent fringe lines correspond to optical pathlength differences of a single wavelength in the optical thickness of the disk upon reflection. This interference pattern may be

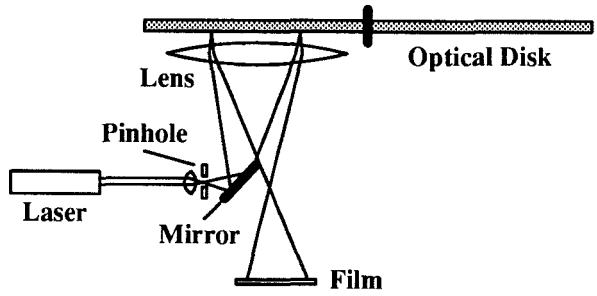



Fig. 3.13 - Fizeau Interferometer

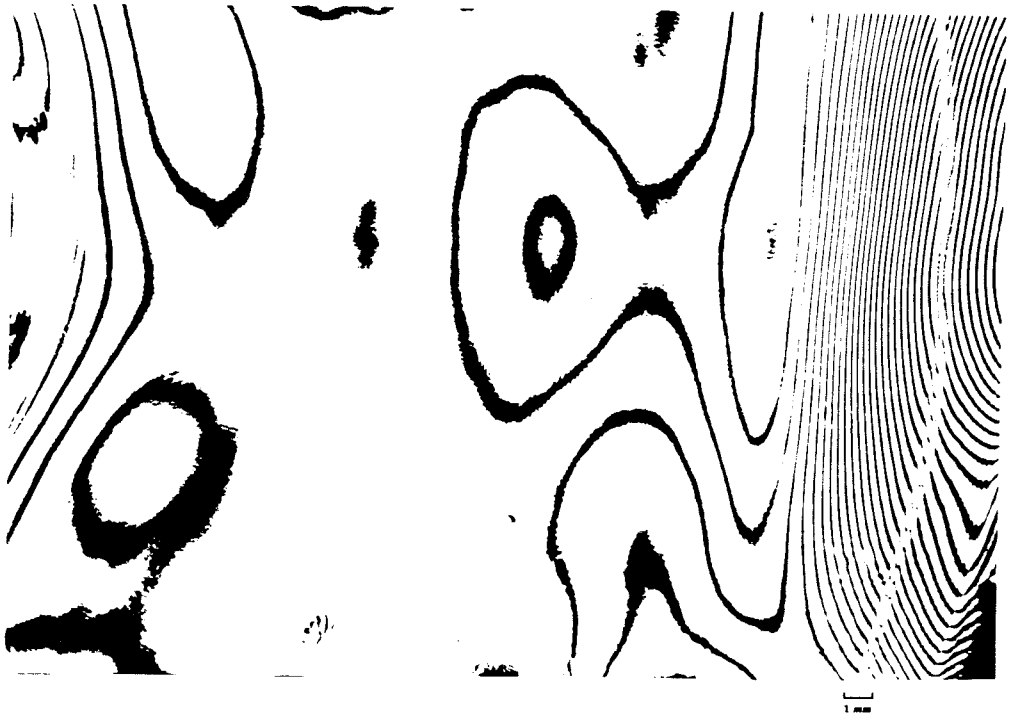
viewed by the naked eye or imaged onto film or a CCD camera.

Figure 3.14a and b show typical fringe patterns from 24 $\times$ 36 $mm$ regions of the glass- and PC-covered disks respectively. The marked curve near the left side of each pattern represents the inner recording radius and the marked curve near the right side represents the outer recording radius. The glass-covered disks generally show a greater degree of optical flatness than the PC-covered disks. In general, polycarbonate will have greater index variations than glass; in fact, polycarbonate can also be highly birefringent. This birefringence can be problematic during polarization dependent readout of magnetooptic disks.

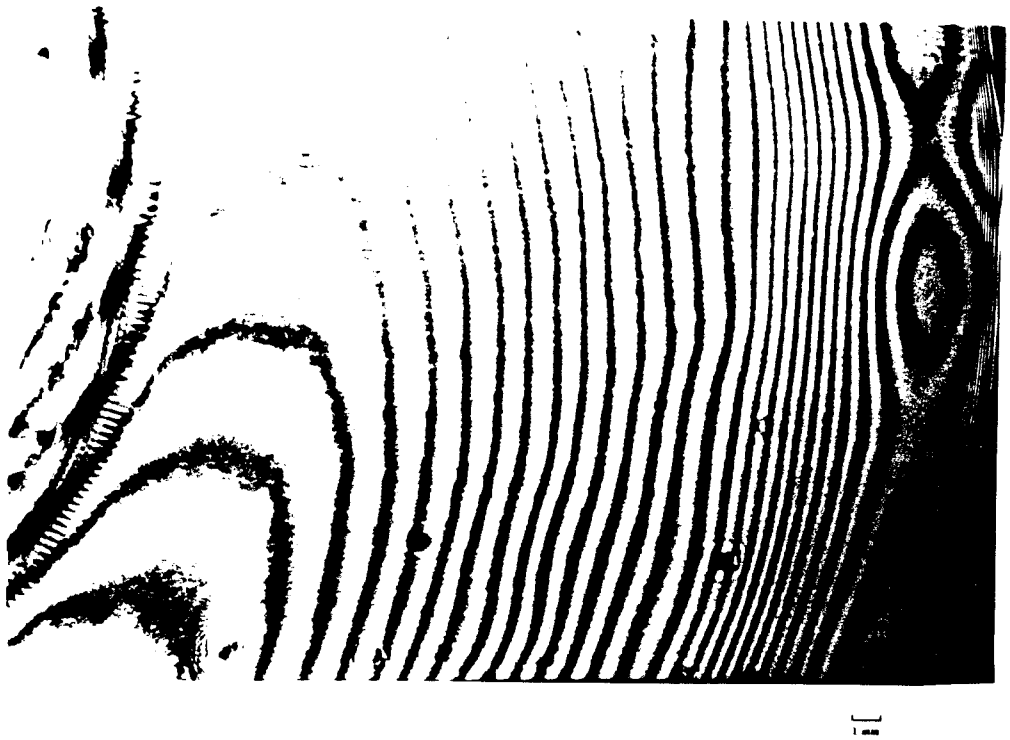
Note that both glass- and PC-covered disks exhibit greater phase variations at outer radii. This is most likely due to index or thickness variations introduced by the molding process used to create the protective base onto which the metal-alloy layers are deposited.

Although phase variations greater than a wavelength are evident in all disks, regions where the phase varies by significantly less than a wavelength over areas corresponding to the size of blocks of data to be read out in parallel are sufficient for parallel readout. In our experiments, this corresponds roughly to 2-D arrays, thousands of pixels or millimeters on a side. Note that candidate recording regions are numerous on the glass-covered disks, especially at the inner recording radii.

### 3.2.6 Disk Diffraction Patterns



(a) Glass



(b) Polycarbonate

Fig. 3.14 - Optical Flatness of Disks

In order to verify that the disk has sufficient optical flatness for diffractive or holographic readout techniques, a simple diffraction grating is recorded on the disk and illuminated by a collimated laser beam. The resulting diffraction pattern is then observed.

The test diffraction grating consists of alternating sets of $L$ tracks with all pixels on and $L$ tracks with all pixels off. In Subsubsec. 3.3.1.4, we estimate the optical efficiency of the disk for reading out an arbitrary image. Using Eq. 3.30, we can model the amplitude reflectivity of the diffraction grating recorded on the disk as follows:

$$i(x,y) = r_0 + (r_1 - r_0)\left\{\left[\text{rect}\left(\frac{y}{l\Delta_r}\right) \otimes \sum_l \delta(y - l2L\Delta_r)\right] \times \sum_{nm} \delta\left(x - n\Delta_\theta, y - m\Delta_r - \frac{(L-1)\Delta_r}{2}\right)\right\} \otimes s(x,y) \tag{3.13}$$

$$s(x,y) = \text{rect}\left(\frac{x}{\Delta_\theta}\right) \text{circ}\left(\frac{2r}{d}\right) \tag{3.14}$$

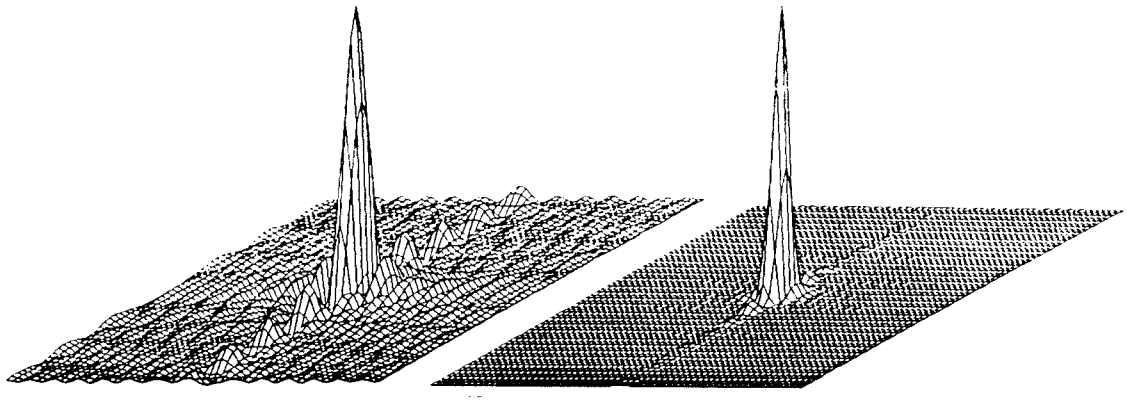
where $s(x,y)$ represents the shape of a recorded pixel (Fig. 3.27). The shift in the pixel sampling $y - (L-1)\Delta_r/2$ is to align the sampling array with the grating.

The farfield or Fraunhofer diffraction pattern is then given using Eq. 3.31 as follows:

$$P(u,v) = r_0\delta(u,v) + (r_1 - r_0)\left\{\left[\frac{1}{2}\sum_l \text{sinc}\left(\frac{l}{2}\right)\delta\left(u, v - \frac{l}{2L\Delta_r}\right)\right] \otimes \frac{1}{\Delta_\theta\Delta_r}\sum_{nm} e^{-j\pi m(L-1)}\delta\left(u - \frac{n}{\Delta_\theta}, v - \frac{m}{\Delta_r}\right)\right\} S(u,v) \tag{3.15}$$

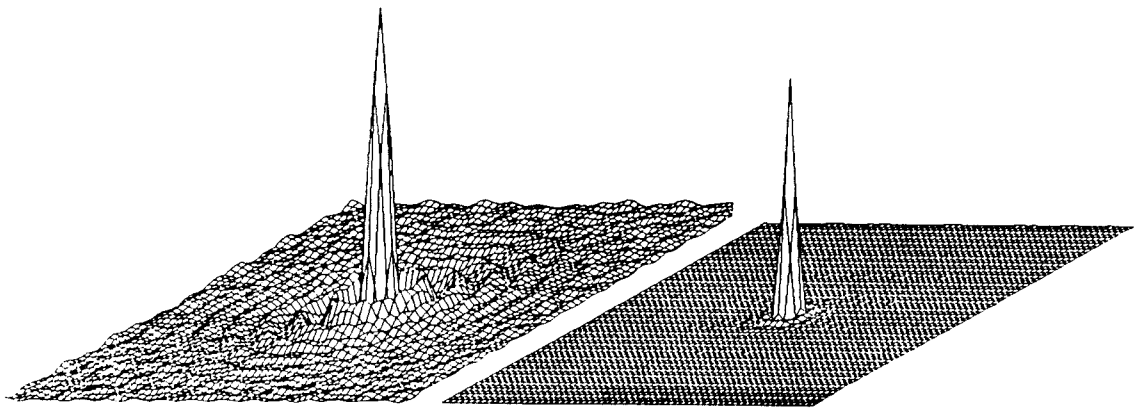
$$S(u,v) = \frac{d}{2}\frac{\text{J}_1(\pi d\rho)}{\rho} \otimes \Delta_\theta \text{sinc}(\Delta_\theta u)\delta(v) \tag{3.16}$$

where $S(u,v)$, the 2-D Fourier transform of the shape of the recorded pixel, acts as an envelope over the entire diffraction pattern. Figure 3.15a and b show the amplitude and magnitude squared of $S(u,v)$ for patterns recorded at the innermost and outermost radii on the disk respectively.



Amplitude Intensity

(a) $R = 3\ cm$



Amplitude Intensity

(b) $R = 6\ cm$

Fig. 3.15 - Amplitude and Magnitude Squared of Diffraction Envelope

We can simplify the Eq. 3.13 by performing the innermost convolution of two delta functions as follows:

$$\alpha_k = \sum_l \text{sinc}\left(\frac{l}{2}\right) \delta\left(u, v - \frac{l}{2L\Delta_r}\right)$$

$$\circledcirc \sum_{n,m} (-1)^{m(L-1)} \delta\left(u - \frac{n}{\Delta_\theta}, v - \frac{m}{\Delta_r}\right) \tag{3.17}$$

$$= \sum_{nk} \delta\left(u - \frac{n}{\Delta_\theta}, v - \frac{k}{2L\Delta_r}\right) \sum_{l+2Lm=k} (-1)^{m(L-1)} \text{sinc}\left(\frac{l}{2}\right) \tag{3.18}$$

$$= \sum_{nk} \delta\left(u - \frac{n}{\Delta_\theta}, v - \frac{k}{2L\Delta_r}\right) \sum_m (-1)^{m(L-1)} \text{sinc}\left(\frac{k - 2Lm}{2}\right) \tag{3.19}$$

$$= \sum_{nk} \delta\left(u - \frac{n}{\Delta_\theta}, v - \frac{k}{2L\Delta_r}\right) \sum_m (-1)^{m(L-1)} \frac{\sin\left(\pi \frac{k-2Lm}{2}\right)}{\pi \frac{k-2Lm}{2}} \tag{3.20}$$

$$= \sum_{nk} \delta\left(u - \frac{n}{\Delta_\theta}, v - \frac{k}{2L\Delta_r}\right) \sum_m (-1)^{m(L-1)} \cos(mL\pi) \frac{\sin\left(k\frac{\pi}{2}\right)}{\pi \frac{k-2Lm}{2}} \tag{3.21}$$

$$= \begin{cases} (-1)^{\frac{k-1}{2}} \frac{2}{\pi} \sum_m (-1)^m \frac{1}{k-2Lm} & k, \text{ odd} \\ (-1)^m & k = 2Lm \\ 0 & \text{otherwise} \end{cases} \tag{3.22}$$

Substituting $\alpha_k$ into Eq. 3.15, we find the following expression for the far-field diffraction pattern:

$$P(u,v) = r_0 \delta(u,v) + \frac{r_1 - r_0}{2} \frac{1}{\Delta_\theta \Delta_r} \times \sum_{nk} \alpha_k \delta\left(u - \frac{n}{\Delta_\theta}, v - \frac{k}{2L\Delta_r}\right) S\left(\frac{n}{\Delta_\theta}, \frac{k}{2L\Delta_r}\right) \tag{3.23}$$

Thus, the expected diffraction pattern is a series of impulses weighted by both $\alpha_k$ and $S(u,v)$.

Refraction at the air-glass or air-PC interface at the surface of the protective layer causes barrel distortion of the diffraction pattern. Taking this factor into account, we find the following pair of equations for the expected diffraction pattern:

$$P(u,v) = r_0 \delta(u,v) + \frac{r_1 - r_0}{2} \frac{1}{\Delta_\theta \Delta_r} \sum_{nk} \alpha_k \delta\left(u - \frac{n}{\beta_{nk}\Delta_\theta}, v - \frac{k}{\beta_{nk} 2L\Delta_r}\right) S(u,v) \tag{3.24}$$

$$\beta_{nk}^2 = 1 - \left(1 - \frac{1}{n_p^2}\right)\left(\left(\frac{n\lambda}{\Delta_\theta}\right)^2 + \left(\frac{k\lambda}{2L\Delta_r}\right)^2\right) \tag{3.25}$$

where $n_p$ is the refractive index of the protective layer.

We record the alternating track diffraction pattern with $L = 2$ and $L = 10$ at the inner $R = 3$ $cm$ and outermost $R = 6$ $cm$ recording radii. At the innermost recording radii, we find that the overlap of the pixels is so great that we can approximate the recorded pattern by sampling in the across-track direction only. In this case, the expected diffraction pattern simplifies to the following equation:

$$P(u,v) = r_0 \delta(u,v) + \frac{r_1 - r_0}{2\Delta_r}$$

$$\times \begin{cases} \sum_k \alpha_k \delta\left(u, v - \frac{k}{\beta_{0k} 2L\Delta_r}\right) \delta(u) d\,\mathrm{sinc}(vd) & \text{inner} \\ \frac{1}{\Delta_\theta} \sum_{nk} \alpha_k \delta\left(u - \frac{n}{\beta_{nk}\Delta_\theta}, v - \frac{k}{\beta_{nk} 2L\Delta_r}\right) \frac{d}{2} \frac{\mathrm{J}_1(\pi d\rho)}{\rho} & \text{outer} \end{cases} \tag{3.26}$$

Figure 3.16 shows the desired pixel patterns for $L = 2$ gratings recorded at both the inner and outer radii. The actual recorded pattern is essentially the same as the desired pattern except that we cannot record over the embossed timing and tracking information at the start of each servo-area.

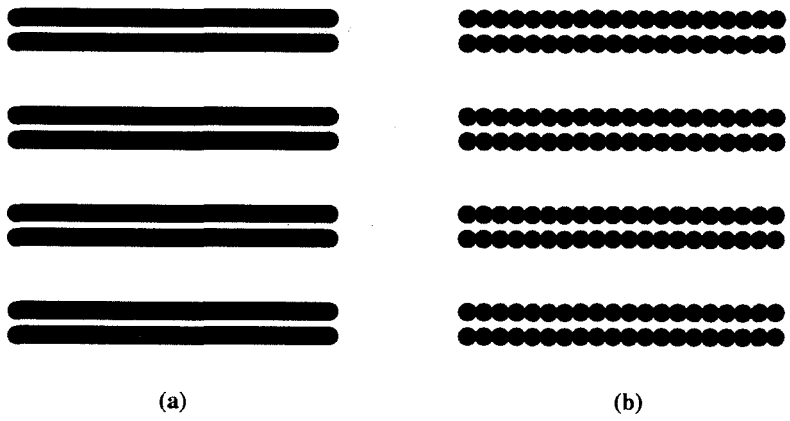



Fig. 3.16 - $L = 2$ Diffraction Gratings Recorded on the Disk at (a) Inner and (b) Outer Radii

Figures 3.17a and b show the expected and actual diffraction patterns for $R$=3,6$cm$ and $L$=2 and 10 respectively. The actual diffraction patterns match the expected ones pretty well. Notable exceptions are the lines passing through points corresponding to a continuum of along-track frequencies at the across-track sampling frequency and its harmonics, and (in the case of the diffraction pattern from the outer radii) lines corresponding to a continuum of across-track frequencies at the along-track sampling frequency and its harmonics. These lines correspond to "noise" in the diffraction pattern nonperiodic along-track and periodic across-track at the sampling frequency, and nonperiodic across-track and periodic along-track at the sampling frequency. Potential sources of such noise include track curvature

and the embossed timing and tracking information as well as the round shape of the pixels.

Another notable difference between the expected and observed diffraction patterns is the elongated shape of the reconstructed points. This is most likely due to track curvature. As later explained, one effect of position error due to track curvature is an effective phase error that differs for each diffracted order. For diffraction in the across-track direction, this phase error corresponds to a cylindrical lens at the surface of the disk.

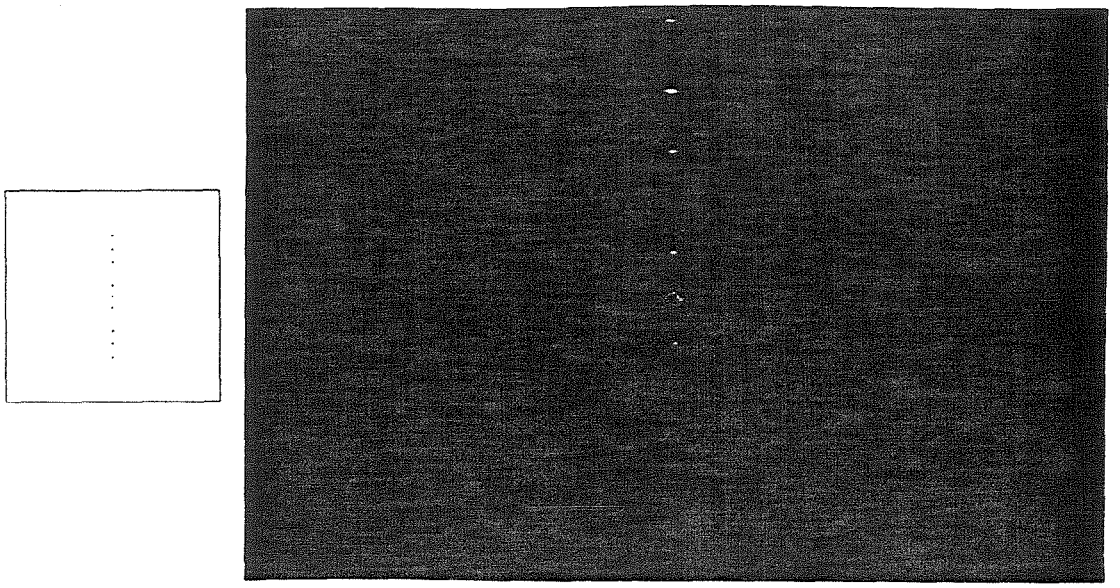
We can estimate the diffraction efficiency by taking the magnitude squared of the coefficient at each diffraction order. For the case of recording at the inner radii, we can estimate the diffraction efficiency $\eta_k$ of the $k^{th}$ order as follows:

$$\eta_k = |r_0|^2 \delta_k + \left| \frac{d}{\Delta_r} \frac{r_1 - r_0}{2} \mathrm{sinc}\left( \frac{dk}{2L\Delta_r} \right) \alpha_k \right|^2 \tag{3.27}$$

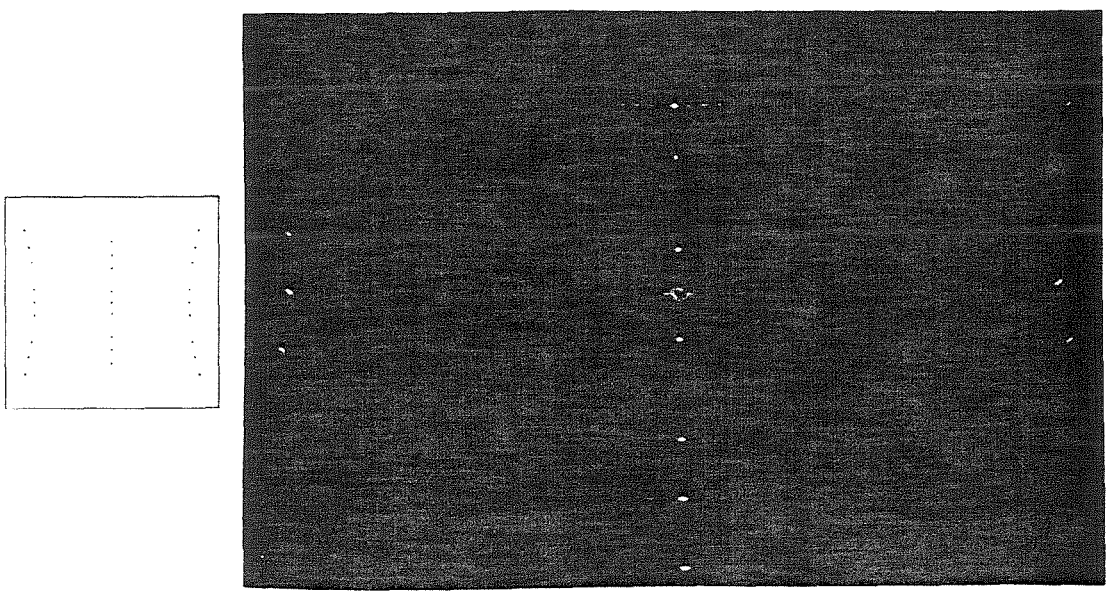
In Tables 3.2 and 3.3, we compare the expected diffraction efficiencies with the measured ones. Zeroes in the measured efficiency column correspond to no observable light above background level. We find that although the predicted and measured diffraction patterns match, the diffraction efficiencies do not match very well. This discrepancy is mostly attributable to the difficulty in isolating the light diffracted into each of the closely spaced orders, especially for those orders with lower diffraction efficiency.

### 3.3 Parallel Readout Techniques

In this chapter, we consider and experiment with many techniques for parallel readout. These techniques can be divided into two major classes: imaging and holographic as shown in Fig. 3.18. In imaging parallel readout, data is recorded on the disk to create a reflectivity pattern that matches as closely as possible, the light intensity pattern that is desired on the detector during readout. In holographic parallel readout, the data is encoded in such a way that when written on the disk

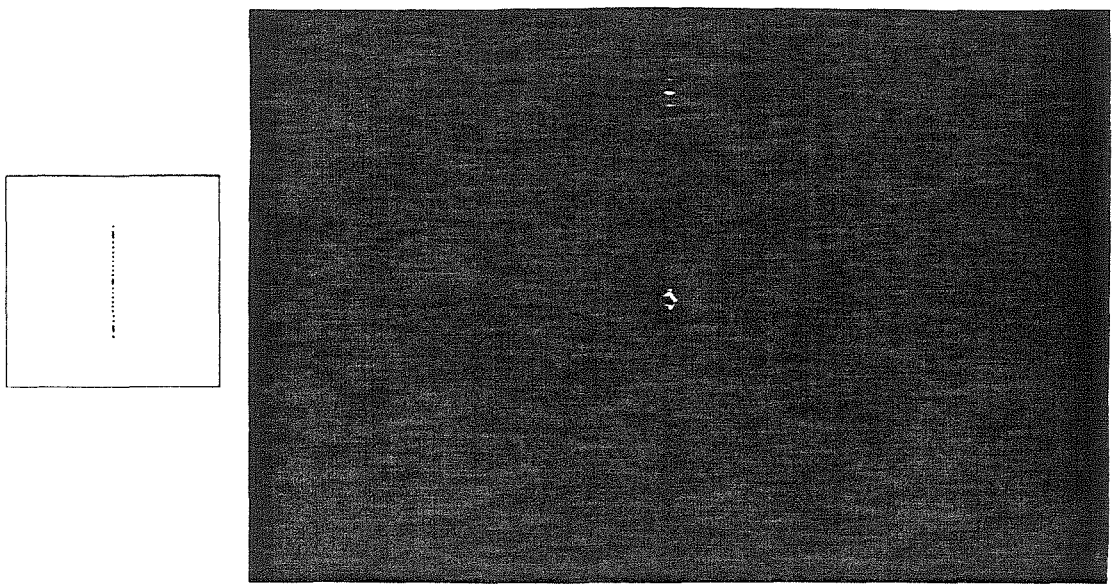


$R = 3\ cm$

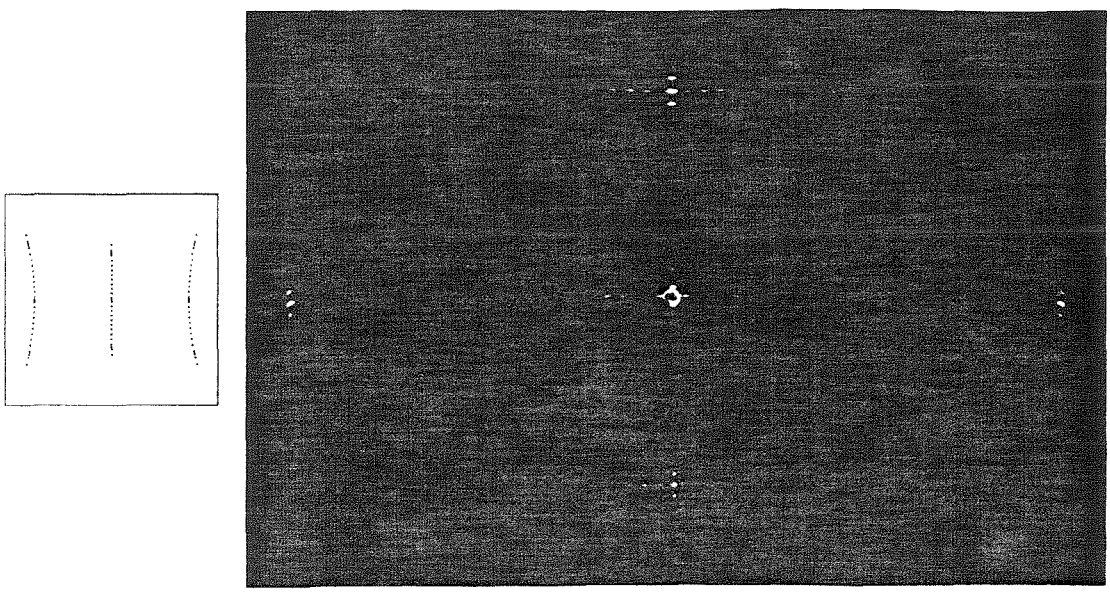


$R = 6\ cm$

Fig. 3.17a - Expected and Observed Diffraction Patterns for $L$=2



$R = 3\ cm$



$R = 6\ cm$

Fig. 3.17b - Expected and Observed Diffraction Patterns for $L$=10

Table 3.2 - Predicted and Measured Disk Diffraction Efficiencies for $L$=2 Alternating Track Grating

| Order ($k$) | $\eta_k$ | $\eta_{k,meas}$ | $\eta_k/\eta_{k,meas}$ |
|---|---|---|---|
| 0 | 5.0e-2 | | |
| 1 | 3.1e-4 | 3.5e-5 | .11 |
| 2 | 0 | 0 | |
| 3 | 1.4e-4 | 3.2e-5 | .23 |
| 4 | 1.2e-4 | 1.0e-4 | .83 |
| 5 | 1.3e-5 | 44.1e-4 | 32 |
| 6 | 0 | 0 | |
| 7 | 6.4e-6 | 2.5e-4 | 390 |
| 8 | 2.9e-5 | 3.3e-4 | 11 |

Table 3.3 - Predicted and Measured Disk Diffraction Efficiencies for $L$=10 Alternating Track Grating

| Order | $\eta_k$ | $\eta_{k,meas}$ | ratio | Order | $\eta_k$ | $\eta_{k,meas}$ | ratio |
|---|---|---|---|---|---|---|---|
| 0 | 5.0e-2 | | | 21 | 3.8e-5 | 1.3e-4 | 3.4 |
| 1 | 3.1e-4 | 0 | | 22 | 0 | 0 | |
| 2 | 0 | | | 23 | 2.6e-6 | 1.5e-5 | 5.8 |
| 3 | 1.4e-4 | 0 | | 24 | 0 | 0 | |
| 4 | 1.2e-4 | | | 25 | 5.0e-7 | 4.1e-6 | 8.2 |
| 5 | 1.3e-5 | 0 | | 26 | 0 | 0 | |
| 6 | 0 | | | 27 | 1.0e-7 | 1.6e-6 | 16 |
| 7 | 7.2e-6 | 3.2e-5 | .44 | 28 | 0 | 0 | |
| 8 | 0 | 0 | 1.3 | 29 | 8.3e-9 | 3.0e-6 | 360 |
| 9 | 5.2e-6 | 7.0e-6 | 1.3 | 30 | 0 | 0 | |
| 10 | 0 | 0 | | 31 | 7.3e-9 | 2.8e-6 | 380 |
| 11 | 4.4e-6 | 9.5e-6 | 2.2 | 32 | 0 | 0 | |
| 12 | 0 | 0 | | 33 | 6.9e-8 | 2.2e-6 | 32 |
| 13 | 4.5e-6 | 6.0e-6 | 1.3 | 34 | 0 | 0 | |
| 14 | 0 | 0 | | 35 | 2.6e-7 | 2.2e-6 | 8.5 |
| 15 | 5.6e-6 | 1.0e-5 | 1.8 | 36 | 0 | 0 | |
| 16 | 0 | 0 | | 37 | 9.9e-7 | 2.4e-6 | 2.4 |
| 17 | 1.0e-5 | 2.3e-5 | 2.3 | 38 | 0 | 0 | |
| 18 | 0 | 0 | | 39 | 1.1e-5 | 7.5e-6 | .68 |
| 19 | 5.9e-5 | 1.6e-4 | 2.7 | 40 | 2.9e-5 | 1.8e-5 | .62 |
| 20 | 1.2e-4 | 4.2e-4 | 3.5 | 41 | 1.3e-5 | 3.5e-6 | .27 |

and presented as input to the holographic reconstruction system, the desired recon-

struction is also obtained. In the following subsections, we consider each of these techniques in greater detail.

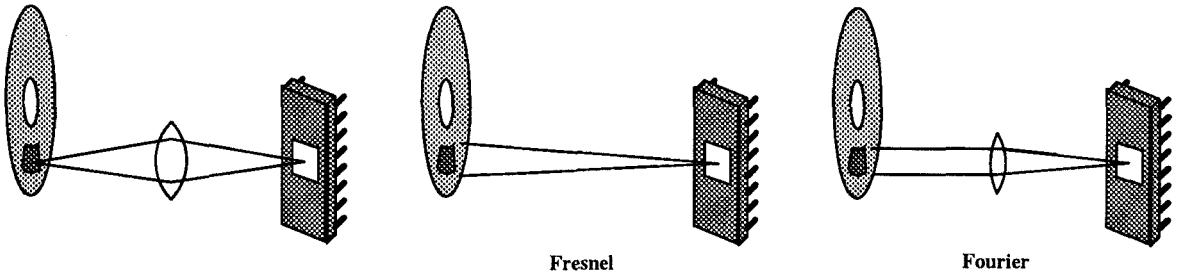



Fig. 3.18 - Parallel Readout Techniques

### 3.3.1 Imaging Parallel Readout

In imaging parallel readout, the data recorded on the disk is illuminated and the reflected light passed through a lens or other optical element before striking the detector. The lens takes light reflected from each point on the disk and focuses it onto a single point on the detector. Thus a pattern of light matching that reflected from the disk strikes the detector.

In encoding the data into a pixel array for imaging parallel readout, we are concerned with reproducing a reflectivity pattern that matches that of the desired data as closely as possible. With the Sony write-once disks, two problems become readily apparent. First, the low contrast between written and unwritten pixels seems to limit the maximum contrast obtainable in the detected light pattern. Second, it seems like only binary information can be retrieved from the disks since only binary pixels may be recorded. As we shall see, each of these problems may be circumvented through techniques described as follows:

#### 3.3.1.1 Recording of Binary Information

Recording of binary information is very simple in an imaging parallel readout system from the Sony write-once disk. A one- or two-dimensional binary array of

data may be recorded as a one- or two-dimensional binary pixel pattern, one pixel for each bit (binary piece of data). If the size of the pixels does not match that of the individual detector elements, a magnification system may be used to enlarge the pixel array. If the spacing of pixels does not match that of a 2-D detector array, either an anamorphic imaging system with different magnifications along the two axes may be used, or pixels formed in larger groups called *superpixels*, each superpixel having the desired pixel shape and spacing.

#### 3.3.1.2 Increasing Image Contrast

For the retrieval of binary information, the low contrast ratio between written and unwritten pixels may not be problematical given detector elements that can threshold between the two nonzero values. By modifying the imaging system, however, it is possible to obtain high contrast images from the disk, thus eliminating the need for special detectors.

Note that the image recorded on the disk will be a sampled version of the desired readout pattern. As will be readily apparent from Subsubsec. 3.3.1.4 on the efficiency of imaging readout, recording of a sampled image on the disk surface causes diffraction of the reflected light into multiple orders. Assuming that the desired readout pattern is band-limited, each of these orders will contain complete spectral information about the desired readout light pattern. However, the background reflectivity that causes low contrast is not itself sampled in the recorded image. The background light is thus completely reflected in the zero order and not diffracted at all. By imaging any or all diffracted orders except for the zero order, a high contrast output may be achieved. Schlieren imaging is a special case of this technique where the zero order and all negative (or positive) diffracted orders are blocked in an imaging system as shown in Fig. 3.19.

In many optical systems, it may be most convenient to simply image a single

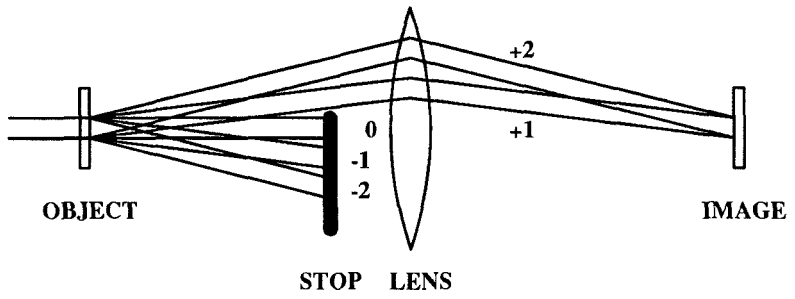



Fig. 3.19 - Schlieren Imaging

track-diffracted order as shown in Fig. 3.20. Figure 3.21 compares the low contrast image obtained using a conventional microscope image with a high contrast image obtained by imaging the first track-diffracted order.

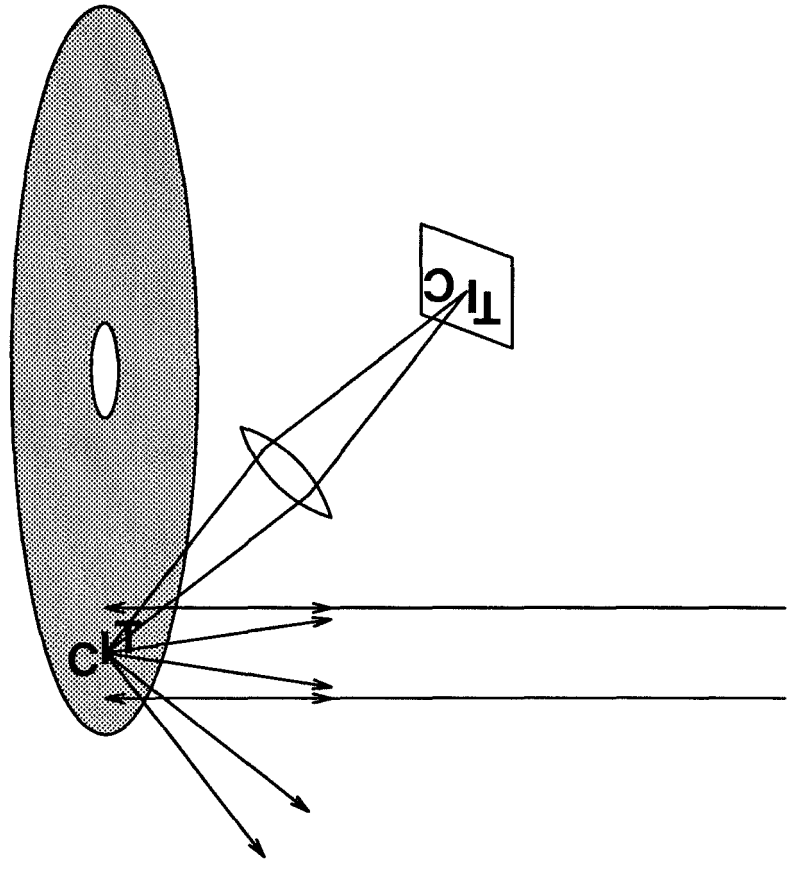
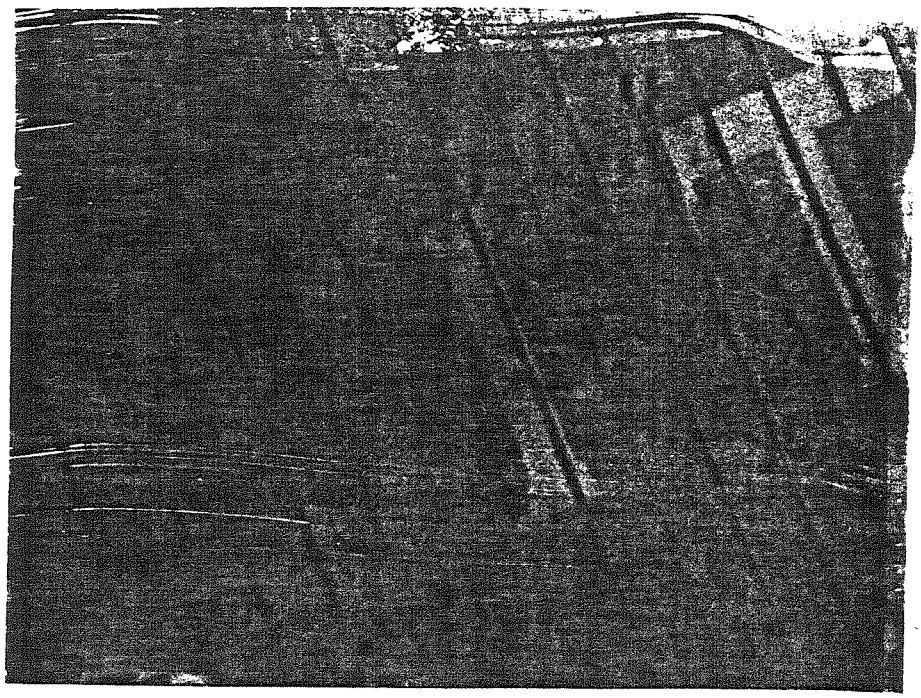
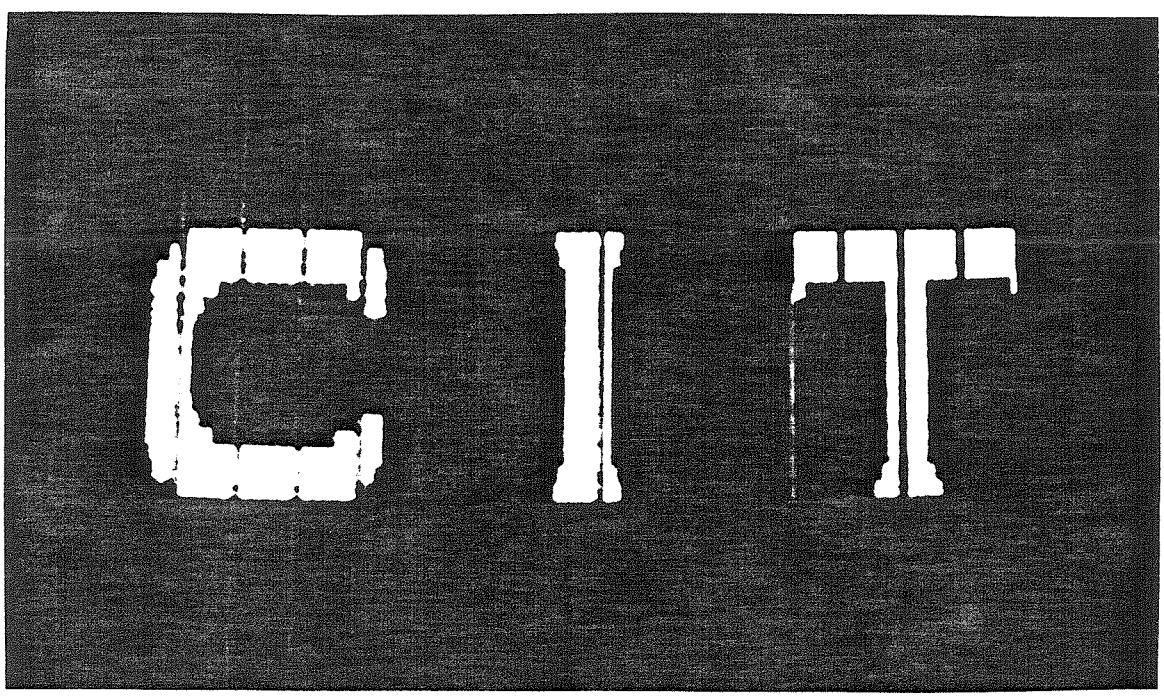


Fig. 3.20 - Imaging the First Diffracted Order

Finally, the offending zero order may be blocked using a stop as in the central dark ground method, or an aperture with oblique illumination as in the darkfield method.



(a) Microscope Image



(b) First Order Diffracted Image

Fig. 3.21 - Image Contrast Enhancement

by Imaging a Single Diffracted Order

### 3.3.1.3 Recording of Grayscale Information for Imaging Readout

We have thus far only considered the imaging parallel readout of binary information from optical disks. Although the disk is a binary recording medium, it is possible to retrieve grayscale data (data with more than two possible values) from the disk. This is achieved by using an array of pixels called a superpixel to encode each piece of grayscale information. By averaging the binary values of each pixel within the superpixel, a number of discrete levels equal to the number of pixels in each superpixel plus one can be encoded. As shown in Fig. 3.22, for example, a superpixel consisting of an array of 4 × 4 pixels can be used to encode 17 different gray levels.

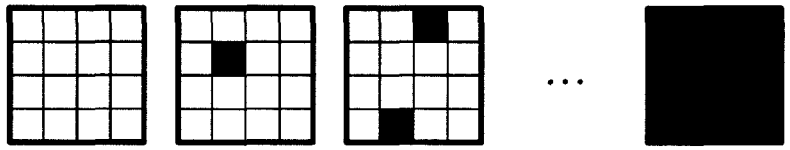



Fig. 3.22 - Superpixels for Grayscale Recording

Averaging over the superpixels may be achieved by defocusing the imaging system or otherwise blurring the image (Fig. 3.23a), or by using a detector that integrates over an area corresponding to the size of the superpixel (Fig. 3.23b). There are many well-established techniques[20] for encoding grayscale images with binary pixels..

We have used a simple technique where each piece of data is encoded using an $n \times m$ rectangular superpixel. The number of pixels within the superpixel is chosen stochastically as follows. If we wish to encode a normalized grey level $x \in [0, 1]$, we write $N$ of the $nm$ pixels where the probability distribution of $N$ is as follows:

$$N = \begin{cases} \lfloor xnm \rfloor & P = \lceil xnm \rceil - xnm \\ \lceil xnm \rceil & P = xnm - \lfloor xnm \rfloor \end{cases} \tag{3.28}$$

The expectation value of $N$ is then $< N >= xnm$. In areas of low spatial variation, by choosing the number of pixels within each superpixel stochastically, we

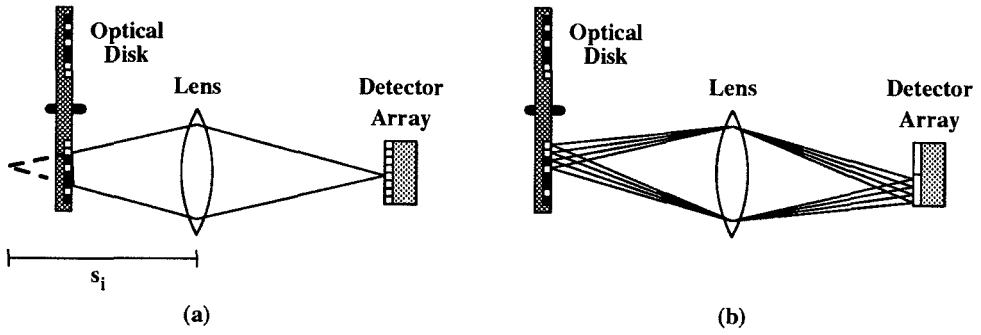



Fig. 3.23 - Pixel Averaging Over Superpixels

(a) Blurring or (b) Integrating Detector

automatically render gray scales with more discrete levels by averaging over groups of superpixels. Figure 3.24 compares stochastic versus fixed choice of pixel number in rendering an image.

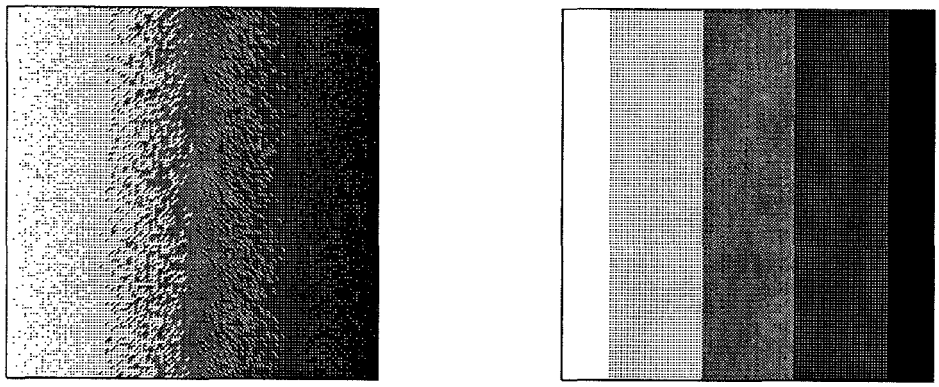



Fig. 3.24 - Stochastic vs. Fixed Choice of Pixel Number

By stochastically choosing the positions of pixels written within each superpixel, we hope to eliminate possible artifacts that are caused by repeatedly using the same pattern within the image. Figure 3.25 compares stochastic versus fixed choice of pixel position in rendering a constant grayscale level. Figure 3.26 shows a picture recorded on the optical disk using 6 $\times$ 9 superpixels for 55 graylevels with stochastic choice of pixel number and position.

### 3.3.1.4 Optical Efficiency of Imaging Parallel Readout

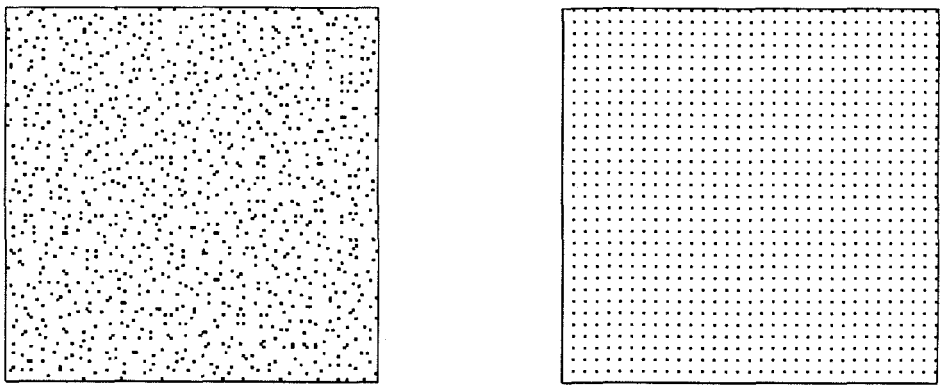



Fig. 3.25 - Stochastic vs. Fixed Choice of Pixel Position

In this section, we calculate the expected optical efficiency when reading out an image in parallel from the optical disk following the derivation in Ref. [21]. We begin by modeling the amplitude of the reflectivity of the disk surface according to the following equation:

$$\begin{aligned} i(x,y) = r_0 + (r_1 - r_0)\left[b(x,y)\sum_{n,m}\delta(x - n\Delta_\theta, y - m\Delta_r)\right] \otimes \mathrm{circ}\left(\frac{2r}{d}\right) \\ + (r_2 - 2r_1 + r_0)\sum_{n,m}\left\{[b(x,y)\delta(x - n\Delta_\theta, y - m\Delta_r)] \otimes \mathrm{circ}\left(\frac{2r}{d}\right)\right\} \\ \times \left\{[b(x,y)\delta(x - (n+1)\Delta_\theta, y - m\Delta_r)] \otimes \mathrm{circ}\left(\frac{2r}{d}\right)\right\} \end{aligned} \tag{3.29}$$

where $r^2 = x^2 + y^2$, $b(x,y)$ is the desired binary image, $\Delta_\theta$ and $\Delta_r$ the along- and across-track pixel separation respectively, $d$ the diameter of written pixels, and $r_i$ the complex amplitude reflectively of the disk surface when written $i$ times. The along-track spacing $\Delta_\theta$ is a function of the radius $R$ at which the image is recorded. In this analysis, we ignore the effects of track-curvature.

Because the Sony write-once material essentially saturates after a single exposure to the write beam, $r_2 \approx r_1$. We simplify Eq. 3.29 by dropping the third term which accounts for the overlap of adjacent pixels and modifying the shape of the pixel in the second term to account for the overlap instead. We do this by assigning to each pixel, half the overlap region with its neighbors as shown in Fig. 3.27. This

Fig. 3.26 - Grayscale Image Recorded on Disk

changes the shape of the pixel from a *circ* to a *rect* $\times$ *circ*. The simplified model of disk reflectivity is as follows:

$$i(x,y) = r_0 + (r_1 - r_0)\left[b(x,y)\sum_{n,m}\delta(x - n\Delta_\theta, y - m\Delta_r)\right] \otimes \left[\text{circ}\left(\frac{2r}{d}\right)\text{rect}\left(\frac{x}{\Delta_\theta}\right)\right] \tag{3.30}$$

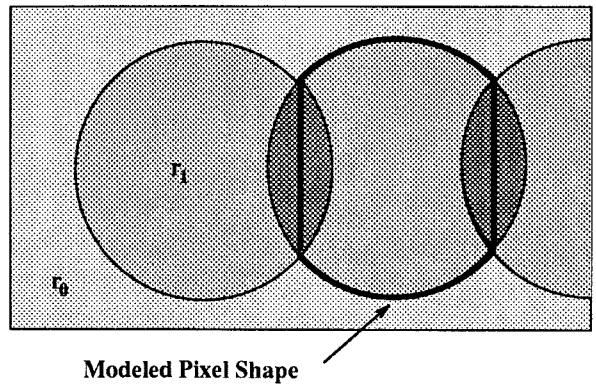



Fig. 3.27 - Modeled Pixel Shape

The above equation models the reflectivity incorrectly in areas where an unwritten pixel is adjacent to a written one in the along-track direction. Fortunately, assuming that the along-track spatial frequencies in the desired image are low compared to the sampling frequency $\Delta_\theta^{-1}$, there will be large clusters of written and unwritten pixels, and fewer written pixels adjacent to unwritten ones. Thus, the energy in the error term will be low.

Ignoring constant factors and phase factors, we find the following expected Fraunhofer diffraction pattern:

$$I(u,v) = r_0\delta(u,v) + (r_1 - r_0)\left[B(u,v) \otimes \frac{1}{\Delta_\theta \Delta_r}\sum_{n,m}\delta\left(u - \frac{n}{\Delta_\theta}, v - \frac{m}{\Delta_r}\right)\right] \times \left[d\frac{J_1(\pi d\rho)}{2\rho} \otimes \Delta_\theta \mathrm{sinc}(\delta_\theta u)\delta(v)\right]_{\rho^2=u^2+v^2} \tag{3.31}$$

Note that sampling in the image plane causes diffraction of the image spectrum into multiple orders. Any imaging system which blocks the zero order will produce a high contrast output because the pixels sample only the image and not the background–the term containing $r_0$–whose energy is contained entirely in the zero order.

The fraction $H_{nm}$ of incident light that goes into the $n,m$-th diffracted order is given by integrating the magnitude squared of the appropriate term from Eq. 3.31 as follows:

$$H_{nm} = \left|\frac{d(r_1 - r + 0)}{2\Delta_\theta \Delta_r}\right|^2$$

$$\times \int\int \left| B(u - \frac{n}{\Delta_\theta}, v - \frac{m}{\Delta_r}) \left[ \frac{J_1(\pi d\rho)}{\rho} \otimes \Delta_\theta \mathrm{sinc}(\Delta_\theta u)\delta(v) \right] \right|^2 du dv \quad (3.32)$$

The shape of the written pixels determine the characteristic $[J_1 \otimes \delta\mathrm{sinc}]$ envelope which modulates the entire diffraction pattern. Assuming that the image is sufficiently oversampled by the pixels, the envelope will be nearly constant over the image spectrum allowing us to simplify Eq. 3.32 as follows:

$$H_{nm} \approx \left| \frac{(r_1 - r + 0)}{\Delta_\theta \Delta_r} \right|^2 \int\int |B(u - \frac{n}{\Delta_\theta}, v - \frac{m}{\Delta_r})|^2 du dv \times \left[ \frac{d}{2} \frac{J_1(\pi d\rho)}{\rho} \otimes \Delta_\theta \mathrm{sinc}(\Delta_\theta u)\delta(v) \right]_{u=n/\Delta_\theta, v=m/\Delta_r} \quad (3.33)$$

The remaining integral corresponds to the energy in the spectrum of the image which, by Parseval's Theorem, is equivalent to the energy in the image itself. Thus, the fraction of incident light energy that goes into the $n, m$-th diffracted order is given as follows:

$$H_{nm} \approx \left| \frac{r_1 - r_0}{\Delta_\theta \Delta_r} \right|^2 \int\int |b(x,y)|^2 dx dy \left| \frac{d}{2} \frac{J_1(\pi d\rho)}{\rho} \otimes \Delta_\theta \mathrm{sinc}(\Delta_x u)\delta(v) \right|^2 \quad (3.34)$$

Defining useful light as the total energy going into image spectra of all orders, we calculate the fraction of useful light contained in a single diffracted order as follows:

$$\frac{H_{nm}}{\sum_{n',m'} H_{n',m'}} = \frac{\left| \frac{d}{2} \frac{J_1(\pi d\rho)}{\rho} \otimes \Delta_\theta \mathrm{sinc}(\Delta_\theta u)\delta(v) \right|^2_{u=n/\Delta_\theta, v=m/\Delta_r}}{\sum_{n',m'} \left| \frac{d}{2} \frac{J_1(\pi d\rho')}{\rho'} \otimes \Delta_\theta \mathrm{sinc}(\Delta_\theta u')\delta(v') \right|^2_{u'=n'/\Delta_\theta, v'=m'/\Delta_r}} \quad (3.35)$$

The denominator is the sum of the squares of the Fourier series coefficients of an image with every pixel written with unity amplitude, and thus equivalent by Parseval's Theorem to the fraction of disk area written when all pixels are recorded. This fraction varies between 0.877 at the innermost radius to 0.785 at the outermost radius. We can thus simplify the equation for useful light in a given diffracted order

as follows:

$$\frac{H_{nm}}{\sum_{n',m'} H_{n'm'}} = \frac{\left|\frac{d}{2}\frac{J_1(\pi d\rho)}{\rho} \otimes \Delta_\theta \mathrm{sinc}(\Delta_\theta u)\delta(v)\right|^2}{\frac{1}{\Delta_\theta \Delta_r}\int_{-\Delta_\theta/2}^{\Delta_\theta/2}\int_{-\Delta_r/2}^{\Delta_r/2}\left|\mathrm{circ}\left(\frac{2r}{d}\right)\mathrm{rect}\left(\frac{x}{\Delta_\theta}\right)\right|^2 dxdy} \tag{3.36}$$

$$= \frac{\left|\frac{d}{2}\frac{J_1(\pi d\rho)}{\rho} \otimes \Delta_\theta \mathrm{sinc}(\Delta_\theta u)\delta(v)\right|^2}{\frac{1}{\Delta_\theta \Delta_r}\left[\frac{d^2}{2\Delta_\theta \Delta_r}\sin^{-1}\left(\frac{\Delta_\theta}{d}\right) + \frac{d}{2\Delta_r}\sqrt{1-\frac{\Delta_\theta}{d}}\right]} \tag{3.37}$$

The equations derived in this subsubsection can be used to estimate the diffraction efficiency of an imaging system that captures any or all of the diffracted orders. For example, for imaging the first track diffracted order, the fraction of incident energy in the first diffracted order is $\left|\frac{r_1 - r_0}{\Delta_\theta \Delta_r}\right|^2$ where we find the third term by sampling the envelope of Figs. x and y at the point corresponding to the diffraction order. For our system $\left|\frac{r_1 - r_0}{\Delta_\theta \Delta_r}\right|^2 = .114\%$. So .1% of the fraction of the incident light will be diffracted into the first track diffracted order.

### 3.3.2 Holographic Parallel Readout

In holographic parallel readout, the data recorded on the disk is illuminated and the reflected light strikes the detector after passing through an optical system. In general, in order to be considered a holographic system, the recorded data must contain both phase and amplitude information about the wave that reconstructs the desired output pattern. This information is often stored in a medium that in and of itself cannot record both phase and amplitude information. The first demonstration of imaging by reconstructed wavefronts, or optical holography, was by Gabor[22] in 1948. There are many good references on holography[23,24].

#### 3.3.2.1 Gabor Hologram

Given the complex wavefront $U(x_1, y_1)$ at a plane $z = 0$, the following equation can be used to calculate the diffracted wavefront $U(x_0, y_0)$ at a distance $z$ from the

original in the Fresnel diffraction regime:[18]

$$U(x_0,y_0) = \frac{e^{jkz}}{j\lambda z} e^{j\frac{k}{2z}(x_0^2+y_0^2)} \times \int\int U(x_1,y_1) e^{j\frac{k}{2z}(x_1^2+y_1^2)} e^{-j\frac{2\pi}{\lambda z}(x_0x_1+y_0y_1)} dx_1 dx_2 \qquad (3.38)$$

In Gabor's experiment, he recorded the wave transmitted by a semitransparent object as shown in Fig. 3.28. Assume that the light transmitted by the object can be modeled by Eq. 3.39 where $B(x,y)$ is the pattern of the complex wavefront diffracted by the object and $A$ is the complex amplitude of the planewave of coherent monochromatic radiation transmitted by the object. Equations 3.40-3.42 give the field amplitude at a distance $z_0$ from the object. The intensity pattern is given by Eq. 3.44. If we place a linear recording medium at this point with transmittivity proportional to the intensity pattern, we will record a Gabor hologram of the wavefront at $z_0$.

$$U(x,y,z=0) = A + B(x,y) \qquad (3.39)$$

$$U(x',y',z=z_0) = A' + B'(x',y') \qquad (3.40)$$

$$A' = Ae^{jkz_0} \qquad (3.41)$$

$$B' = \frac{e^{jkz_0}}{\lambda z_0} e^{j\frac{k}{2z_0}(x'^2+y'^2)} \int\int B(x,y) e^{j\frac{k}{2z_0}(x^2+y^2)} e^{-j\frac{2\pi}{\lambda z}(xx'+yy')} dxdy \qquad (3.42)$$

$$I(x',y',z_0) = ||A' + B'(x',y')||^2 \qquad (3.43)$$

$$= ||A'||^2 + A'^*B' + A'B'^* + ||B'||^2 \qquad (3.44)$$

If we illuminate the hologram with a planewave $A'$, the wave transmitted by the second term will be $||A'||^2 B'(x',y')$, a reconstruction of the recorded wavefront. If we propagate the light transmitted by the third term through a distance $z_0$, we get the following expression for the amplitude:

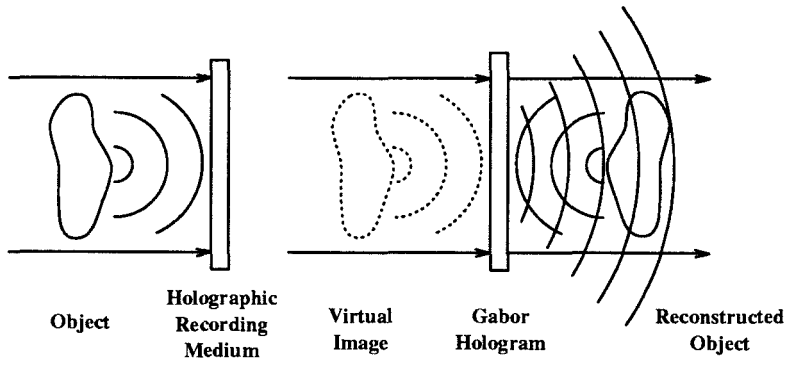



Fig. 3.28 - Recording and Reconstructing an On-Axis Hologram

$$U_3(x'', y'', z = 2z_0) = A'^2 \frac{1}{(\lambda z_0)^2} e^{j\frac{k}{2z_0}(x''^2+y''^2)} \int\int e^{-j\frac{k}{2z_0}(x'^2+y'^2)} \int\int B^*(x,y) e^{-j\frac{k}{2z_0}(x^2+y^2)} e^{j\frac{2\pi}{\lambda z}(xx'+yy')} dxdy \; e^{j\frac{k}{2z_0}(x'^2+y'^2)} e^{-j\frac{2\pi}{\lambda z}(x'x''+y'y'')} dx'dy' \tag{3.45}$$

$$= A'^2 B^*(x'', y'') \tag{3.46}$$

which except for conjugated phase is a reconstruction of the original object. A problem with the Gabor hologram is that all four terms of the transmitted light overlap along the optical axis as shown in Fig. 3.28. This means, for example, that if we would like to reconstruct an image on a detector, it would not only detect the desired image, but also a planewave, a diffracted version of the image, and a crossterm.

### 3.3.2.2 Leith-Upatnieks Hologram

The crosstalk problem of the Gabor hologram can be circumvented by modification of the architecture used to capture the hologram as shown in Fig. 3.29. In the Leith-Upatnieks hologram, a coherent reference wave strikes the recording medium at an angle relative to the optical axis defined by the medium and the object. We can thus refer to the Leith-Upatnieks hologram as an *off-axis* hologram in contrast

to the *on-axis* Gabor hologram. The equations describing off-axis holography are essentially the same as those used to describe on-axis holography, except the reference wave propagates in a direction with an angle $\theta$ from the optical axis. If we orient the $x' - y'$ coordinate system such that the optical axis and the direction of propagation of the reference wave are in the $x' - z$ plane, we can replace the $A'$ of the on-axis equations with $A'e^{jk_x x'}$ where $k_x = k\sin\theta$ for small $\theta$.

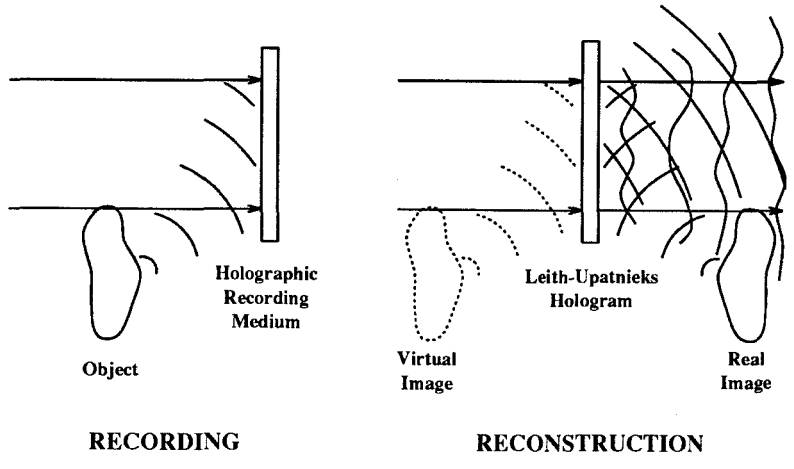



Fig. 3.29 - Recording and Reconstruction an Off-Axis Hologram

The intensity pattern recorded in the holographic medium will then be given by Eq. 3.47. If we illuminate this with an on-axis plane wave $C$, the second term again produces a reconstruction (Eq. 3.48) of the original wavefront at an angle $-\theta$ with the optical axis. The third term produces a reconstruction (Eq. 3.49) of the original object centered at an angle $\theta$ from the optical axis. If we choose $\theta$ large enough, we can separate the reconstruction of the original wave, that of the object, and the planewave and self-intereference terms, from one another.

$$U(x', y', z = z_0) = ||A'||^2 + A'^* B' e^{-jk_x x'} + A' B'^* e^{jk_x x'} + ||B'||^2 \quad (3.47)$$

$$U_2(x', y', z = z_0) = CA'^* B'(x', y') e^{-jk_x x'} \quad (3.48)$$

$$U_3(x'', y'', z = 2z_0) = CA'B(x'' - z_0 \sin\theta, y'') \quad (3.49)$$

If we consider the impulse response of the Leith-Upatnieks recording architecture, given an impulse $B$ (Eq. 3.50), the resulting hologram will be given by Eq. 3.51. This corresponds to a Fresnel zone plate, which acts like a lens with focal length $z_0$ centered at $(x_0 + z_0 \sin\theta, y_0)$. We thus see that the Gabor and Leith-Upatnieks holograms act to generate a superposition of Fresnel lenses, each corresponding to a point in the object and focusing the plane reference wave to a point in the real image. We thus call these recordings Fresnel holograms.

$$B(x,y) = \delta(x - x_0, y - y_0) \tag{3.50}$$

$$U(x', y', z = z_0) = ||A||^2 + \frac{2|A'|}{\lambda z_0} \cos\left[\frac{\pi}{\lambda z_0}(x' - (x_0 + z_0 \sin\theta))^2 + (y' - y_0)^2\right] + k\left(z_0 - x_0 \sin\theta - z_0 \frac{\sin^2\theta}{2}\right) - \frac{\pi}{2}\Bigg] + \frac{1}{(\lambda z_0)^2} \tag{3.51}$$

#### 3.3.2.3 Fourier Transform Hologram

A spherical lens can be used to generate the 2-D Fourier transform of an object if placed in the proper configuration. We can approximate a thin spherical lens with focal length $f$ as a transparency with the transmittance function in by Eq. 3.52. Note that a planewave incident on the lens will be converted into a converging spherical wave upon transmission through the lens. If we place the object one focal length in front of the lens and calculate the field a focal length after the lens, we will find that the field in the back focal plane is the exact 2-D Fourier transform of the field in the front focal plane. Specifically, given the field $B(x,y)$ in the front focal plane, Eq. 3.53 gives the field in the back focal plane.

$$t(x,y) = e^{-j\frac{k}{2f}(x^2+y^2)} \tag{3.52}$$

$$U(x', y', z = 2f) = -\frac{e^{j2kf}}{2\pi} \int\int B(x,y) e^{-j\frac{2\pi}{\lambda f}(xx' + yy')} dx dy \tag{3.53}$$

A Fourier transform hologram may be recorded, as shown in Fig. 3.30, by placing the object in the front focal plane of a spherical lens and placing the recording medium in the back focal plane, and interfering the resulting 2-D Fourier transform of the object with an on- or off-axis planewave. By illuminating the hologram with a planewave, we can reconstruct the original wavefront representing the Fourier transform of the desired object, its conjugate, as well as the magnitude squared of the transform and reference beams. If we pass the reconstructed waves through another spherical lens in the Fourier transforming geometry, the original Fourier transform will reconstruct an inverted image, the conjugate of the transform will reconstruct the original image, the DC will reconstruct a planewave, and the final term will construct an autocorrelation of the object.

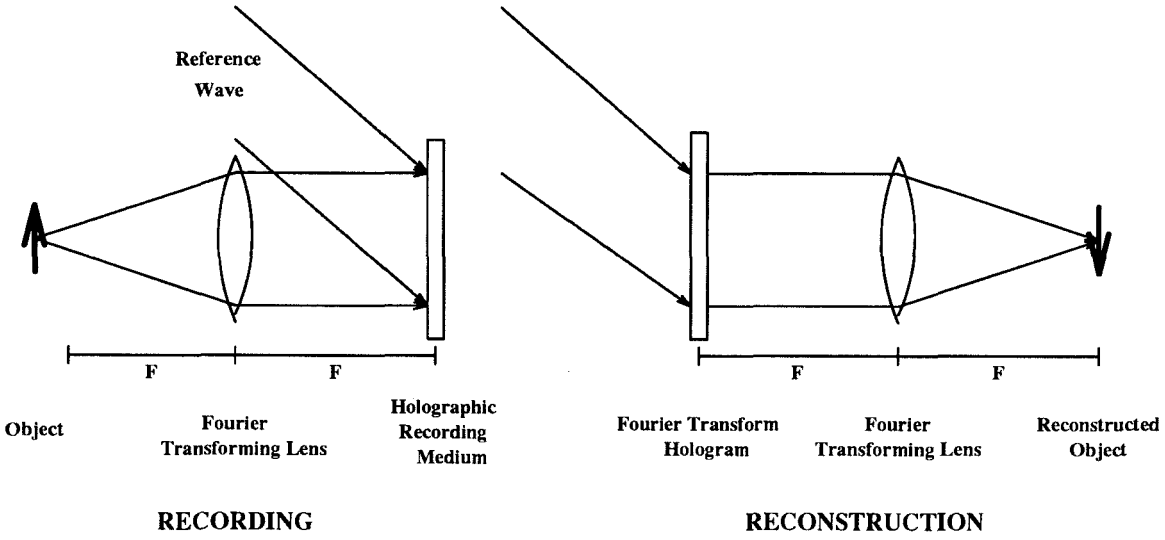



Fig. 3.30 - Recording and Reconstructing a Fourier Transform Hologram

### 3.3.2.4 Computer Generated Hologram

In all the holographic techniques that we have discussed thus far, we have assumed that the desired complex wavefront has been generated by a real object and interfered with a reference wave in order to capture both amplitude and phase information in a series of fringes. This pattern of fringes can then be faithfully recorded

in a medium which stores continuous amplitude-only information. With the advent of powerful digital computers, it is now possible to calculate the fringe pattern that would result from interference between arbitrary waves. These fringe patterns can then be recorded in a number of computer accessible media to produce holograms of objects. Some advantages of these computer generated holograms (CGHs) are as follows: unlike photographic films in conventional holography, for instance, the media used for CGHs typically require no processing; and the objects that generate the wavefronts to be recorded need never actually exist physically. Media, such as the SONY optical disk, for recording CGHs are typically binary and often pixellated. Brute-force hard-thresholding of the desired analog fringe pattern to produce binary fringes often introduces a great deal of noise into the reconstruction; optimization techniques may however be introduced while calculating binary fringes on the computer that reduce the effect of this noise in the desired reconstruction.

The first CGH was described by Brown and Lohmann in 1966 for spatial filtering applications.[25] In this type of hologram, the hologram itself is divided into cells of width $\Delta_x$ and height $\Delta_y$ as shown in Fig. 3.31. Within each cell is placed an aperture of height $h_{nm}$ and width $w$. The aperture is centered at coordinates $(x = n\Delta_x + p_{nm}, y = m\Delta_y)$. Each cell is responsible for encoding the amplitude $A_{nm}$ and phase $\phi_{nm}$ at its center $(x = n\Delta_x, y = m\Delta_y)$. In the Lohmann hologram, $h_{nm}$ and $p_{nm}$ are chosen using $A_{nm}$ and $\phi_{nm}$ respectively, thus encoding the phase and amplitude as follows:

$$h_{nm} = \frac{A_{nm}}{A_{max}}\Delta_y \tag{3.54}$$

$$p_{nm} = \frac{\phi_{nm}}{2\pi M}\Delta_x \tag{3.55}$$

where $M$ is a positive integer. We can then model the transmitance of the recorded hologram, and Fourier transform thereof using the following equation:

$$t(x,y) = \sum_{n,m} \text{rect}\left(\frac{x - n\Delta_x - p_{nm}}{w}, \frac{y - m\Delta_y}{h_{nm}}\right) \tag{3.56}$$

$$T(u,v) = \sum_{n,m} w h_{nm} \mathrm{sinc}(wu, h_{nm}v) e^{-j2\pi(u(n\Delta_x - p_{nm}) + vm\Delta_y)} \tag{3.57}$$

$$= \sum_{n,m} \left[ w \frac{A_{nm}}{A_{max}} e^{ju\Delta_x \phi_{nm}/M} \Delta_y \right.$$

$$\left. \mathrm{sinc}(wu, h_{nm}v) e^{-j2\pi(un\Delta_x + vm\Delta_y)} \right] \tag{3.58}$$

$$T\left(\frac{M}{\Delta_x} + u', v\right) = \frac{w\Delta_y}{A_{max}} \sum_{n,m} \left[ A_{nm} e^{j(\phi_{nm} + u'\Delta_x \phi_{nm}/M)} \mathrm{sinc}\left(w\left(\frac{M}{\Delta_x} + u'\right), h_{nm}v\right) \right.$$

$$\left. e^{-j2\pi((\frac{M}{\Delta_x} + u')n\Delta_x + vm\Delta_y)} \right] \tag{3.59}$$

The zeroth order term around $u' = v = 0$ is given by the following equation:

$$T\left(\frac{M}{\Delta_x} + u', v\right) = \frac{w\Delta_y}{A_{max}} \mathrm{sinc}\left(w\frac{M}{\Delta_x}, 0\right) \sum_{n,m} A_{nm} e^{j\phi_{nm}} \tag{3.60}$$

This corresponds to the pattern that we wish to record. Because the lateral position of the hologram encodes the phase at the center of the superpixel, this type of hologram is called a detour phase hologram.

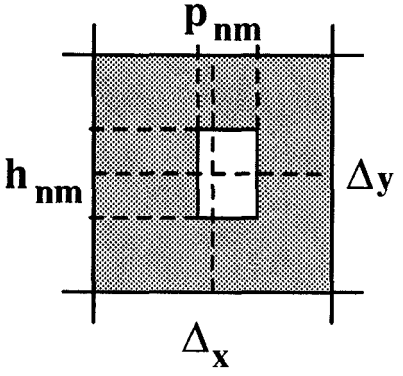



Fig. 3.31 - Lohmann Hologram

Lee developed another CGH,[26] often implemented in detour phase form, in which the complex amplitude $A_{nm}e^{\phi_{nm}}$ at the center of each superpixel is decomposed into nonnegative components along four basis vectors $\{1, i, -1, -i\}$. As shown in Fig. 3.32, the superpixel itself is laterally divided into four regions each representing one of the basis vectors and containing a centered aperture whose height is

proportional to the contribution of the basis vector that it represents. At most, two of the regions will contain apertures of nonzero height. Other detour phase CGHs have been proposed[27,28] in addition to the Lohmann and Lee holograms.

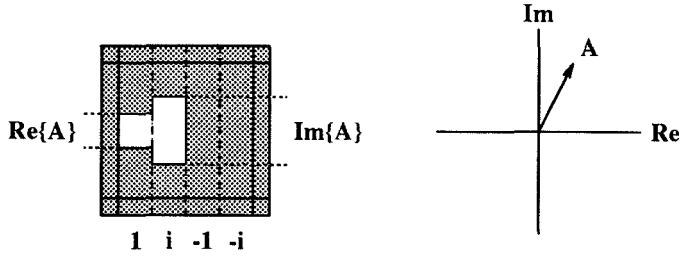



Fig. 3.32 - Lee Hologram

Lee was also responsible for developing a nondetour phase hologram in 1979.[29] In this method, the CGH is calculated placing fringes as if actually interfering the desired wavefront with a reference beam. The Lohmann method may be modified by positioning the aperture so that it encodes the phase and amplitude at the center of the aperture instead of the center of the cell.[30] Lee's original method was actually proposed in a nondetour phase form with the the height of each aperture encoding the magnitude of the component at the center of each aperture instead of the center of the cell. In Burch's method,[31] each cell contains a square aperture with area proportional to the value of the following function sampled at the center of the cell:

$$T(x,y) = \frac{1}{2}\left[1 + A(x,y)\cos\left(2\pi u x - \phi(x,y)\right)\right] \tag{3.61}$$

Effectively, the desired wavefront is placed on a carrier, combined with its conjugate to make it real, then added to a bias to make it positive-real. Other nondetour phase CGHs have been proposed[32] in addition to those mentioned here. For a historical review of CGHs, see the Ref. [33].

Because the CGH generation techniques described above are deterministic in nature, it is possible to compare them on the basis of bandwidth, efficiency, signal-to-noise ratio (SNR), *etc.*. In Ref. [34], these are compared with an eye to electron-beam fabrication of computer generated holograms. A typical e-beam hologram has

a space-bandwidth product of $10^{10}$, a feature size of $1/2\ \mu m$, and placement resolution of $.1\ \mu m$. These numbers are comparable with those of the optical disk. Optical efficiency and signal-to-noise ratio are image-dependent, but under the assumption of a planewave hologram for efficiency and a uniform random amplitude and phase distribution for SNR, Farhoosh, *et al.*, concluded that the non-detour Lohmann hologram was optimal in terms of bandwidth and efficiency, and the Burch in terms of SNR.

Many of the above approaches, though originally designed for spatial light modulators with a continuum of sizes and positions of the apertures, can be extended to SLMs like the optical disk which have discrete pixel locations. In this case, an array of pixels or superpixel is used to represent each cell. A group of pixels within the superpixel is then turned on in each cell. The number of pixels turned on, and their position, are chosen to best approximate the size and position of the aperture. The total SBWP used on the SLM is $N_x N_y M_x M_y$, where $N_x N_y$ is the number of samples in the desired wavefront ($N_x$ and $N_y$ being the number of samples in the $x-$ and $y-$dimensions respectively) and $M_x M_y$ is the number of pixels in each superpixel ($M_x$ and $M_y$ being the number of pixels in the $x-$ and $y-$dimensions respectively).

The approaches described thus far can be considered cell-based. There exist another group of so called pixel-based approaches. In a pixel-based approach, the $N_x \times N_y$ field of samples and its conjugate would be embedded in an $N_x M_x \times N_y M_y$ array of zeroes. This zero-padding operation has the effect of interpolating a new set of points in the transform domain, while the addition of the conjugate insures that the transform takes on only real values. If a unipolar SLM is used, a DC bias is added to the transform which contributes to a single "DC" pixel in the reconstruction. Because many spatial light modulators are binary in nature, a variety of techniques have been proposed for encoding the analog real transform

into a binarized representation. If we simply threshold the hologram to binarize it, we typically introduce quantization noise spread throughout the reconstruction. It would be desirable, however, to push the quantization noise out of the area containing the reconstructed wavefront and into the zero-padding area.

One distinct advantage of the pixel-based techniques is that they reconstruct fewer reconstruction-conjugate pairs. Because cell- or superpixel-based techniques have larger pixels corresponding to a lower sampling frequency, they will generate $M_x M_y$ reconstructions for every one generated by pixel-based techniques thus giving pixel-based techniques a higher diffraction efficiency. In addition, pixel-based techniques provide the zero-padding area into which quantization noise can be pushed through optimization techniques yielding higher SNRs.

Iterative quantization is one pixel-based technique that has been proposed.[35] In iterative quantization, we repeatedly transform between the reconstruction and hologram planes, imposing a different set of constraints in each domain (Fig. 3.33). When we transform to the hologram, we convert the continuous analog values therein into discrete quantized values. When we transform to the reconstruction, we set the values of the reconstructed points to their desired values, leaving the values in the zero-padded regions alone. Because we often care only about the magnitude and not the phase of the reconstruction, we often set only the magnitude of the reconstructed points to their desired values leaving the phase alone. The unspecified phase provides us with an additional degree of freedom that can be used to generate a hologram with higher efficiency or SNR. In other methods, a random phase is often imposed on the reconstruction in order to reduce the DC component in the hologram, because if we use a constant phase in the reconstruction plane and an SLM with limited dynamic range, we are often forced to choose between faithfully representing either the DC peak or the rest of the reconstruction frequencies–either case yielding a distorted reconstruction. Although the random

phase allows us to more faithfully represent all the frequencies, it is often undersampled resulting in a noisy reconstruction. This noise takes the form of bright spots in dark regions and dark spots in bright regions called speckle. However, because the magnitude is controlled in iterative quantization, the resulting phase generates much less speckle-type noise. The basic iterative quantization problem has been found to have a problem in that it can become stuck in an unsatisfactory solution. It has been found that these "stuck-states" are the result of the hard quantization of values near decision boundaries. Wyrowski[36] has proposed a technique that begins with a soft threshold that becomes harder as the number of iterations increases.

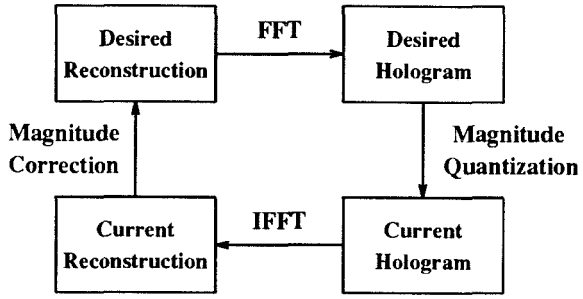



Fig. 3.33 - Iterative Quantization

Error diffusion is another technique, popular for binarization of images, that has been applied to binarization of holograms.[37] In error diffusion, the entire hologram is scanned pixel by pixel. Each pixel is hard-thresholded, and the resulting error between the desired and thresholded-values is used to adjust the desired values of unthresholded neighbors. By choosing the way in which error is assigned to neighboring pixels, one can choose the way the resulting noise is directed in the reconstruction.

In the direct binary search (DBS) technique,[38] a random hologram is initially chosen and scanned pixel-by-pixel. Each pixel is individually flipped and the resulting reconstruction compared to the original. If the error in the reconstruction is reduced, the change is kept; otherwise, the pixel is flipped back. This process

is repeated until the entire hologram is scanned with no flips thus converging to a locally optimum hologram. Because we are flipping only single pixels and are interested in the result only in a window of the reconstruction, one can calculate the result of a flip more efficiently by direct calculation in the window instead of applying an FFT to the entire hologram.

A variation of DBS is akin to simulated annealing in that pixel flips that increase error are kept probabilistically with lower probability the larger the error and the lower the system temperature. The temperature follows a cooling schedule that starts at high temperature and drops monotonically to a lower temperature so that initially almost all changes are kept, but later only those that reduce the error are kept. As in simulated annealing, we expect that with the appropriate cooling schedule, the final hologram might converge to the global optimum with probability approaching unity. If there are many local optima of similar quality to the global optimum, the iterative technique will more quickly converge to a good hologram. However if there a very few local optima comparable to the global optimum, the simulated annealing method may be desirable. Seldowitz, *et al.*, have found that the resulting holograms from the DBS and simulated annealing algorithms are of similar quality.

In measuring the error between the actual and desired reconstruction, there is an arbitrary scale that corresponds to the diffraction efficiency of the final hologram. This scaling factor can either be set *a priori* thus imposing a desired diffraction efficiency, or it can be adjusted dynamically at any point in the algorithm to most closely match the current and desired reconstructions. Although one expects a tradeoff between efficiency and SNR, Seldowitz, *et al.*, have found that dynamic adjustment of efficiency achieves maximum SNR; one can achieve a higher efficiency at the cost of lower SNR, but one gets lower SNR by lowering efficiency further.

Finally, Kobayashi has proposed a technique[39] similar to DBS but cast in

the form of a Hopfield-style network as follows. Let $G_{mn}$ represent the desired reconstruction and $U_{mn}$ the actual reconstruction of the binary hologram $u_{mn}$. Because we do not care about the light outside the reconstruction region, we define a window $W_{mn}$ equal to 1 in the region of interest and 0 outside. The window function may be chosen to have a soft transition at the boundary. We define the following reconstruction error:

$$E = \sum_{mn} ||G_{mn} - \alpha W_{mn} U_{mn}||^2 \tag{3.62}$$

where $\alpha$ is proportional to the desired hologram diffraction efficiency. By Parseval's theorem, this energy can be rewritten as follows:

$$E = \sum_{mn} \left( g_{mn} - \alpha \sum_{kl} u_{kl} w_{m-k,n-l} \right)^2 \tag{3.63}$$

$$= \sum_{mn} (g_{mn}^2 - 2\alpha g_{mn} \sum\sum_{kl} u_{kl} w_{m-k,n-l} + \alpha^2 \sum_{kl} \sum \kappa\lambda u_{kl} w_{m-k,n-l} u_{\kappa\lambda} w_{m-\kappa,n-\lambda}) \tag{3.64}$$

$$= -\frac{1}{2} \sum_{kl \neq \kappa\lambda} u_{kl} u_{\kappa\lambda} T_{kl,\kappa\lambda} + \sum_{kl} u_{kl} I_{kl} + C \tag{3.65}$$

$$T_{kl,\kappa\lambda} = \begin{cases} -2\alpha^2 \sum_{mn} w_{m-k,n-l} w_{m-\kappa,n-\lambda} & kl \neq \kappa\lambda \\ 0 & kl = \kappa\lambda \end{cases} \tag{3.66}$$

$$I_{kl} = -2 \sum_{mn} g_{mn} w_{m-k,n-l} \tag{3.67}$$

$$C = \alpha^2 \sum_{kl} \sum_{mn} w_{m-k,n-l}^2 + \sum_{mn} g_{mn}^2 \tag{3.68}$$

The error can thus be viewed as the energy of a Hopfield network (Subsec. 1.5.13) with the hologram represented by the state of the neurons, the connections derived from the window function, and the input derived from the window and the desired reconstruction. Since $T$ is symmetric with zero diagonal, the state of the network will converge to a local minimum of the energy function. Like Seldowitz, Kobayashi found that there exists a hologram efficiency $\alpha$ corresponding to a maximal value

of SNR. Although Kobayashi's technique and DBS are quite similar, Kobayashi's technique is in a form suitable for efficient implementation on a neural network parallel processor.

#### 3.3.2.5 Fresnel Holograms on Disks

A number of authors[10–14] have proposed utilizing holographic storage of information on optical disks. The use of diffraction patterns on optical disks to scan a laser beam to read Universal Product Code information in supermarket checkout stands is actually in commercial use. We have successfully recorded a number of CGHs on our disks. Seiji Kobayashi, a visiting researcher in our group, was primarily responsible for the development of holographic readout techniques.

Figure 3.34 shows the reconstruction of a Fresnel hologram of the names "Sony" and "Caltech" recorded on a write-once optical disk. This hologram was calculated by entering and recording the positions of a series of discrete points tracing out the two names. The calculated Fresnel diffraction pattern at a focal length of 10 *cm* was then sampled and thresholded to generate a 1024×1024 pixel fringe pattern. This pattern was then recorded on the disk. The hologram was reconstructed by illumination with a raw HeNe laser beam passed through a hole in a planar screen placed 10 *cm* from the disk surface. A component of the reflected light then generates a reconstruction of "Sony" and "Caltech" at the screen. Figure 3.35 shows the reconstruction of a Fresnel hologram of the acronym "CIT" recorded on a magnetooptic disk.

#### 3.3.2.6 Fourier Holograms on Disk

Figure 3.36 shows the reconstruction of a Fourier Transform hologram of a cartoon chicken recorded on the optical disk. This chicken was entered on a 64×64 grid of points. A random phase was superimposed over the image to reduce the DC peak. An FFT of the data is used to transform the data into its Fourier

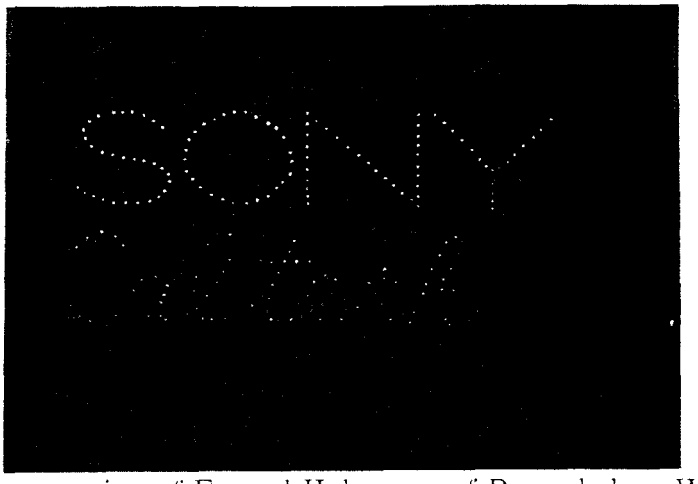


Fig. 3.34 - Reconstruction of Fresnel Hologram of Recorded on Write-Once Disk

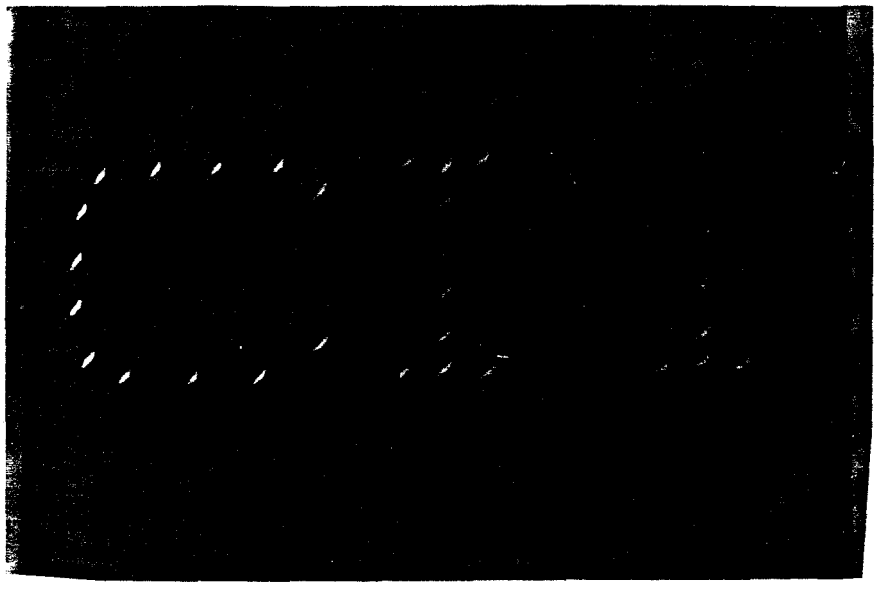


Fig. 3.35 - Reconstruction of Fresnel Hologram Recorded on Magnetooptic Disk

representation. Each complex element in the FFT is recorded in a $4 \times 1$ superpixel that represents 9 different possible combinations of amplitude and phase as shown in Fig. 3.37. The entire hologram is repeated twice in each direction during recording to generate more tightly focused spots in the reconstruction.

A cartoon short 16 frames long, with the chicken laying an egg which hatches and grows into a chicken is recorded on the disk. The holograms for each frame are recorded side-by-side at the same radius on the disk. When a laser beam strikes

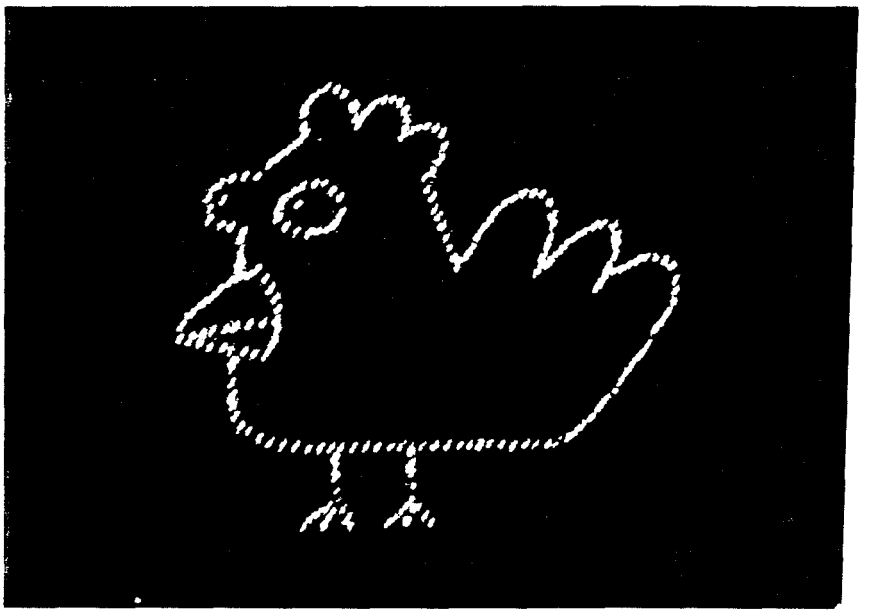


Fig. 3.36 - Reconstruction of Fourier Hologram Recorded on Write-Once Disk

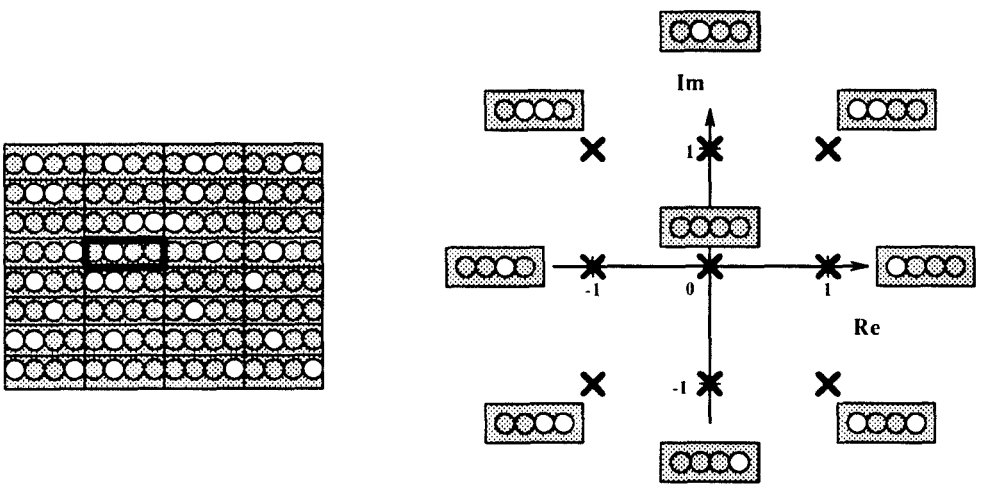



Fig. 3.37 - Fourier Hologram Superpixel

the disk at that radius, with the reflected light passing through a Fourier transform lens and the disk spinning, the cartoon short is reconstructed.

### 3.3.2.7 Kobayashi Hologram on Disk

Figure 3.38 shows the reconstruction of another Fourier Transform hologram of the cartoon chicken recorded on the optical disk. In this case, the pixel-based Kobayashi technique was used.

Fig. 3.38 - Reconstruction of Kobayashi Hologram

### 3.3.3 Alignment Considerations During Parallel Readout

Alignment essentially consists of tracking (radial alignment) and timing (azimuthal alignment) considerations in a disk system. Ideally, once the detector has been aligned with a certain track with the disk spinning at a constant velocity, the data recorded on that track should automatically become aligned with the detector at certain predictable times. Unfortunately, nonidealities such as disk center offset, disk wobble, *etc.*, make this idealistic alignment rather unachievable. Here, we consider a number of these nonidealities and their effects in the various readout systems.

The difference between the center of recorded data on the disk and the disk rotational center is called disk center offset. The specification for disk center offset in the SONY disk system is a maximum of 70 $\mu m$. As the disk rotates, then, a detector at a fixed position would see a variation of $\pm 70$ $\mu m$ in the radial position of a block of data on the disk.

Disk wobble is a measure of the angle between the detector and disk data planes. Surface irregularity and improper seating can cause disk wobble. The specification for disk wobble in the SONY disk system is a maximum of .6 °. Disk wobble introduces a deviation in the propagation direction of light diffracted from the disk.

The area on the disk from which we read information out in parallel can be misaligned with the detector or head. As shown in Fig. 3.39, we can have four basic types of misalignment: rotation, translation, focusing, and tilt. If we define the optical axis of the system as the line passing through the center of the detector and normal to its surface, rotation is measured by an angle $\phi$ around the optical axis. Translation $\delta_x$ measures the offset of the center of the information on the disk from the optical axis. Focusing error $\delta_z$ measures the offset of the center of the information on the disk from its intended location along the optical axis. Tilt $\theta$ measures the angle between the normal of the surface on which the information is recorded and the optical axis. In the following section, we examine how each of these errors in alignment distort the readout image or reconstruction. This distortion takes a number of forms: rotation, translation, defocusing, spatial distortion, and phase distortion. Rotation $\phi'$ of the readout data is measured around the optical axis. Translation $\delta'_x$ is measured from the center of the detector. Defocusing is measured by a spot size $\delta'_s$ larger than the diffraction limit. Spatial distortion is measured by $\Delta'_D$, twice the distance between the center and edge of the readout data. With intensity-sensitive detectors, we ignore phase distortion.

Consider the geometry shown in Fig. 3.40, for an imaging readout system. There is a detector of width $\Delta_D$ and an imaging lens with focal length $F$ and aperture $A$ placed a distance $s_i$ from the detector. The image recorded on the disk has width $\Delta_I$ and is supposed to be a distance $s_o$ from the information recorded on the disk with width $\Delta_I$. We must choose the system parameters such that the

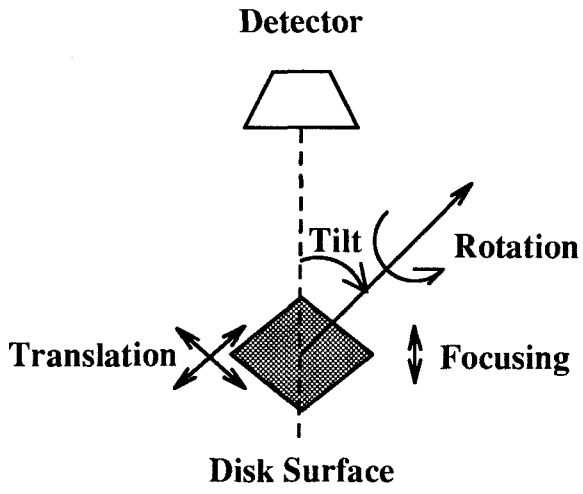



Fig. 3.39 - Optical Disk - Neural Network Chip Misalignment

following imaging condition is met:

$$\frac{1}{s_o} + \frac{1}{s_i} = \frac{1}{f} \tag{3.69}$$

In this case, we have a magnification $M = s_i/s_o$ between the information on the disk and the detector. Assume that the magnification is chosen such that the recorded information is imaged exactly onto the detector such that $\Delta_D/\Delta_I = M$.

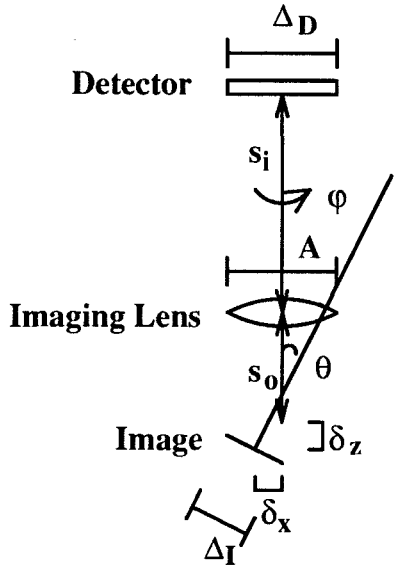



Fig. 3.40 - Alignment Considerations for Imaging Readout

Rotation of the recorded information through an angle $\phi$ results in an equal

rotation of the readout pattern $\phi' = \phi$. Translation of the recorded information by $\delta_x$ results in a magnified translation of the readout data by $\delta'_x = (s_i/s_o) * \delta_x$.

From geometric optics, we see that an offset of the recorded data from the object plane by a distance $\delta_z$ results in a change in magnification and in spot size. Magnification changes to $s_i/(s_o+\delta_z) \approx M(1-\delta_z/s_o)$ such that $\Delta'_D = (1-\delta_z/s_o)\Delta_D$. As the object distance $s_o$ changes, the image distance $s_i$ changes as follows: $ds_i = -(s_i/s_o)^2 ds_o$. According to geometric considerations, light that passes through a lens of aperture $A$ to form a point in the image at distance $s_i(A + ds_i)$ from the lens will form a spot of size $A|ds_i|/s_i = A(s_i/s_o^2)|\delta_z|$ at a distance $s_i$ from the lens. Although the spot size is diffraction limited for very small sizes, as the imaging system deviates from the ideal, geometric considerations begin to dominate.

As the object plane tilts, the imaging characteristics change due to two considerations. First, points in the object plane experience a change in effective object distance. Second, is the change in effective spacing in the object plane as the viewing aspect changes (a tilted square looks like a rectangle). The former is a first order change whereas the latter is a second order change dependent on the tilt angle $\theta$. The maximum deviation due to focusing error is at the edge of the object plane and given by $\pm(\Delta_I/2)\theta$ yielding maximum spot size $(A\Delta_D/2s_o)|\theta|$ and maximum distortion $(1 \pm (\Delta_I\theta/2s_o))\Delta_D$.

Next, consider the Fresnel holographic system of Fig. 3.41. We have a Fresnel reflection hologram of size $\Delta_H$ and focal length $F$. Upon illumination, the hologram reconstructs an encoded pattern of size $\Delta_D$ onto a detector of equal size. A rotation of the hologram by $\phi$ produces an equal rotation of the reconstruction $\phi' = \phi$. A translation of the hologram by $\delta_x$ produces an equal translation of the reconstruction by $\delta'_x = \delta_x$.

As in the imaging case, a focusing error (translation of the hologram along the optical axis) results in both a change in the scale of the reconstruction and

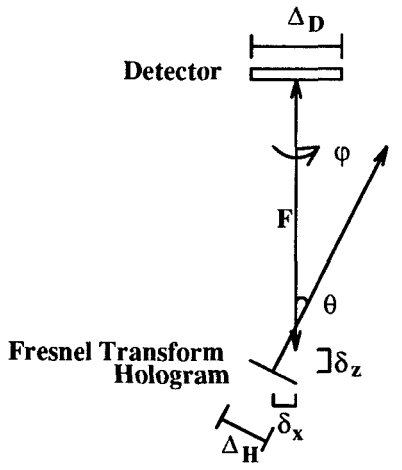



Fig. 3.41 - Alignment Considerations for Fresnel Holographic Readout

an increase in the spot size. In many ways, we can treat the Fresnel holographic system the same way we treat the imaging system if we replace the aperture $A$ of the imaging system with the hologram size $\Delta_H$ and the image distance $s_i$ with the focal length $F$. In addition, we must add the focusing error $\delta_z$ to the image distance $s_i$ instead of the object distance $s_o$. Once again using geometric optics, we find that scale of the reconstruction changes to $\Delta'_D = (1 + \delta_z/F)\Delta_D$ and the spot size $\delta'_s = (\Delta_H/F)|\delta_z|$.

In the Fresnel holographic system, tilt of the hologram results in the reconstruction of a tilted object. As in the imaging case, this tilt results in image distortion and spot size bounded by those corresponding to a focusing error in the object plane of $(\Delta_D/2)\theta$. Using this measure in the equations for focusing error in the Fresnel system, we find distortion $\Delta'_D$ bounded $(1 \pm (\Delta_D\theta/2F)\Delta_D$ and spot size $\delta'_s < (\Delta_H\Delta_D/2F)|\theta|$. In addition, reflection in the Fresnel holographic system results in a translation of the reconstruction corresponding to $\delta'_x = 2F\theta$.

Finally, we consider the Fourier holographic system of Fig. 3.42. We have a Fourier transform hologram of size $\Delta_H$ placed a focal length $F$ in front of a spherical lens. Upon illumination, the system produces a reconstruction of size $\Delta_D$ on a detector of equal size placed a focal length $F$ behind the lens. As described

in Subsubsec. 3.3.2.3, the reconstruction corresponds to the 2-D Fourier transform of the hologram. A rotation of the hologram by an angle $\phi$ produces an equal rotation of the reconstruction $\phi' = \phi$. A translation of the hologram corresponds to a phase variation in the reconstruction with no change of the intensity pattern. Likewise, a focusing error (translation along the optical axis) also results in phase-only variation in the reconstruction. In fact, even tilt of the hologram does not create the distortion and spotsize problems in the imaging and Fresnel holographic systems. As in the Fresnel system however, tilt of a Fourier hologram by an angle $\theta$ produces a translation of the reconstruction by a distance $\delta'_x = 2F\theta$.

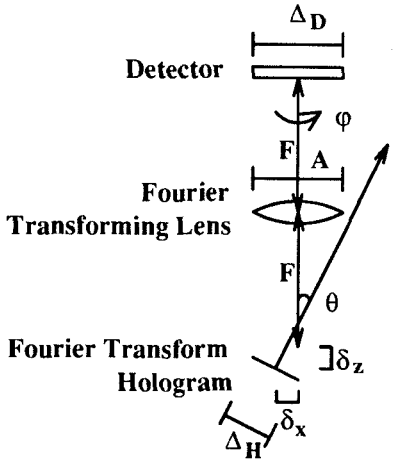



Fig. 3.42 - Alignment Considerations for Fourier Holographic Readout

It was not obvious to us at first that rotation of the disk would lead to motion in the reconstruction. Fig. 3.43 shows a series of double exposures of reconstructions from a hologram on the disk with reconstructions from the same hologram with different amounts of disk rotation. As the figure shows, rotation of the disk leads to rotation of the reconstruction and thus misalignment between the reconstruction and a detector. Fig. 3.44 shows the measured rotation of the reconstruction $\phi'$ versus rotation of the disk $\phi$ and confirms that $\phi' = \phi$.

Table 3.4 summarizes the errors caused by the different forms of misalignment with resulting limitations in imaging, Fresnel, and Fourier holographic systems.

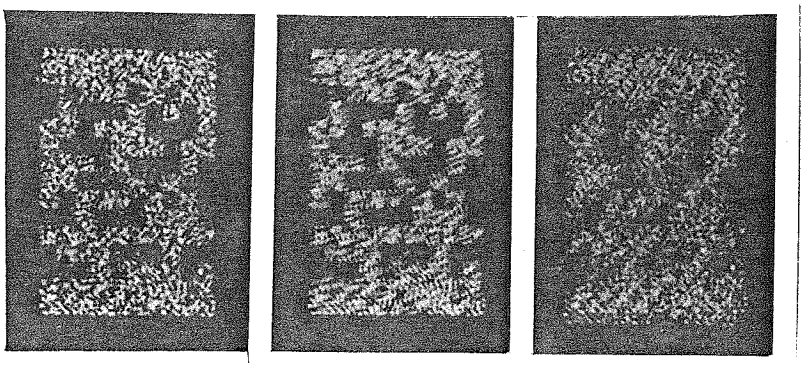


Fig. 3.43 - Effect of Disk Rotation on Fourier Hologram Reconstruction

We now consider how different forms of misalignment affect the parallel readout of information in disk systems. Disk center offset, disk wobble, *etc.* can be broken down into effective rotation, translation, focusing, and tilt errors of the disk surface that the tracking system must accomodate at a rate less than the readout rate. In addition, rotation of the disk results in rotation and translation errors that must be accomodated with a time constant corresponding roughly to the pixel rate.

Rotation, translation, and distortion of the readout pattern all contribute to translation of individual pixels of the readout. When this translation exceeds one-half the size of a detector, we assume that we can no longer read out the information. Alignment in the azimuthal direction is handled by proper timing operation of the light source and/or the detector. Rotation of the disk causes rotation and translation of the readout image and a dwell time during which the readout pattern may be aligned with the detector to within one-half the size of a single detector element. In an imaging system, the dwell time corresponds to the time it takes the disk to rotate half the size of a pixel or superpixel recorded on the disk. In a Fresnel holographic system, the dwell time corresponds to the time it takes the disk to rotate half the size of a detector element. In a Fourier holographic system, the

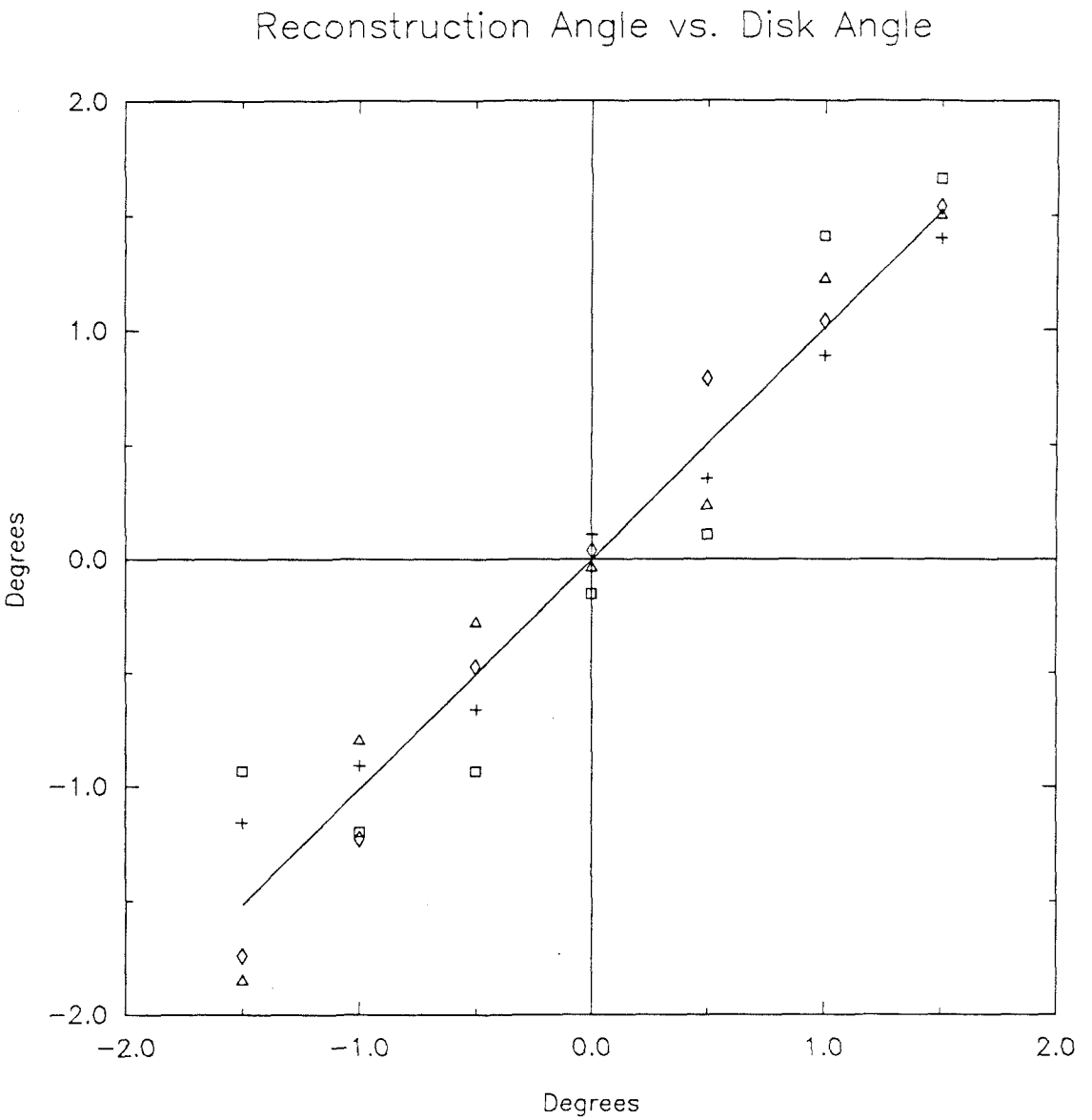



Fig. 3.44 - Rotation Angle of Reconstruction vs. Disk Rotation

Table 3.4 - Misalignment in Parallel Readout From Optical Disks

| | Imaging | Fresnel | Fourier |
|---|---|---|---|
| $\phi'$ | $\phi$ | $\phi$ | $\phi$ |
| $\delta'_x$ | $\frac{s_i}{s_o}\delta_x$ | $\delta_x + 2F\theta$ | $2F\theta$ |
| $\frac{\Delta'_D}{\Delta_D}$ | $1 + \frac{\lvert\delta_z\rvert}{s_o} + \frac{\Delta_I\lvert\theta\rvert}{2s_o}$ | $1 + \frac{\lvert\delta_z\rvert}{F} + \frac{\Delta_D\lvert\theta\rvert}{2F}$ | 1 |
| $\delta'_s$ | $A\frac{s_i}{s_o}\lvert\delta_z\rvert + \frac{\Delta_D}{2s_o}\lvert\theta\rvert$ | $\frac{\Delta_H}{F}\lvert\delta_z\rvert + \frac{\Delta_H\Delta_D}{2F}\lvert\theta\rvert$ | $\frac{\lambda F}{A}$ |

dwell time corrsponds to the time it takes to rotate the disk sufficiently to cause

an effective translation of a reconstructed pixel through a distance corresponding to one-half the detector spacing. If the hologram is small enough in a Fourier holographic system, the dwell time will be limited by the time it takes to rotate through the hologram instead of misalignment due to rotation. For the Sony disk system, misalignment due to rotation can be ignored for Fourier holograms up to 300 pixels on a side[21]. Table 3.5 summarizes the dwell time estimates for the various parallel readout techniques where the reconstruction has $N_x \times N_y$ pixels, $n_x \times n_y$ pixels on the disk are used to encode each pixel of the reconstruction, $\Delta_x \times \Delta_y$ is the size of a detector element, and *omega* is the angular velocity of the disk.

Table 3.5 - Dwell Time for Parallel Readout

| Readout Scheme | Dwell Time |
|---|---|
| Imaging | $\frac{n_x \Delta_\theta}{2\omega R}$ |
| Fresnel | $\frac{\Delta_x}{2\omega R}$ |
| Fourier (Misal. Ltd.) | $\frac{\Delta_x}{2\omega\sqrt{(N_x\Delta_x)^2+(N_y\Delta_y)^2}}$ |
| Fourier (Hol. Size Ltd.) | $\frac{n_x N_x \Delta_\theta}{2\omega R}$ |

Assuming $n_x$=10, $N_x = N_y$=100, and $\Delta_x = \Delta_y = 50\Delta_\theta$, the Fourier hologram will be misalignment limited, and Fourier readout will provide a dwell time twice as long as does imaging readout and 10 times as does Fresnel. If we use a pulsed illumination system, this dwell time corresponds to the maximum pulse-width and time resolution of the optical source. If we use continuous illumination, the dwell time corresponds to the maximum amount of time that we can guarantee valid readout from the detector elements.

Because radial alignment is handled through mechanical motion of the detector in current systems, the radial speed of the disk must be significantly lower than the azimuthal speed. Assuming that translation and tilt misalignment are dominated by disk center offset and a uniform tilt of the disk surface, we find that the maximum translational misalignment speed is on the order of $4\nu\delta_0$, twice the maximum disk

center offset times twice the disk rotation rate $\nu$, and the maximum time derivative of tilt is on the order of $4\nu\theta_0$, twice the maximum tilt times twice the rotation rate. Using Table 3.4 with these estimates, we can find the maximum translational misalignment speed of the readout pattern. In an imaging system, the maximum readout translational misalignment speed corresponds to the maximum translational misalignment speed of the disk times the magnification. In the Fourier system, it is twice the maximum time rate of tilt times the focal length. In the Fresnel system, it is the maximum translational misalignment speed plus twice the maximum time rate of tilt times the focal length. Table 3.6 summarizes the translational misalignment speed estimates for the various parallel readout techniques.

Table 3.6 - Translational Misalignment Speed for Parallel Readout

| Readout Scheme | Trans. Mis. Speed |
|---|---|
| Imaging | $4(s_i/s_o)\nu\delta_0$ |
| Fresnel | $4\nu\delta_0 + 8\nu\theta_0 F$ |
| Fourier | $8\nu\theta_0 F$ |

Assuming $100 \times 100$ readout pixels with $1000 \times 1000$ holograms, detectors with $50\mu m$ spacing, we need to use $F \approx 50mm$ in the Fresnel and Fourier systems. Using the parameters for the Sony system, the imaging system has one fourth the translational misalignment speed of the Fourier system and one fifth that of the Fresnel system.

Although we would like to minimize misalignment in both azimuthal and radial directions while using the least disk space possible, translation is more critical in the radial direction while disk space is more critical in the azimuthal direction. Translation is more critical in the radial direction because it requires mechanical motion of the head versus the simpler timing adjustments required for azimuthal alignment. Disk space is more critical in the azimuthal direction because it not only reduces storage capacity, but also the peak readout rate for a given disk rotation

rate. If the disk wobble can be reduced, the Fourier transform would provide the best performance in terms of translation while the imaging system consumes the least disk space (at least when we consider the transfer of binary information). Thus, imaging in the azimuthal direction with a Fourier hologram in the radial direction may provide the best solution. This solution would require the design of anamorphic optical elements.

### 3.3.4 Effects of Track Curvature

In most applications, it would be most convenient if the pixels within each block that we read out in parallel from the disk were arranged in a 1- or 2-D Cartesian array. Thus far, we have assumed this to be the case and have ignored the fact that pixels are actually arranged in a 1- or 2-D approximately polar array along essentially circular tracks. In this subsection, we explicitly consider the effects of track curvature on our parallel readout techniques and consider techniques for obviating these effects.

Before we begin, note that track curvature only affects the parallel readout of a single block of information because we record the information serially in the azimuthal direction. Were we to record the information in parallel in the azimuthal direction using a linear source or modulator array, the pixels within each block would be arranged in a Cartesian array although the blocks themselves would be in an essentially polar array. For example, when we arrange the information within each block in a 1-D array in the across-track direction, because the array is only one-pixel wide in the azimuthal direction, we meet the condition of recording the information within the block in parallel in the azimuthal direction. In this case, there is no difference between a 1-D Cartesian array and a 1-D polar array with constant azimuth ($\theta$).

Our assumption that the pixels are arranged in a Cartesian array is valid if we

restrict attention to a small area of the disk. To see this, consider a region at a distance $R$ from the disk center. As shown in Fig. 3.45, we establish a Cartesian coordinate system with x-axis in the azimuthal or along track direction and y-axis in the radial or across track direction with origin centered in the region of interest. The following equation can then be used to convert between the polar coordinates of the disk and the Cartesion coordinates we have established:

$$\left\{ \begin{array}{l} x = r\sin\theta \\ \\ y = r\cos\theta - R \end{array} \right\} \tag{3.70}$$

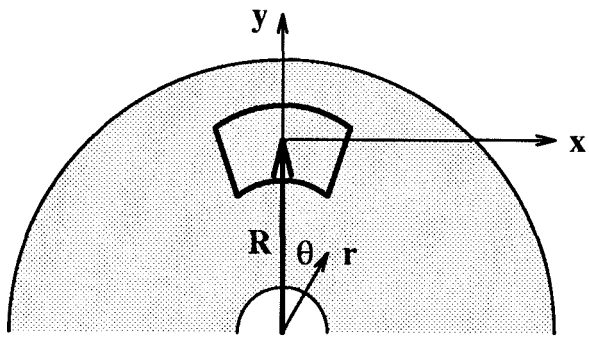



Fig. 3.45 - Coordinate Transformation Axes

The center-to-center spacing of pixels in the radial dimension is $\Delta_r$ and the angular separation between adjacent pixels is $\Delta_\theta$ in azimuth. We now superimpose a Cartesian grid on this pixel structure with $x$ and $y$ spacings as follows:

$$\left\{ \begin{array}{l} \Delta_x = R\Delta_\theta \\ \\ \Delta_y = \Delta_r \end{array} \right\} \tag{3.71}$$

This choice for $\Delta_x$ and $\Delta_y$ provides the best match between the pixels recorded on the disk and the points on the Cartesian grid. We now calculate the deviation of the pixel locations from their presumed Cartesian locations. The presumed coordinates of the $n, m^{th}$ point on the Cartesian grid is

$$\left\{ \begin{array}{l} x' = n\Delta_x \\ \\ y' = m\Delta_y \end{array} \right\} \tag{3.72}$$

whereas its actual location is

$$\left\{ \begin{array}{l} r = R + m\Delta_r + n\dfrac{\Delta_\theta \Delta_r}{2\pi} \\ \theta = n\Delta_\theta \end{array} \right\} \tag{3.73}$$

Therefore, the actual Cartesian coordinates of the recorded pixels are

$$\left\{ \begin{array}{l} x = \left( R + m\Delta_r + n\dfrac{\Delta_\theta \Delta_r}{2\pi} \right) \sin(n\Delta_\theta) \\ \quad \approx n\Delta_x + nm\dfrac{\Delta_x \Delta_y}{R} \\ y = \left( R + m\Delta_r + n\dfrac{\Delta_\theta \Delta_r}{2\pi} \right) \cos(n\Delta_\theta) - R \\ \quad \approx m\Delta_y + n\dfrac{\Delta_x \Delta_y}{2\pi R} - n^2 \dfrac{\Delta_x^2}{2R} \end{array} \right\} \tag{3.74}$$

We calculate the deviation between the actual and presumed pixel positions by subtracting Eq. 3.72 from Eq. 3.74 as follows:

$$\left\{ \begin{array}{l} \epsilon_x = x' - x \\ \quad \approx nm\dfrac{\Delta_x \Delta_y}{R} = \dfrac{x'y'}{R} \\ \epsilon_y = y' - y \\ \quad \approx n\dfrac{\Delta_x \Delta_y}{2\pi R} - n^2 \dfrac{\Delta_x^2}{2R} = x' \dfrac{\Delta_y}{2\pi R} - \dfrac{x'^2}{2R} \end{array} \right\} \tag{3.75}$$

For an array of $1000 \times 1000$ pixels on the Sony disks, the worst case pixel placement error is 1.25% of the array size (12.5 pixels) in the $x$-direction (at $R = 3\ cm$, $\Delta_x = 0.5\ \mu m$, and $\Delta_y = 1.5\ \mu m$) and 0.14% of the array (1.4 pixels) in the $y$-direction (at $R = 6\ cm$, $\Delta_x = 1\ \mu m$, and $\Delta_y = 1.5\ \mu m$).

For diffractive readout, the light reflected from each pixel carries both amplitude and phase information. The amplitude will not be sigifnicantly affected unless the position error is of the same order as the distance between the disk and detector or lens. However, the phase can be significantly affected by even a slight change in pixel position. Given a wavevector $\underline{k}$ between a hologram on the disk and a

diffracted beam, the phase error is given by multiplying the wavevector with the position error:

$$\phi_e(x, y; \underline{k}) = \underline{k}.\epsilon(x, y) \tag{3.76}$$

For example, if we consider the first order track-diffracted beam, $k_x = 0$ and $k_y = 2\pi/\Delta_y$. For light diffracted in this direction, track curvature induces the following phase error:

$$\phi_e\left(x, y; k_x = 0, k_y = \frac{2\pi}{\Delta_y}\right) = \frac{k}{2f}x^2 \tag{3.77}$$

$$f = R\frac{k}{k_y} \tag{3.78}$$

This phase error can be modeled as a cylindrical lens at the disk plane with focal length $f = \pm R\Delta_y/\lambda$. For typical experimental parameters: $R = 4.5\ cm$, $\Delta_y = 1.5\ \mu m$, and $\lambda = 633\ nm$, the cylindrical focal length is $10.7\ cm$.

For our imaging systems, we are primarily concerned with the effect of track curvature induced pixel position error on misalignment between recorded pixels and detector elements. The significance of this position error in imaging readout depends upon the application. For the readout of images, the slight curvature of the image induced by track curvature may be negligible. Fig. 3.26 shows an image 3000 $\times$ 4500 pixels in size with no apparent visible distortion. However, for the readout of random data where a single pixel error can be significant, systematic pixel position error could be fatal. We can compensate for this error in either the image, the detector, or some combination of both. For instance, given the position at which we will record the data on the disk, we can predict how the image will be warped by track curvature. By "pre-warping" the image such that after it is recorded on curved tracks, the positions of the pixels on the disk match precisely those on the detector, we can effectively eliminate the effect of track curvature. By designing the detector such that the individual detecting elements themselves are on a polar coordinate grid instead of a Cartesian grid, we can also eliminate the effect

of track curvature. Because curvature varies between the inner and outer tracks, pre-warping remains the desirable method since it can compensate for the track curvature at any point on the disk. A customized detector can only be designed optimally for a single set of tracks on the disk. However, since all tracks are curved, a certain amount of compensation in the detector design, although unnecessary, could be helpful.

For the holographic systems, we can compensate for the effective phase error in a couple of ways. First, as with images, we can obviate the position error by assuming a Cartesian pixel array and pre-warping the hologram before recording. Second, we can eliminate the position error by calculating the hologram under the assumption that it will be recorded on a polar pixel array in the first place. Finally, we can reconstruct the hologram using illumination containing a compensatory phase–for example a cylindrical wave for a track-diffracted reconstruction. As in the imaging readout case, the latter course is undesirable because the position error (and thus required compensatory phase) depends upon the location at which we record the information on the disk and we must change an element in the illumination system as we shift from track to track. Because algorithms such as the Fast Fourier Transform exist that can be used to rapidly calculate the expected diffraction from a Cartesian grid, the first course is probably the most practical, although the second course may be pursued given sufficient computational resources.

### 3.4 Applications and Future Directions

Parallel readout from optical disks has many potential applications. First, it can be used to simply increase the bandwidth of data read out from the disk. Second, it is a natural storage format for applications requiring rapid access to a large library of information. Examples of the latter case include optical correlation architectures such as the one shown in Fig. 3.46. Neifeld, *et al.*, describe numerous

architectures for optical correlation.[37,38] Correlation rates corresponding to over 26 million binary operations per second (BOPS) have been demonstrated, with maximum correlation rates on the order of $10^{11}$ BOPS.

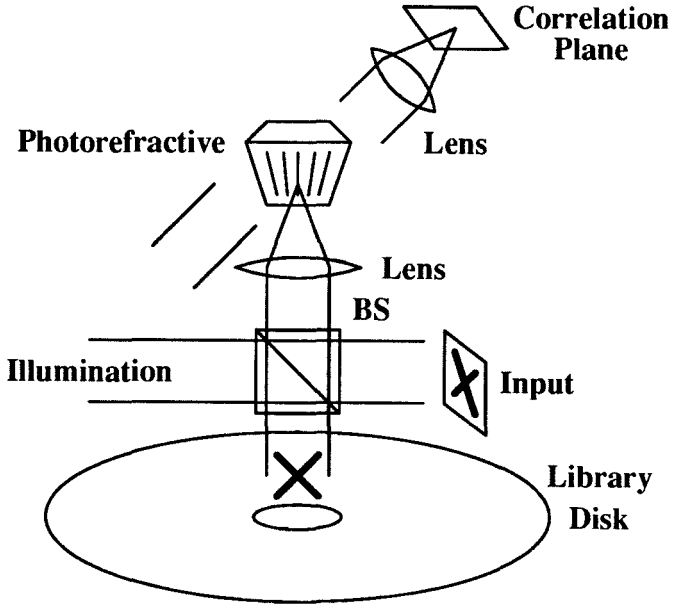



Fig. 3.46 - Optical Disk Correlator

An optical character recognition system which uses an optical disk correlator to compare an input character with a library of stored templates and electronically postprocesses the correlation results has demonstrated an accuracy of 83%.

Database preprocessors are another obvious application of optical parallel readout. Databases must naturally be stored on high capacity archival storage. An optical preprocessor thus can rapidly search for desired keywords on the disk and mark the records that have a potentially good match for further electronic postprocessing.[39,40] Figure 3.47 shows how a simple optical preprocessing system can be built to rapidly perform database query operations. A query is encoded on an input spatial light modulator (SLM). The light is imaged from the SLM to the disk. The reflected light contains the inner product which is integrated onto a photodetector. If the inner product exceeds a threshold, salient portions of the record are latched into a photodetector array and passed into an electronic buffer for further electronic

postprocessing. A single rotation of the disk is all that is required to process a single query.

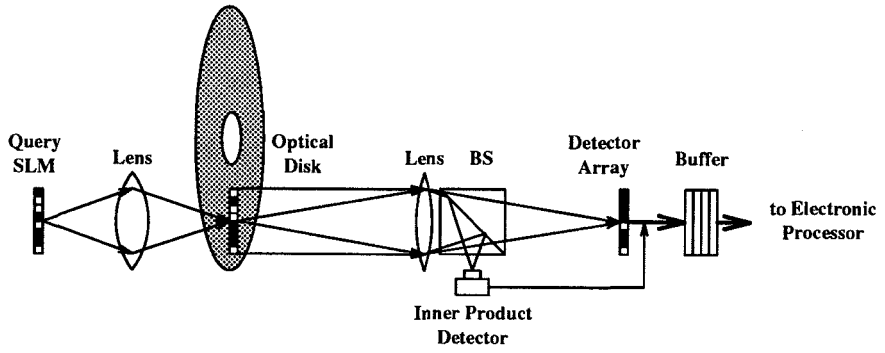



Fig. 3.47 - Optical Database Preprocessor

An optical disk can also be used to rapidly change the connection patterns in optoelectronic neural networks. The disk can play one of two roles in this application. As Figure 3.48 shows, the optical disk can either (a) act simply as storage for the connection patterns or (b) play a role in implementing the connections themselves. The Optoelectronic Neural Network Chip described in Chap. 4 is an example of the former.

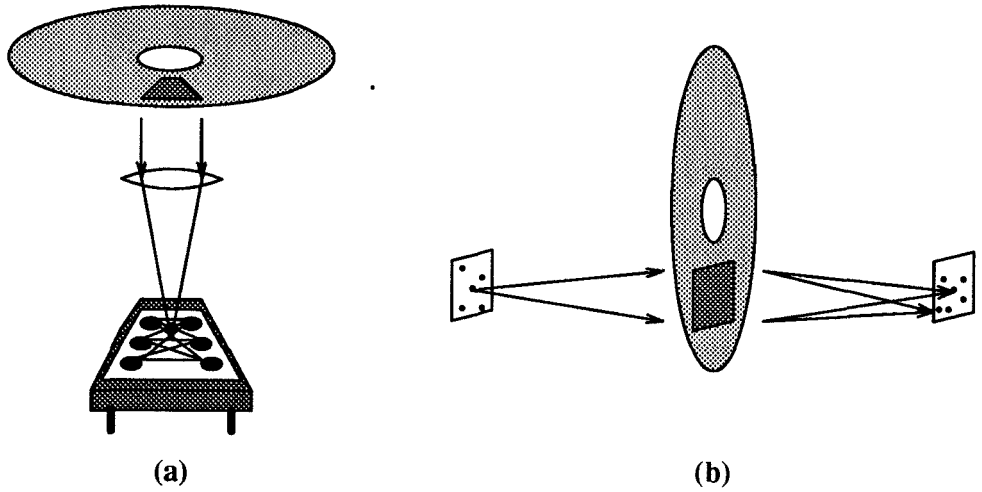



Fig. 3.48 - Optical Disk (a) Storage
and (b) Storage and Implementation of Neural Connections

It is difficult to precisely estimate the maximum bandwidth for a parallel read-

out optical disk. Using exotic technology like air bearings with plastic disks, the maximum disk rotation rate would be about $35000 rpm$[44]. If we were able to maintain alignment at this disk rotation rate, given a $100 \times 100$ array of detector elements, we could transfer $10 Gbps$ of data from the disk to a high speed detector or special purpose processor.

In addition to the ease of alignment offered by Fourier transform holographic encoding of the data, we could build shift-register like structures into the detector array to electronically "shift" the position of the head faster than possible mechanically. The amount of shift required could be determined by enconding special alignment markers into the parallel readout data.

In terms of future research into disk technology itself, the main thrust is into making smaller lighter heads for faster access times and a variety of techniques such as shorter wavelengths, multiple wavelengths, superresolution, and 3-D storage to increase storage density. Of these techniques, 3-D storage as depicted in Fig. 3.49 seems to hold the most potential for orders of magnitude increase of optical storage density.

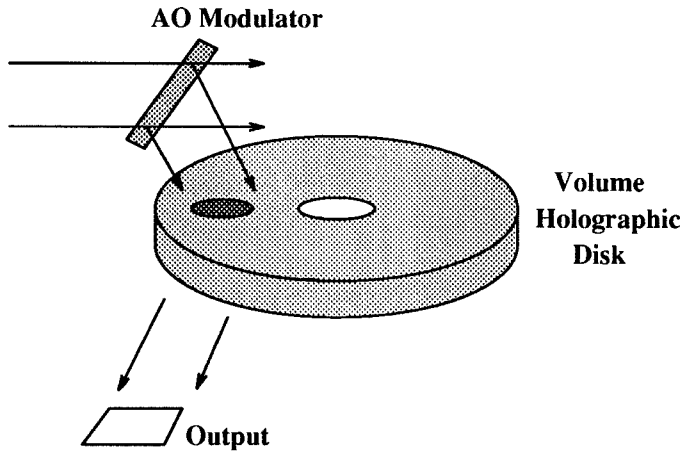



Fig. 3.49 - Volume Holographic Storage in Optical Disks

## 3 References

[1] J. Isailović, *Videodisc and Optical Memory Systems*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1985.

[2] J. Hecht, "Magnetic Discs Inch Forward in The Race for Data Storage," *New Scientist*, Vol. 125, No. 1699, p. 40, Jan., 1991.

[3] W.E. Moerner, "Photon-Gated Persistent Spectral Hole-Burning," *Int'l Symp. on Opt. Mem. 1989*, pp. 119-120, Sep 1989.

[4] Y. Yamanaka, Y. Hirose, and K. Kubota, "High Density Optical Recording by Superresolution," *Int'l Symp. on Opt. Mem. 1989*, pp. 99-100, Sep 1989.

[5] D. Psaltis, A.A. Yamamura, and H. Li, "Mass Storage for Digital Optical Computers," *Proceedings of SPIE OE-LASE*, Jan. 1990.

[6] Y. Nakagome, *et al.*, "An Experimental 1.5-V 64-Mb DRAM," *IEEE Jour. Solid State Circ.*, Vol. 26, No. 4, pp. 465-472, Apr. 1991.

[7] R.A. Bartolini, H.A. Weakliem, and B.F. Williams, "Review and Analysis of Optical Recording Media," *Optical Engineering*, Vol. 15, No. 2, March-April, 1976, pp. 99-108.

[8] F.M. Smits and L.E. Gallagher, "Design Considerations for a Semipermanent Optical Memory," *Bell System Technical Journal*, Vol. XLVI, No. 6, Jul-Aug 1967, pp. 1267-1278.

[9] J.P. Huignard, F. Micheron, and E. Spitz, "Optical Systems and Phototosensitive Materials for Information Storage," *Optical Properties of Solids*, B.O. Seraphin, ed., Ch. 16, pp. 847-925, North Holland, Amsterdam, 1976.

[10] A.D. Mikaélyan, *et al.*, "Holographic Disk for Data Storage," *Sov. J. Quant. Elec.*, Vol. 17, No. 5, pp. 680-687, May 1987.

[11] I. Satoh and M. Kato, "Holographic Disk Recording of Digital Data with Fringe Stabilization," *Applied Optics*, Vol. 27, No. 14, pp. 2987-2992, 15 July 1988.

[12] Y. Tsunoda, *et al.*, "Holographic Video Disk: An Alternative Approach to

Optical Video Disks," *Applied Optics*, Vol. 15, No. 6, pp. 1398-1403, June 1976.

[13] W.F. Heagerty, "Ideographic Composing Machine," *Applied Optics*, Vol. 9, No. 10, pp. 2291-2294, October 1970.

[14] T. Yatagai, J.G. Camacho-Brasilio, and H. Onda, "Recording of Computer-Generated Holograms on an Optical Disk Master," *Proc. SPIE*, Vol. 1052, pp. 119-124, 1989.

[15] J.H. Rilum and A.R. Tanguay, Jr., "Utilization of Optical Memory Disks for Optical Information Processing," in *Technical Digest, OSA Annual Meeting*, paper M15, 1988.

[16] Y. Nakane, *et al.*, "Principle of Laser Recording Mechanism by Forming an Alloy in the Multilayer of Thin Metallic Films," *Proceedings of the SPIE Optical Mass Data Storage*, Vol. 529, pp. 76-82, 1985.

[17] D. Psaltis, E.G. Paek, and S.S. Venkatesh, "Optical Image Correlation with a Binary Spatial Light Modulator," *Opt. Eng.*, Vol. 23, pp. 698-704, 1984.

[18] J.W. Goodman, *Introduction to Fourier Optics*, McGraw-Hill Publ. Co., San Francisco, CA, 1968.

[19] H. Fizeau, *Ann. Chim. Phys.*, Vol. 3, No. 66, pf. 429, 1862.

[20] J.F. Jarvis, C.N. Judice, and W.H. Ninke, "A Survey of Techniques for the Display of Continuous Tone Pictures on Bilevel Displays," *Computer Graphics Image Processing*, Vol. 5, pp. 13-40, 1976.

[21] D. Psaltis, M.A. Neifeld, A. Yamamura, and S. Kobayashi, "Optical Memory Disks in Optical Information Processing," *Applied Optics*, Vol. 29, No. 14, pp. 2038-2057, 10 May 1990.

[22] D. Gabor, *Nature*, Vol. 161, pf. 777, 1948.

[23] J.B. DeVelis and B.O. Reynolds, *Theory and Applications of Holography*, Addison-Wesley Publishing Company, Reading, MA, 1967.

[24] H.M. Smith, *Principles of Holography*, J. Wiley and Sons, New York, NY, 1969.

[25] B.R. Brown and A.W. Lohmann, "Complex Spatial Filtering with Binary Masks," *Applied Optics*, Vol. 4, pf. 387, 1966.

[26] W.H. Lee, "Sampled Fourier Transform Holograms Generated by Computer," *Applied Optics*, Vol. 9, pf. 639, 1970.

[27] C.K. Hsueh and A.A. Sawchuck, "Computer Generated Double Phase Holograms," *Applied Optics*, Vol. 17, pf. 3874, 1978.

[28] C.B. Burckhardt, "A Simplification of Lee's Method of Generating Holograms by Computer," *Applied Optics*, Vol. 9, pf. 1949, 1970.

[29] W.H. Lee, "Binary Computer Generated Holograms," *Applied Optics*, Vol. 18, pf. 3661, 1979.

[30] B.R. Brown and A.A. Lohmann, "Computer Generated Binary Holograms," *IBM Journal of Research and Development*, Vol. 13, pf. 160, 1969.

[31] J.J. Burch, "A Computer Algorithm for the Synthesis of Spatial Frequency Filters," *Proc. IEEE*, Vol. 55, pf. 599, 1967.

[32] S.M. Arnold, "Electron Beam Fabrication of Computer Generated Holograms," *Optical Engineering*, Vol. 24, pf. 803, 1985.

[33] G. Tricoles, "Computer Generated Holograms: An Historical Review," *Applied Optics*, Vol. 26, No. 20, pp. 4351-4360, 15 Oct. 1987.

[34] H. Farhoosh, *et al.*, "Comparison of Binary Encoding Schemes for Electron-Beam Fabrication of Computer Generated Holograms," *Applied Optics*, Vol. 26, No. 20, pp. 4361- 4372, 1987.

[35] P.M. Hirsch, J.A. Jordan, and L.B. Lesem, "Method of Making an Object-Dependent Diffuser," U.S. Patent 3,619,022, 1971.

[36] F. Wyrowski, "Iterative Quantization of Digital Amplitude Holograms," *Applied Optics*, Vol. 28, No. 18, pp. 3864-3870, 15 September 1989.

[37] E. Barnard, "Optimal Error Diffusion for Computer Generated Holograms,"

*JOSA A*, Vol. 5, No. 11, pp. 1803-1827, Nov. 1988.

[38] M.A. Seldowitz, J.P. Allebach, and D.W. Sweeney, "Synthesis of Digital Holograms by Direct Binary Search," *Applied Optics*, Vol. 26, No. 14, pp. 2788-2798, 15 July 1987.

[39] S. Kobayashi, *Technical Digest, OSA Annual Meeting*, 1990.

[40] D. Psaltis, M. Neifeld, and A. Yamamura, "Image Correlators Using Optical Memory Disks," *Optics Letters*, Vol. 14, No. 9, pp. 429-431, 1989.

[41] M. Neifeld, Ph.D. Thesis, California Institute of Technology, 1991.

[42] P.B. Berra and N.B. Troullinos, "Optical Techniques and Data/Knowledge Base Machines," *IEEE Computer*, pp. 59-70, Oct. 1987.

[43] D. Psaltis, *et al.*, "Parallel Readout of Optical Disks," *Optical Computing Technical Digest Series*, Vol. 9, pp. 206-209, Feb. 1989.

[44] Personal communication with engineers of the Sony Corporation.

# 4 An Optoelectronic Multilayer Feedforward Neural Network

## 4.1 Introduction

Most neural network research involves testing of the networks. Because neural networks are complex nonlinear systems that are difficult to theoretically analyze, everything from theoretical results to *ad hoc* algorithms based on heuristics are usually confirmed through testing. This testing typically involves many "runs" to compare the different sets of parameters or different networks and to generate sufficient statistics for analysis. Most of this testing consists of simulation of neurons and connections on general computational platforms ranging from lowly PCs to the most powerful supercomputers. Unfortunately, as the complexity and especially the size of networks increases, the time required to simulate the networks can impede research progress.

Because of this simulation problem, many researchers interested in neural networks are working to develop special purpose hardware to implement the networks. Although many physical processes may be used as analogs to the processes occuring in neural networks, electronics and optics are the primary technologies being used to implement special neural network hardware.

### 4.1.1 Electronic Implementations

Although we may consider the implementation of a neural network in software running on a general computing platform ranging from a PC to a Cray as

an "electronic implementation," we will concentrate on electronic hardware specifically designed for the implementation of neural networks. There exists an enormous number of special purpose electronic implementations of neural networks in the literature. Here, we will concentrate on those that can implement multilayer feedforward neural networks. This excludes a large body of exciting research such as the neuromorphic analog VLSI systems of Mead.[1]

Even after narrowing the field to special-purpose electronic hardware for the implementation of the multilayer feedforward neural network, there still remains a large number of implementations to consider because of the popularity of this network model. These networks have been implemented at chip, wafer, and board level in both analog and digital forms. Some have chosen to implement neural networks using systolic arrays of special purpose processors designed to perform rapid, high accuracy, digital multiplication and addition using DSP chips, multiplier chips, and/or (semi-)custom integrated circuits. Pomerleau, *et al.*[2] achieved 17 million connections per second (MCPS) using a systolic array of 10 processors operating at 5 $MHz$. In this case, a connection counted both forward propagation of an input and back propagation of the error. More recently, systolic arrays capable of 25 to 50 MCPS per processor have been reported.[3–6] Advantages of this approach include flexibility, high speed, and accuracy, and disadvantages include large area and high power consumption–aspects characteristic of their general purpose digital computing brethren. Because these implementations require areas corresponding to one or more computer boards, the number of processors is relatively small and the neurons are typically time-multiplexed on the processors. Nevertheless, because of their raw computing power, these implementations are able to compete in terms of speed with other current electronic implementations. However, for most neural network applications, the high accuracy calculations that these implementations work so hard to provide are not necessary, and a better compromise may be achieved by

trading off accuracy for lower power consumption, greater fault-tolerance, *etc.*

Most of the remaining implementations that we discuss are single-chip implementations of neural networks. Here, we summarize the results of a comparison by Holler[7] of a number of VLSI implementations. Micro Devices has a chip with eight neurons and a single time-multiplexed synapse per neuron performing a digital multiplication and accumulation between a binary input and a 16-bit weight. AT&T has a chip which contains an array of synapses that multiply a digital weight with a binary input to generate a current which gets summed in an analog fashion before hard-thresholding by a neuron.[8] Adaptive Solutions has a chip called CNAPS which is a systolic array of 64 processors running at 25 $MHz$ with digital fixed point multiplication and addition on a single chip.[9] Intel has a chip provides nonvolatile EEPROM storage of connection weights.[10]

### 4.1.2 Optical Implementations

There have also been many optical implementations of neural networks. Many of these implementations are variations of the canonical vector-matrix multiplier based associative memory[11] where the intensity of each LED in a linear array represents the output of a neuron, the transmittance of a 2-D mask the connection strength between a pair of neurons, and the light incident on each photodetector in a linear array the input of the neuron. By expanding the light from each LED such that it illuminates a column of the 2-D mask and integrating the light from each row onto a photodetector, each neuron receives as input the weighted sum of the outputs of the other neurons. A dual-rail encoding scheme in the mask with a pairs of photodetectors in the linear array can be used to implement bipolar weights.

Psaltis *et. al.* have demonstrated an optical associative memory[12] using a Hughes liquid crystal light valve (LCLV)[13] as a 2-D array of neurons. The output reflectivity of the neurons is roughly proportional to a soft threshold applied to the

input light intensity. The output of the neurons is readout by a laser beam and enters a optical system (Vanderlugt correlator[14]) that performs a 2-D correlation between the output of the neurons and a library of reference images stored in holographic form. The correlation between the neuron output and each reference is then sampled by a pinhole array and used to reconstruct a superposition of the reference images on the input side of the neuron array. Each reference image is weighted by the magnitude of its correlation with the output of the neurons. When the loop is closed, the output of the neurons tends to be driven towards the reference image that most closely matches the initial state of the neurons. The associative memory loop has been demonstrated capable of recalling stored images given a certain degree of distortion and noise at the input. With approximately 400×400 resolution elements on the LCLV, the optical memory loop implements some 160000 neurons and the connections between them, thus demonstrating the massive interconnection capabilities afforded by optics.

Photorefractive crystals can be used to dynamically store volume holograms. Because they store in volume, they provide $O(L^3)$ degrees of freedom to arbitrarily specify connections where $L$ scales as the linear dimension of the system. This means that photorefractives can be used to store arbitrary connections between an input and output plane consisting of $O(l_1)$ and $O(l_2)$ neurons respectively where $l_1 l_2 = L^3$. For example, we can choose $l_1 = l_2 = L^{3/2}$. Fractal sampling grids can be used to specify the location of the neurons in each of the neural planes.[15]

Because photorefractives can be dynamically recorded and erased, they can be used in architectures with learning. An optical implementation[16] of the perceptron algorithm (Subsec. 1.5.3) has been demonstrated where holograms are recorded in a photorefractive that upon illumination with a training pattern reconstruct the same pattern which is then integrated onto a point detector. The amount of light incident on the photodetector in response to a training pattern depends in part

on the strength of the hologram associated with that pattern. The strength of this hologram can be adjusted by interfering the training pattern with itself in the photorefractive. A piezoelectric mirror can be used to adjust the modulation depth and phase of the interference pattern which then either strengthens or weakens the hologram. A computer interfaced to the experiment can then cycle through the training patterns and selectively strenghten or weaken the holograms so that the output of the photodetector is either above or below a given threshold depending on the classification of the pattern. An optical implementation of a multilayer feedforward neural network with back error propagation (Subsec. 1.5.6) has also been proposed.[17]

### 4.1.3 Optoelectronic Implementations

Although most optical implementations involve some aspects of electronics, we will mainly consider here those applications where each plays a strong role. Typically, each technology is used in areas where they are strong–electronics for processing and optics for connections and communications.

Agranat, *et al.*, of Caltech have designed a CCD based neural network[18] with optical loading of the synaptic weights onto a CCD detector, followed by selective gating of the charge representing each weight, according to the activation of the corresponding neuron, into an accumulator for summation with subsequent electronic detection.

Ohta, *et al.*, of Mitsubishi have constructed a 32 neuron optoelectronic chip[19] with a metal connection mask sandwiched between crossed strip LEDs and strip photodetectors, thus implementing a collapsed version of the vector-matrix architecture described in Subsec. 4.1.2. The intensity of each LED represents the output of a neuron, the transmittance of each mask element the connection strength between a pair of neurons, and the total light incident on each photodiode the input of

each neuron. The photogenerated current through each photodiode is proportional to the weighted sum of the neuron outputs.

Rietman, *et al.*, of Bell Labs have demonstrated a 120 neuron optoelectronic chip[20] with an amorphous silicon photoconductor sandwiched between orthogonal conducting strips. The conducting strips on the "top" side of the chip are made of ITO, a transparent conductor. The current through the photoconductor at the intersection between crossed conductors approximates the product of the applied voltage and a monotonic function of the incident light. By proper encoding of light intensity, the Bell Labs chip can also generate a series of output currents proportional to the weighted sum of neuron outputs. In all three of these optoelectronic implementations, the thresholding operation is performed off-chip and the result fed back to the optoelectronic chips.

## 4.2 An Optoelectronic Multilayer Feedforward Neural Network

In this section, we describe special-purpose hardware that implements certain types of neural networks using available, mature technologies. Fig. 4.1 shows the basic idea behind the system. A VLSI circuit called the Optoelectronic Neural Network Chip (ONNC) is designed incorporating both electronic neurons and electronic connections. Photodetectors are placed on the chip so that the pattern and strength of the connections can be set by illuminating the chip with different light patterns. A library of different light patterns that program the chip to implement different functions or parts of functions are encoded and stored on an optical disk and optically read out in parallel from the disk to the chip.

### 4.2.1 Design of the ONNC

The neurons and synapses on the ONNC are laid out in a crossbar as shown in Fig. 4.2 so that each neuron can potentially be connected to any other. The design incorporates optically reconfigurable unipolar connnections and hard-thresholding

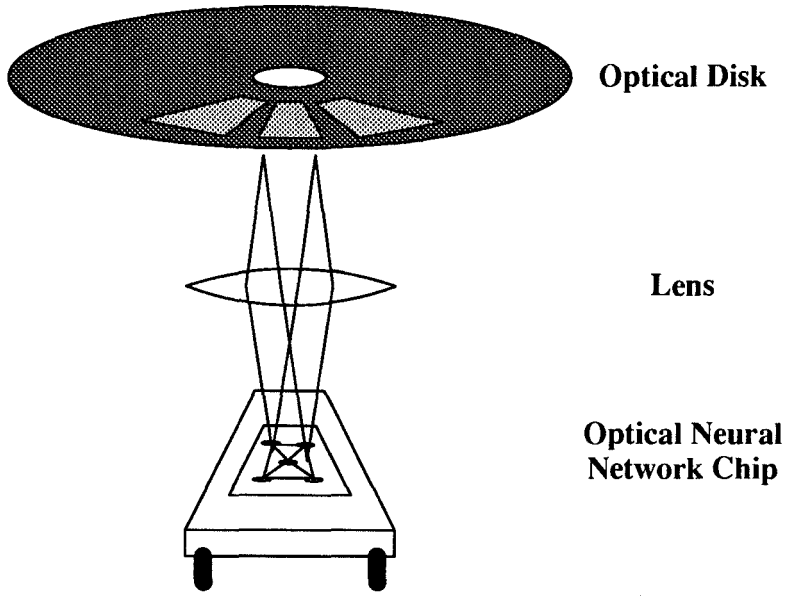



Fig. 4.1 - Optical Disk/Chip Architecture

binary bipolar neurons. We first discuss the decision to provide optically reconfigurable connections, then describe the actual hardware design and operation of the neurons and synapses.

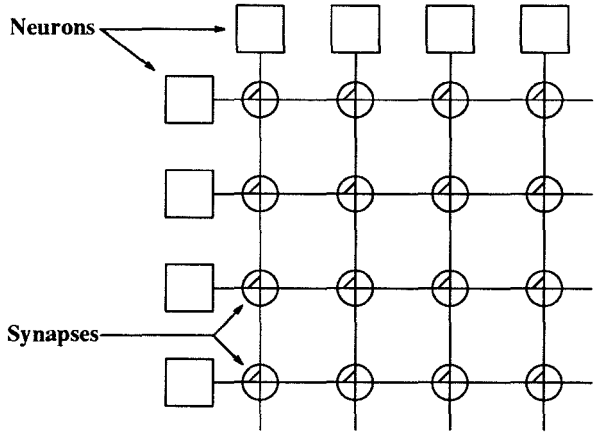



Fig. 4.2 - Crossbar Configuration of Neurons and Synapses

#### 4.2.1.1 Reconfigurable Chips

In a typical neural network, the processing performed at each neuron is fixed and the functionality determined by modifying the pattern and strength of connections. When building special purpose hardware to implement neural networks we

typically fix the function of the neurons but must choose between fixed and reconfigurable connections. In some applications, we wish to implement a single specific function using a connection pattern that can be determined once and then fixed. In this and only this case, we can build a network with fixed connections on our chip. In all other cases, we must use reconfigurable connections to provide more flexible functionality. This flexible functionality may simply consist of choosing from a set of predetermined connection patterns much as the function of a conventional computer may be altered by choosing from a set of machine code programs. This is the path chosen for the system described in this chapter. Alternatively, the flexible functionality may actually consist of learning and/or adaptation of the connection pattern and function.

Reconfigurable connections also provide another advantage for networks that are too large to fit on a single chip or wafer. When using fixed connections we must divide the network into smaller pieces with off-chip connections between neurons on separate chips. These off-chip connections are considerably slower than on-chip connections and can limit the maximum processing rate. With certain types of networks, it is possible however to use reconfigurable connections to time-multiplex the hardware and implement a large network on a smaller chip. In this case, although we pay a penalty in pipelined processing rates, we can reduce the required hardware and possibly the delay time as well.

#### 4.2.1.2 On-Chip vs. Off-Chip Storage

The pattern and strength of connections used to program the function of a reconfigurable chip must be stored either on- or off-chip. If they are stored on-chip, they consume valuable area that could otherwise be used for more neurons and connections. Because the capacity of a network and the complexity of functions that it can implement depend on the number of neurons and connections, we would

prefer to place as large a number of neurons and connections as possible on our chip. Further, although there is a fixed cost in area for the circuitry to communicate connection information from off-chip to the appropriate synapses, the marginal cost in area of storing an additional weight will exceed the fixed cost of communication circuitry after a small number of weights.

#### 4.2.1.3 Optical Reconfigurability

Assuming off-chip storage of connection information, we may access the information either electronically or optically. For electronic access, we see a potential communications bottleneck. In a crossbar architecture with $N$ neurons, we need to specify $N^2$ potential connections between them. However, using one side of the crossbar to select a row of connections and the other to convey information to every column in that row, we can only load $N$ neurons at a time with information–thus requiring $N$ cycles to fully program the connections. Even if we could provide individual lines to each connection, the number of pads or external communication channels will only grow as $O(N)$, once again leading to $O(N)$ cycles to program the connections. With optical access, however, we can bring all the connection information in parallel from above the chip in a fixed amount of time no matter how large the chip.

Both electronic and optical access may ultimately be limited by the communications bandwidth of the mass storage element. However, optical reconfiguration still has an advantage since it interfaces nicely with parallel optical readout from optical storage elements such as the optical disk (see Chap. 3).

#### 4.2.1.4 Fabrication Technology

The ONNC is fabricated using a 2.0 $\mu m$ p-well CMOS process with double-metal and double-poly layers. This means that the smallest linewidth we can specify on the chip is 2.0 $\mu m$ wide. The chip is built on an n-type substrate into which

we can fabricate p-channel MOSFETs; we can also construct p-wells into which we can fabricate n-channel MOSFETs. With both NMOS and PMOS FETs, we have Complementary MOS or CMOS. We also have two mask layers of metal and two mask layers of polysilicon to use as interconnection and, in the case of polysilicon, as gate material. This process allows us to build good capacitors using two parallel plates of polysilicon with a thin layer of gate-oxide in between. Most importantly for our chip, as pioneered by Mead,[1] one can construct both photodiodes and phototransistors. Figure 4.3 shows the basic structures that we can use.

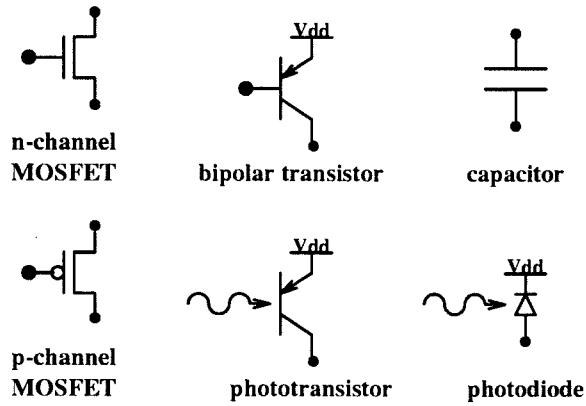



Fig. 4.3 - Available Components

Because we will allow light to strike our device, we must cover all circuitry excluding photodetectors with an opaque layer of material. We sacrifice the second layer of metal for this purpose leaving a single metal layer for interconnection. The second metal layer may still be used to distribute power or global signals.

#### 4.2.1.5 The Synapse

Because most of the chip area will be consumed by synapses, optimization of the synapse design is by far the most important. Although a large number of approaches may be taken for the design of optically reconfigurable synapses, we decide to minimize the size of the synapse in order to maximize the number of connections available on the chip. Figure 4.4 shows a circuit diagram of the

synapse that we have designed. The synapse circuit can be divided into two parts: the connection and the photodetector. The connection consists of a single synaptic PFET that connects the output of one neuron to the input of the next. The voltage on the gate of this FET determines the strength of the connection. This gate voltage is controlled by the photodetector which consists of a pulldown PFET and a reverse-biased photodiode. With no incident light, the photodiode is essentially an open circuit and the gate voltage will be low, turning the synaptic transistor on. With light incident on the photodiode, current will flow from the positive voltage ($V_{dd}$) rail, raising the gate voltage and turning the synaptic transistor off.

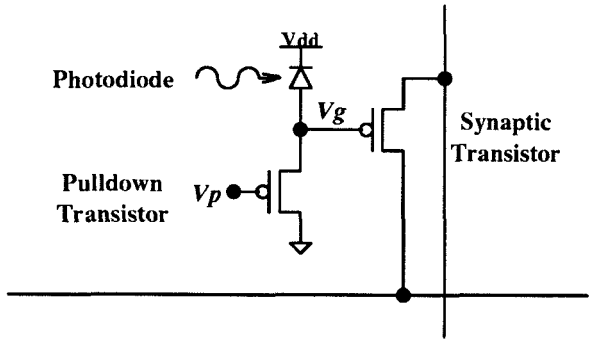



Fig. 4.4 - Circuit Diagram of Synapse

Figure 4.5 shows the layout and Fig. 4.6 a photograph of rougly six synapses. The dark regions are the photodiodes and only the metal connections and contacts are easily discernible. Minimization of area given a square-shaped synapse was a major concern in the synapse design in order to maximize the number of connections and generate a square array of synapses. Because the pixel-spacing on the optical disk varies from a 2:1 aspect ratio at the inner recording radius to a 1:1 aspect ratio at the outer recording radius, there is no single optimal aspect ratio for spacing the photodetectors. A square aspect ratio was chosen because, although not universal, it is the most obvious standard for SLMs in general.

The synapse was not quite designed to be of minimal size. Although smaller synapses increase the maximum number of connections, larger photodiodes simplify

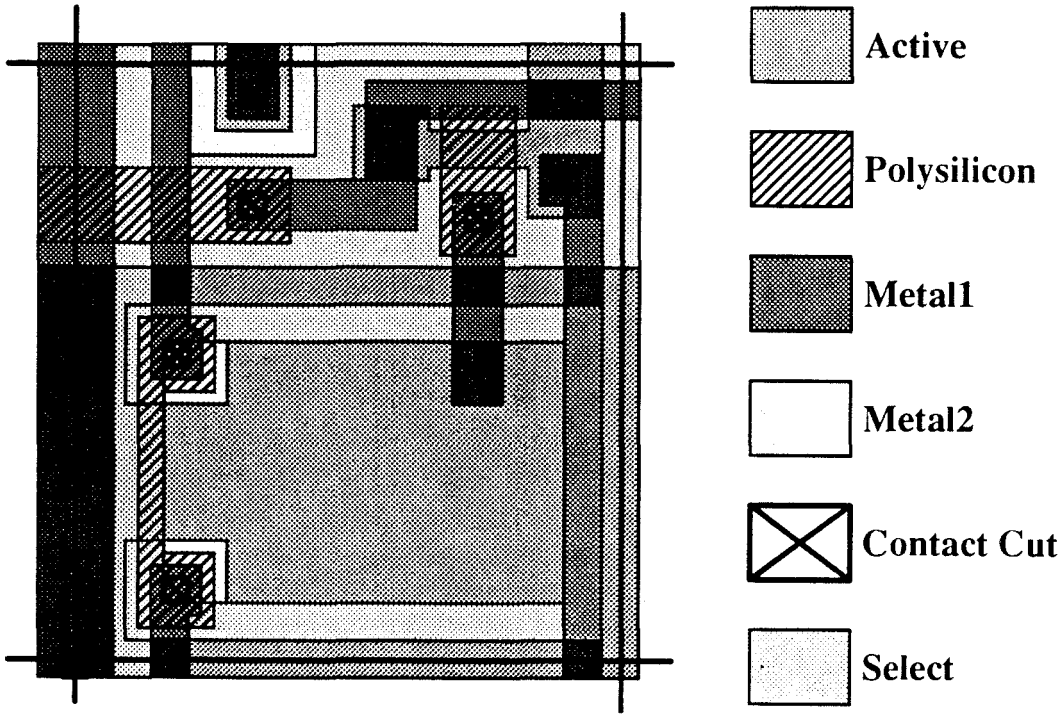



Fig. 4.5 - Layout of Synapse

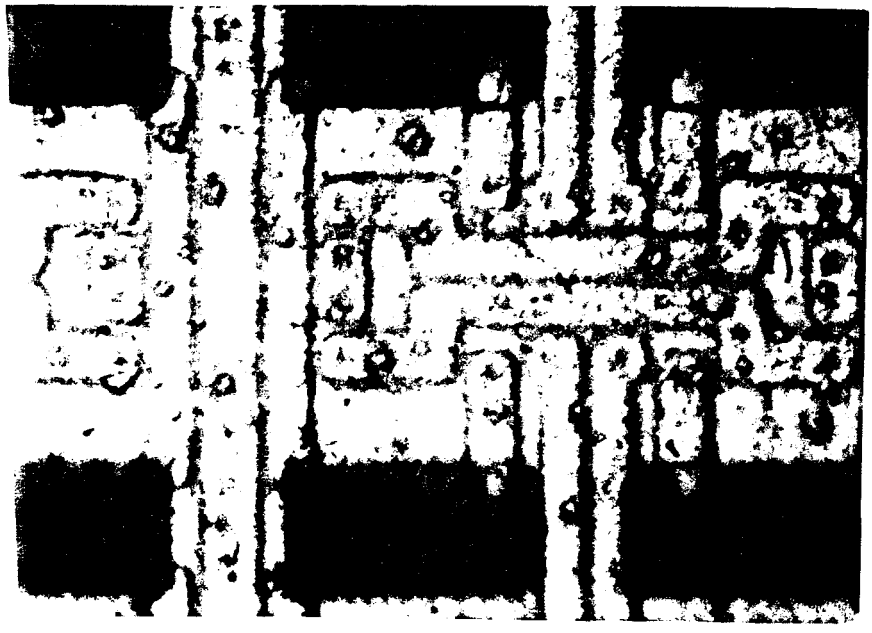


Fig. 4.6 - Photograph of Synapse

manual alignment of illumination patterns with photodetectors. An arbitrary compromise was chosen in the design in that the remaining circuitry was designed in as small a thin strip as possible, with the photodiode filling the rest of the synapse

out into a square shaped region. The resulting synapse was $43\frac{1}{2}$ $\mu m$ on a side with a total area of 1892 $\mu m^2$. The photodiode was roughly rectangular in shape with a maximum length of 32 $\mu m$ and width of 17 $\mu m$ and area of 474 $\mu m^2$ (part of the photodiode is obscured by the contacts). The synapse transistors ($3\times6$ $\mu m^2$) were slightly longer than minimum to increase their resistance because in steady-state, the chip has current flowing from $V_{dd}$ to $Gnd$ through the synapse transistors. On the other hand, the pulldown transistors ($8\times2$ $\mu m^2$) were made as wide as possible given space constraints in order to reduce the turn-on RC time-constant at the gate of the synaptic transistor.

We can operate the synapse in either an instantaneous or clocked mode. In the instantaneous mode, given the gate voltage of the pulldown transistor, the gate voltage of the synaptic transistor is determined by the instantaneous photocurrent. Actually, the synaptic gate voltage lags the photocurrent, and the photocurrent must remain constant for a given time before the neurons evaluate their inputs. The time constant of the lag is upper-bounded by the RC time constant with the maximum impedance of the pulldown transistor and the maximum capacitance of the gate of the synaptic transistor given the voltage swing corresponding to the change in photocurrent. An additional RC margin is required for the response of the synapse furthest from each neuron to reach the neuron.

In order to predict the electrical characteristics of the synapse, we will use the following simple models for the NFETs (Eq. 4.1), PFETs (Eq. 4.2), and photodiodes (Eq. 4.3) that make up the synapse circuit:

$$I_{dn} = \begin{cases} 0 & V_{gs} < V_{tn} \\ \kappa_n S(V_{gs} - V_{tn} - \frac{V_{ds}}{2})V_{ds} & V_{ds} < V_{gs} - V_{tn} \\ \kappa_n S\frac{(V_{gs}-V_{tn})^2}{2} & V_{ds} > V_{gs} - V_{tn} \end{cases} \tag{4.1}$$

$$I_{dp} = \begin{cases} 0 & V_{gs} > V_{tp} \\ -\kappa_p S(V_{gs} - V_{tp} - \frac{V_{ds}}{2})V_{ds} & V_{ds} > V_{gs} - V_{tp} \\ -\kappa_p S\frac{(V_{gs}-V_{tp})^2}{2} & V_{ds} < V_{gs} - V_{tp} \end{cases} \tag{4.2}$$

$$I = I_0(e^{qV/nkT} - 1) - I_p \tag{4.3}$$

Although an NFET would typically be used as a pulldown, we have chosen to use a PFET in order to reduce area loss to the spacing required between NFETs and PFETs or substrate photodiodes. Using Eqs. 4.2 and 4.3, we can estimate the voltage $V_g$ at the gate of the synaptic transistor as a function of the photocurrent $I_p$. Figure 4.7 shows the modeled dependence of synaptic transistor gate voltage $V_g$ on input photocurrent $I_p$. In steady-state, $I_p$ will be proportional to the light power incident on the photodiode.

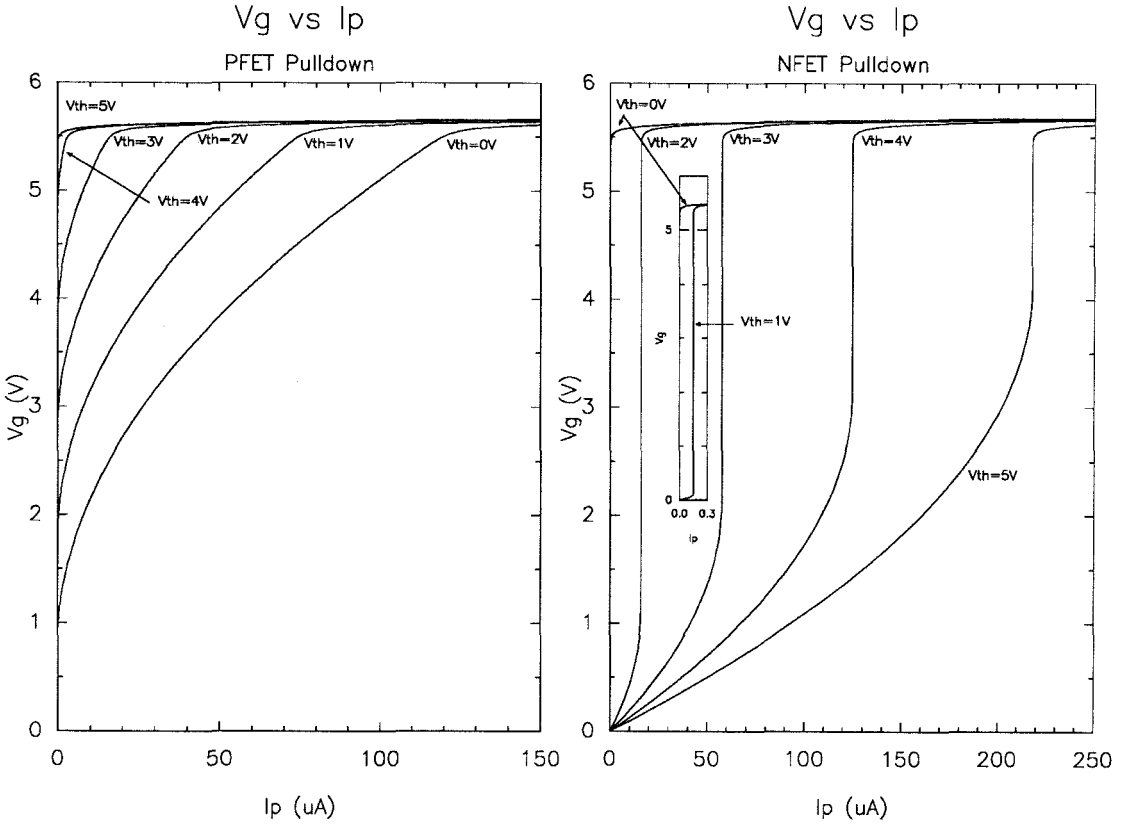



Fig. 4.7 - $V_g$ vs. $I_p$ for PFET and NFET pulldowns

We see that as long as the photodiode is reverse-biased, we can drop the first term of Eq. 4.3 and set the current through the diode equal to the photocurrent. Because the gate of the synaptic transistor acts essentially as a capacitor, no current flows through it in steady-state. Thus, the source voltage of the pulldown PFET

adjusts itself so that the current through the pulldown transistor equals the photocurrent. With no photocurrent, $V_g$ is free to float between $Gnd$ and $V_p - V_{tp}$, a threshold above the gate voltage $V_p$ of the pulldown transistor. With an applied photocurrent, the gate will charge-up until the pulldown transistor turns on at $V_g = V_p - V_{tp}$. Because the drain voltage of the pulldown PFET will always be below its gate voltage, the pulldown will always be in cutoff or saturation. Once the transistor turns on, the load behavior is determined by the V-I characteristic of the PFET in saturation and the synaptic gate voltage increases like the square root of the photocurrent. The pulldown gate voltage acts as a bias on the synaptic gate voltage. The synaptic gate voltage continues to increase with increasing photocurrent until the photodiode becomes forward biased. At this point, the forward diode current increases exponentially, essentially constraining the synaptic gate voltage to rise no higher than .6V above $V_{dd}$. We see that the PFET pulldown gives us better control of the gate voltage near $V_{dd}$ while the NFET gives better control near $Gnd$.

In the clocked mode, we apply a 0 $V$ pulse to the pulldown transistors with a pulsewidth sufficient to precharge the gate capacitors to their minimum value of $-V_{tp} \approx .8V$. After turning off the pulldown, we illuminate the photodiodes and integrate the resulting photocurrent on the gates of the synaptic transistors, raising the gate voltage towards a maximum value of $V_{dd} + .6V$. Although the maximum processing rate of the chip is lower for the clocked mode because of the precharge time, the synchronization between the source of the illumination pattern and the chip clock is not as critical, because the neurons may be operated at any time between the end of illumination and the beginning of the next pulse on the pulldown transistors.

It was thought that the synapse could be used to implement either binary digital or continuous analog weights. In either instantaneous or clocked modes, for binary digital weights we set the gate voltage of the synaptic transistor to one of

two voltages representing an *on* and an *off* connection; for analog weights we would set the voltage anywhere in between. Further analysis, however, shows that analog weights can only be implemented in one special case.

Whether neurons have a current or voltage input, the conductance of the synapse $g_i$ is proportional to the weight $w_i$ that it implements. For current inputs, we set the input voltage of the neuron to some fixed voltage $V_0$ and measure the resulting current (Eq. 4.4). For voltage inputs, we set the input current of the neuron to some $I_0$ (typically $I_0$=0) and compare the resulting voltage with a threshold $V_0$ (Eq. 4.5).

$$I = \sum_i g_i(V_i - V_0) \tag{4.4}$$

$$V = \frac{\sum_i g_i V_i - I_0}{\sum_i g_i} \tag{4.5}$$

With a hard threshold, however, the two cases, fixed current or fixed voltage, are equivalent because we can determine whether the voltage will be above or below the threshold by fixing the voltage to the threshold $V_0$ and measuring the current $I$. If $I > I_0$ then $V > V_0$ when $I = I_0$, while if $I < I_0$ then $V < V_0$ when $I = I_0$. With a soft threshold, a current input (fixed voltage) is more desirable since we can set the weights arbitrarily and rely on Kirchoff's Current Law to generate the desired weighted sum. With a voltage input (fixed current), the effective weight of a synapse $g_i$ will depend in a nonlinear fashion on both the input and output voltage of the synapse such that its weight relative to the other synapses will change depending on the input of the synapses and resulting weighted sum at their output. Thus, we will consider only the case for a neuron with a fixed voltage, current input noting that the results also apply for a fixed current, voltage input with hard-thresholding neurons.

With a fixed voltage $V_0$ at the input of the neuron, the weight implemented by the synapse is proportional to its conductance which depends now only on the synaptic gate voltage $V_g$ and the input voltage of the synapse $V_{in}$ (*i.e.*, the output voltage of the neuron at its input). Using Eq. 4.2, we can estimate the conductance $g$ of the synapse as a function of $V_g$ and $V_{in}$ for a fixed $V_0$ as follows:

$$g = \begin{cases} 0 & V_g \geq V_0 + V_{tp} \text{ and } V_g \geq V_{in} + V_{tp} \\ S\kappa_p \frac{(V_{in}+V_{tp}-V_g)^2}{2(V_{in}-V_0)} & V_g \geq V_0 + V_{tp} \text{ and } V_g \leq V_{in} + V_{tp} \\ S\kappa_p \frac{(V_0+V_{tp}-V_g)^2}{2(V_0-V_{in})} & V_g \geq V_{in} + V_{tp} \text{ and } V_g \leq V_0 + V_{tp} \\ S\kappa_p \left(V_{tp} - V_g + \frac{V_0+V_{in}}{2}\right) & V_g \leq V_{in} + V_{tp} \text{ and } V_g \leq V_0 + V_{tp} \end{cases} \tag{4.6}$$

Figure 4.8 plots the synapse conductance $g$ as a function of $V_g$ and $V_{in}$. In order to implement an analog connection, we must be able to control the conductance over a range of gate voltages $V_g$ independent of the synaptic input voltage $V_{in}$ for at least two values of the synaptic input voltage $V_{in}^{(1)}$ and $V_{in}^{(2)}$. Unfortunately, we find that the shape of the $g - V_g$ characteristic varies continuously as we change $V_{in}$. Thus, we generally cannot implement analog weights using a single MOSFET simply by adjusting its gate voltage with the following exception.

Note that in the special case of $V_{in} = V_0$, the source and drain of the FET are at the same voltage and no current flows through the device regardless of its $g - V_g$ characteristic. Thus it is possible to implement analog weights in the special case of unipolar binary neurons with $V_{in} = V_0$ representing a "0" output. Using the voltage $V_{in}$ representing a "1" output, we can choose a curve from Fig. 4.8 that allows us to predict the voltage $V_g$ required at the gate of the synaptic transistor to implement a given analog weight. Using Fig. 4.7, we can then convert this required voltage $V_g$ into a required photocurrent $I_p$.

It is also possible to implement binary connections when using our synapse with binary neurons. In this case, we need only match the conductance at two values of $V_g$. We can match the conductance at $V_g$=$V_{dd}$ because $g = 0$ independent of the input voltage $V_{in}$. We match the conductance at some other $V_g$ by adjusting

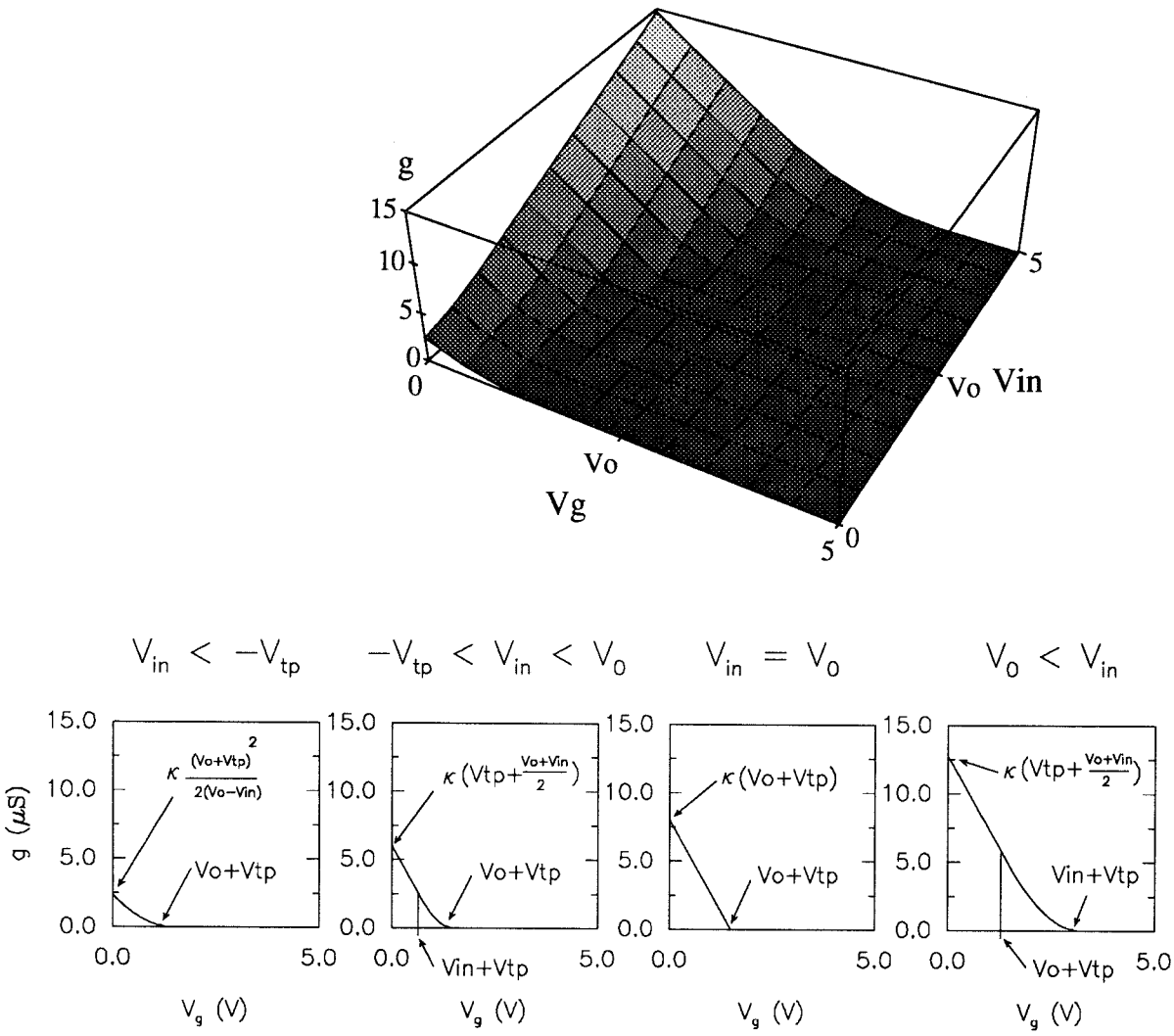



Fig. 4.8 - Synapse Conductance vs $V_g$ and $V_{in}$

the neuron input voltage (for current inputs) or the neuron threshold voltage (for voltage inputs) $V_0$ to equalize the conductance of the synapse when connected to either neuron input voltage.

Because of the design of the neuron (described in the next subsubsection), $V_{in}$=0$V$ or 5$V$ only. Unfortunately, because we made no provision for adjusting the threshold of the neuron away from the origin, it makes no sense to use unipolar analog connections with unipolar binary neurons. We therefore restrict ourselves to binary bipolar neurons and binary unipolar connections. Because we implement

only binary weights, we choose to perform all our experiments with clocked binary connections because of their simpler implementation.

#### 4.2.1.6 Neuron Design

The design of the neuron is also critical to the proper functionality of the chip. However, because the neurons occupy only the periphery of the array, minimization of their size is not a critical aspect of the design. The only constraint on their size is that each neuron have a maximum width equal to that of two synapses. Each neuron can occupy two synapse widths because every pair of synapse rows or columns can have one neuron on each side. Although the synapse design calls for neurons with current inputs, none of the designed neurons operated correctly when tested using the analog simulation tool, *anaLOG*, used at the California Institute of Technology. Instead, a last minute substitute was used to meet the MOSIS deadline. The actual neuron circuit used is pictured in Fig. 4.9. It takes a voltage input, applies a hard threshold, and generates a voltage output. Although such a neuron is not conducive for performing the summation of synapse outputs via KCL, it turns out that the neuron will operate properly for binary synapses with a zero threshold. With bipolar binary neurons and unipolar binary synapses, such a neuron should generate a HIGH output if it is connected to more neurons with HIGH outputs than LOW and vice versa. If it is connected to an equal number of neurons with HIGH and LOW outputs, the input voltage will be $V_{ref}$–the voltage generated by a voltage divider consisting of equally sized synaptic transistors between $V_{dd}$ and *Gnd* with $V_g$ corresponding to a 1 weight. Using Eq. 4.2, we find that the resulting threshold voltage should be about $V_{ref} = 3.3V$. When the input of a neuron is connected to more HIGH neurons than LOW, the input voltage will be above $V_{ref}$ and vice versa. Thus, by setting $V_{th} = V_{ref}$, the neuron will implement a zero threshold. Using the voltage divider view of the crossbar, we see that the crossbar

implements a threshold at a given ratio of connections to neurons with HIGH and LOW outputs–the zero threshold corresponding to a 1:1 ratio. We see that by adjusting $V_{th}$, we can choose different ratios, but not different thresholds, at which to switch the neuron. Because of the neuron design then, we are limited to binary neurons, binary connections, and (essentially) switching at a zero threshold.

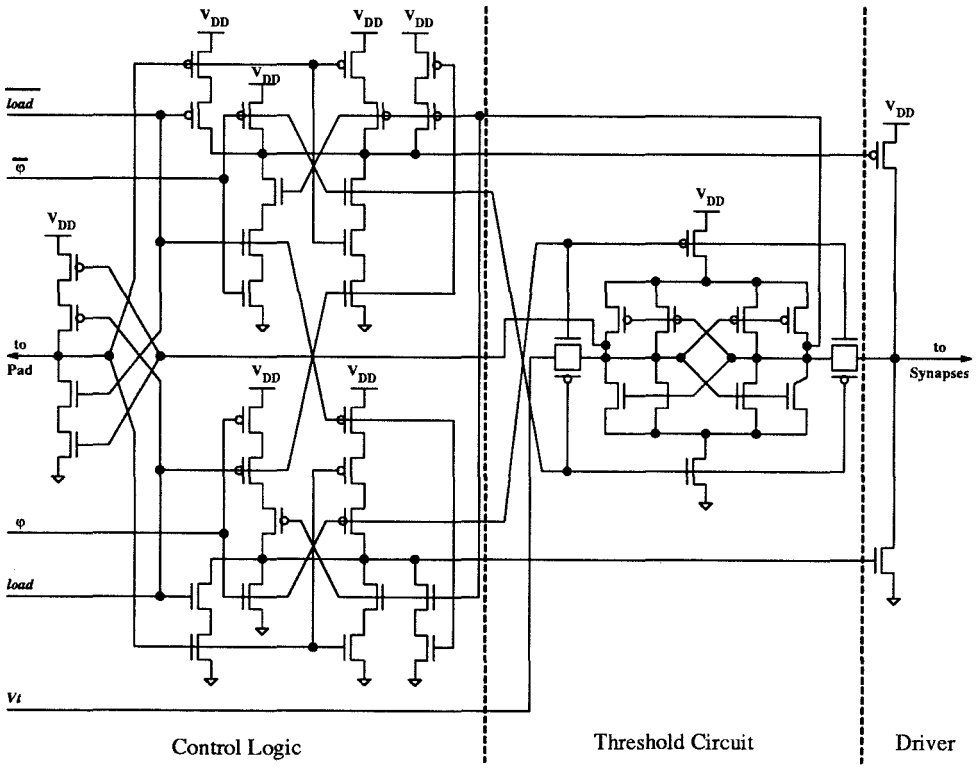



Fig. 4.9 - Neuron Circuit Diagram

#### 4.2.1.7 Pads

There are 40 pads on the ONNC. Four of these pads are for power and ground. Eight of these are *barepads* connected directly into the chip. The other 28 pads are *inpads*, input pads with protection circuitry. Each of these inpads were converted to one of three pads shown in Fig. 4.10: input/output, input w/complement, or threshold voltage. The input/output circuitry consisted of a three stage gated buffer to drive an off-chip output, and an inverter to buffer the input. The input with complement circuitry passed the input and generated an inverted version of

the input for internal use. The threshold voltage circuit used a voltage divider consisting of a pair of PFETs whose gates were biased by a PFET pullup to generate the reference voltage, $V_{ref}$. This voltage could be viewed externally or set by driving the pad with an external voltage.

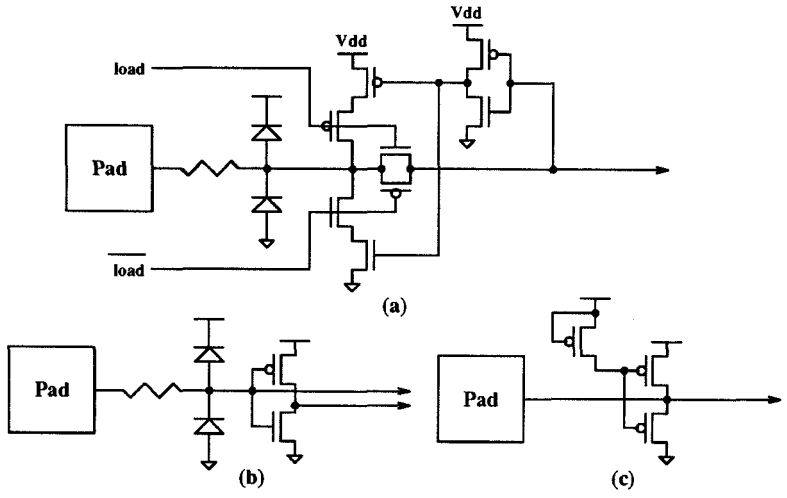



Fig. 4.10 - (a) Input/Output, (b) Input with Complement, and (c) Threshold Voltage Pads

#### 4.2.1.8 The ONNC

Figure 4.11 shows the ONNC as fabricated on a MOSIS TinyChip. The TinyChip measures $1.4\times1.4mm^2$. There is a $15\times15$ array of synapses, $.645\times.645mm^2$ in size, at the center of the chip. Because of the nature of the padframe as described below, direct connections could only be provided for 22 neurons. Thus, 22 neurons are placed around the synaptic array. The neuron and synapse circuitry combined occupy approximately $.8\times.8mm^2$ in the center of the chip.

The TinyChip padframe that we used had 40 pads (Fig. 4.12), 12 of them fixed. Four of the fixed pads provided power and ground connections. Eight of the pads provided direct, unprotected connections to the inside of the chip, and as such, could not be connected directly to MOSFET gates at the neuron inputs. These eight pads connected to 4 rows and 4 columns of neurons and were used for testing

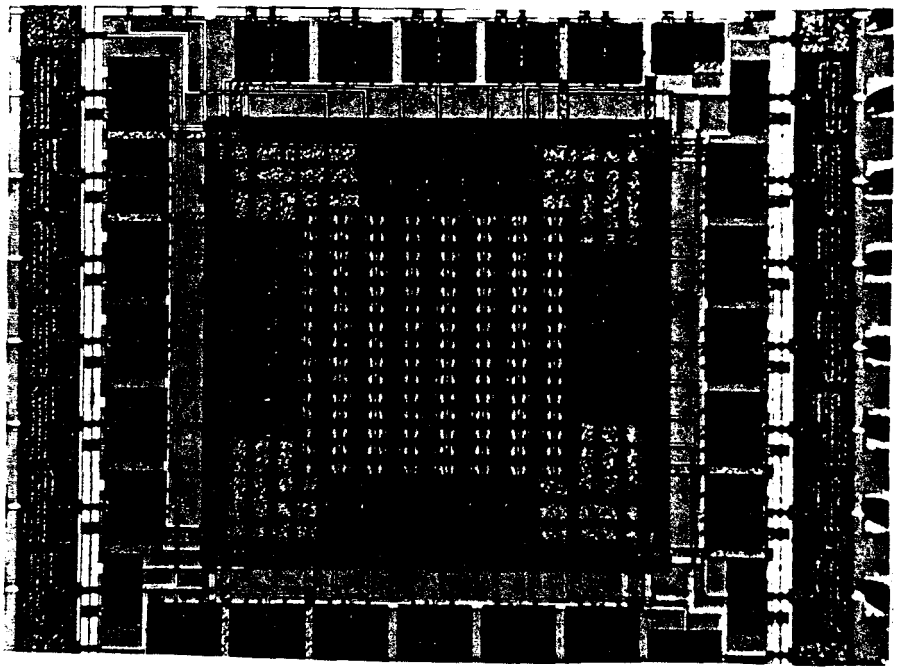


Fig. 4.11 - The ONNC

of the chip. The remaining 28 pads are configured as inputs. Six of these are used for control and clocking signals. The remaining 22 connect to neurons grouped into two layers of 11 neurons each.

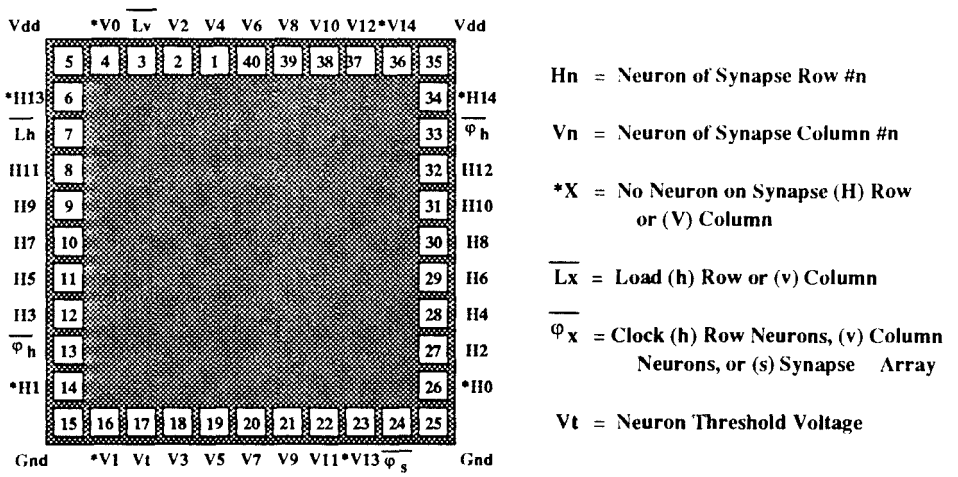



Fig. 4.12 - Pinout of ONNC

### 4.2.2 Characterization of the ONNC

Although we may perform a number of experiments to characterize the chip as completely as possible, we report only those experiments that measure characteristics important to the clocked unipolar binary operation of the synapse.

#### 4.2.2.1 Synapse Switching Time

In the first experiment, we measure the synapse switching time as follows. The ONNC, has 4 rows and columns of test synapses. We disable the neuron outputs and connect to $Gnd$ a line connecting a column of the test synapses. We also connect a line connecting a row of test synapses, to a current sense amplifier that maintains the row voltage at $V_{dd}$ as shown in Fig. 4.13b. The voltage across the sense amplifier measures the net current flowing through the 15 synapses in that row to/from the different columns. Because all but one of those columns is floating, the measured current corresponds to that flowing through the synapse at the intersection between the chosen row and column.

We begin by electrically precharging the synapse into its "on" state. We then uniformly illuminate the chip using a collimated beam from a 633 $nm$ HeNe laser that passes through an electronically controlled shutter as shown in Fig. 4.13a. Fig. 4.14 shows the typical output of the sense-amplifier during one of the experiments. The upper flat region in each trace represents $0V$ across the resistor and thus zero current. We measured the time required to switch the synapse from its initial low impedance state to its high impedance state as shown in Table 4.1. For intensities in the 10s of $\mu W/cm^2$, we found switching times on the order of $ms$.

Table 4.1 - Switching Time and Energy

| Incident Intensity ($\mu W/cm^2$) | Switching Time ($ms$) | Switching Energy ($pJ$) |
|---|---|---|
| 7.9 | 27 | 1.01 |
| 15.3 | 8.5 | .62 |
| 16.6 | 7.0 | .55 |
| 33.7 | 4.5 | .72 |

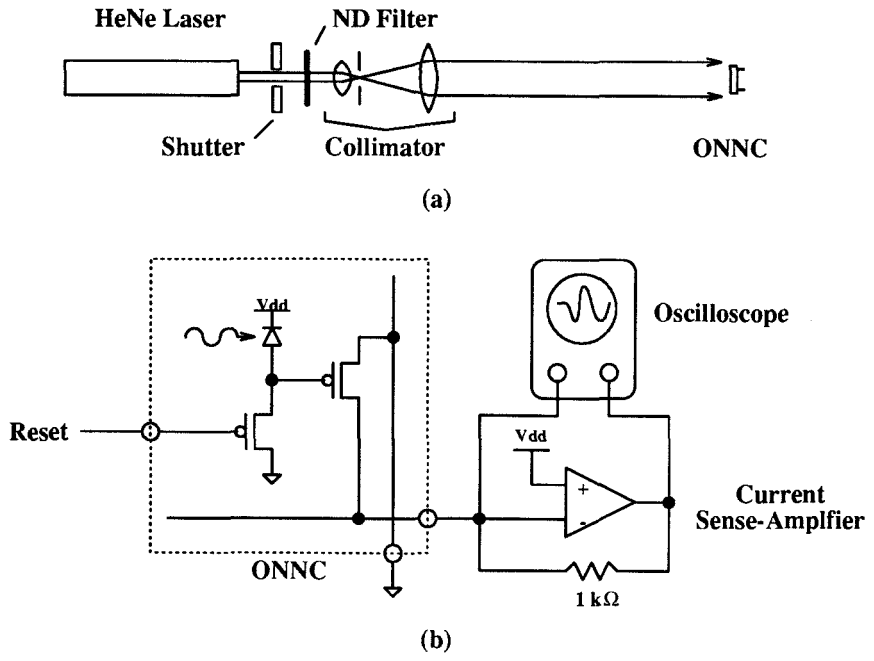



Fig. 4.13 - (a) Optical and (b) Electronic Setup
to Measure Synapse Switching Time

#### 4.2.2.2 Synapse Switching Energy

The optical energy $\epsilon$ required to switch the synapse is given by the intensity $\Im$ of the incident beam times the area $A_p$ of the photodiode times the switching time $\tau$, and is also calculated in Table 4.1. We expect the switching time to be inversely proportional to intensity as follows:

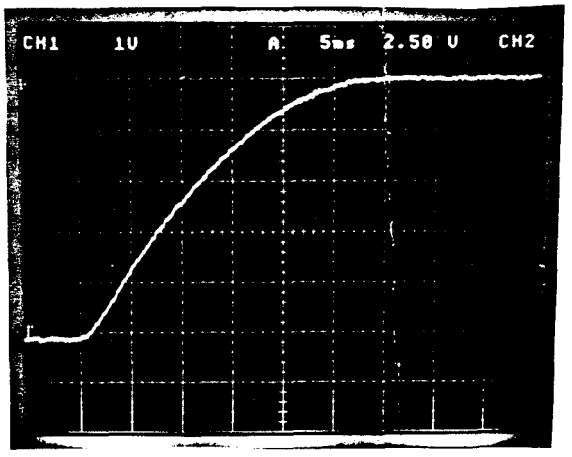
$$\epsilon = \tau \Im A_p \tag{4.7}$$

$$\tau = \frac{\epsilon}{\Im A_p} \tag{4.8}$$

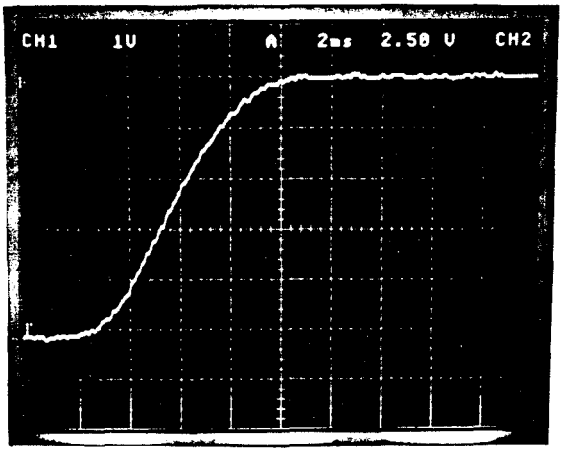
Figure 4.15 shows switching time plotted against intensity for the data in Table 4.1. The curve in the figure represents a switching energy of .865 $pJ$ calculated through a least squares fit of the data.

#### 4.2.2.3 Detector Efficiency

We can also calculate the quantum efficiency of our detector as follows. The quantum efficiency $\eta$ equals the number of electrons $n_e$ divided by the number of




$7.9\ \mu W/cm^2$ $27\ ms$

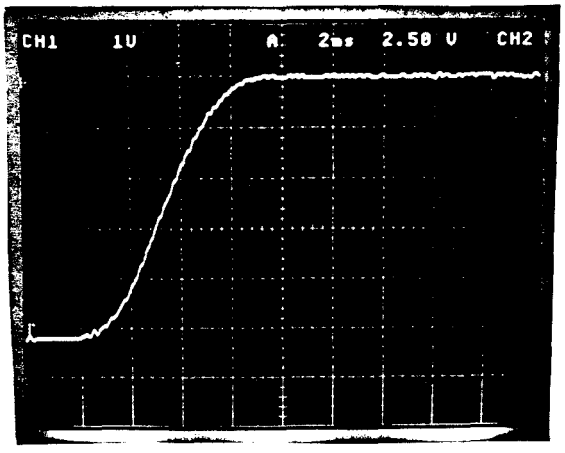



$15.3 \mu W/cm^2$ $8.5\ ms$

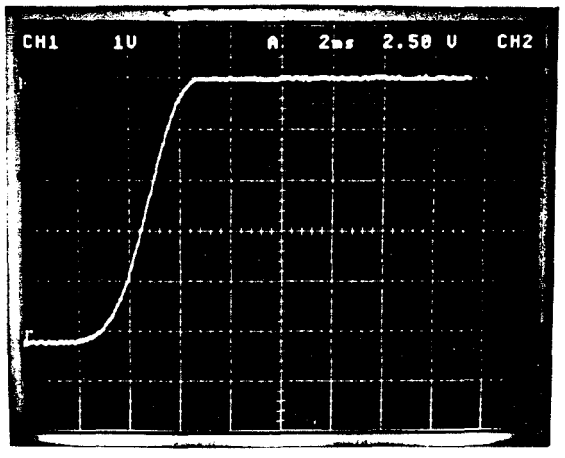
Fig. 4.14a - Synapse Switching Times

photons $n_p$. The number of electrons equals the change in charge $Q$ at the node connected to the gate of the synaptic transistor, divided by the charge $q$ of a single electron. The number of photons equals the switching energy divided by the energy per photon $hc/\lambda$.

$$\eta = n_e/n_p \tag{4.9}$$




16.6 $\mu W/cm^2$ 7 $ms$




33.7$\mu W/cm^2$ 4.5 $ms$

Fig. 4.14b - Synapse Switching Times

$$= \frac{Q/q}{\epsilon/\frac{hc}{\lambda}} \tag{4.10}$$

$$Q = C(V_f)V_f - C(V_i)V_i \tag{4.11}$$

$$V_i = -V_{tp} \tag{4.12}$$

$$V_f = V_{dd} + V_d \tag{4.13}$$

We calculate the capacitance of the node at the gate of the synaptic transistor

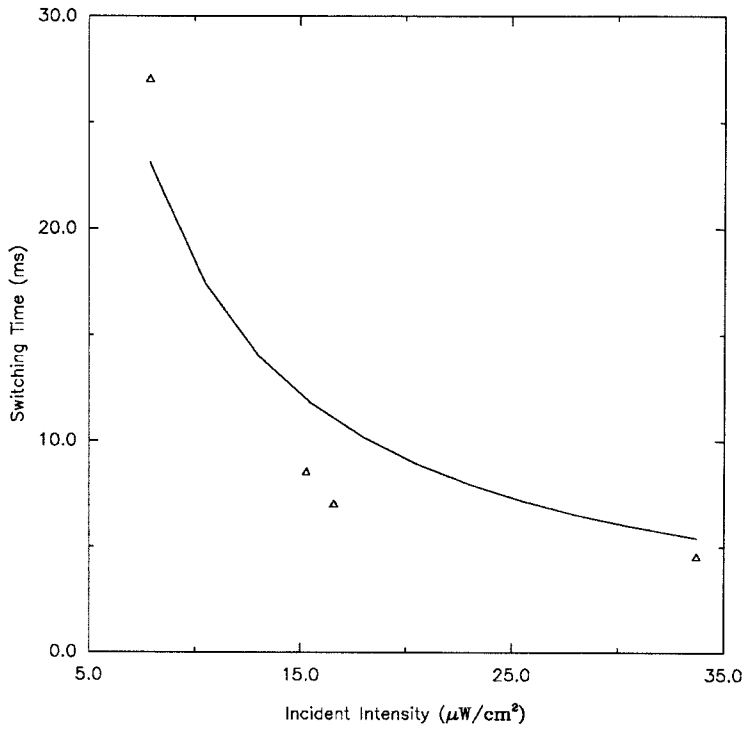



Fig. 4.15 - Synapse Switching Time vs. Incident Intensity

using the area and circumference measurements extracted from the layout of Fig. 4.5 and the MOSIS measured capacitances, both listed in Table 4.2. We estimate the gate capacitance using the following equation with $\epsilon_{SiO_2}$=3.9$\epsilon_0$ and $t_{ox}$ =370 $\mathring{A}$:

$$C_g \approx \frac{\epsilon_{SiO_2}}{t_{ox}} = .933fF/\mu m^2 \tag{4.14}$$

Using the following parameters, $V_{tp} = -.801V$, $V_d = .6V$, $V_{dd} = 5V$, $\lambda = 633nm$, and $C \approx 162fF$ we find the quantum efficiency to be 124%. Because we do not expect any avalanche-type process, we do not expect the quantum efficiency to exceed 100%. However silicon photodiodes have a quantum efficiency above 90% in the red, and the difference could easily be accounted for by a slight underestimate of the switching energy and/or an overestimate of the capacitance.

Table 4.2 - Synaptic Node Capacitance Parameters

| Type | Capacitance ($fF/\mu m^2$) | Area ($\mu m^2$) |
|---|---|---|
| Poly-Subs | .071 | 54 |
| Pdiff-Subs | .216 | 494 |
| Metal1-Subs | .032 | 28 |
| Metal1-Poly | .043 | 6 |
| Metal2-Poly | .022 | 46 |
| Metal2-Metal1 | .037 | 46 |
| Gate | .933 | 25 |
| Pdiff-Subs (Fringe) | .263 ($fF/\mu m$) | 91 ($\mu m$) |

#### 4.2.2.4 Synapse Reset Time

In the next experiment, we measure the time required to precharge or reset the synapse. We do this by using the same electrical setup as in the previous experiment (Fig. 4.13) and measuring the time between the assertion of the reset line $V_p$ and the cutoff of current flowing through the synapse. Figure 4.16 shows a typical pair of traces measured on the oscilloscope with one channel measuring $V_p$ and the other the current through the synapse. The figure shows that the precharge time is on the order of $\mu s$. We find that although there is ringing on the response, the current settles to zero in about 4 $\mu s$.

#### 4.2.2.5 Synapse Current

With all synapses on, we measured the amount of current required to maintain the neuron input at a specified voltage. For a synapse connected to a neuron with a +1 (5 $V$) output, the input of the synapse corresponds to the source and the output to the drain of the synaptic transistor. As we sweep the output voltage $V_o$, the synaptic transistor will be in the linear regime for $V_o > V_{dd} - V_t$ and in saturation for $V_o < V_{dd} - V_t$. For a synapse connected to a neuron with a -1 (0 $V$) output, the input of the synapse corresponds to the drain and the output to the source of the synaptic transistor. In this case, as we sweep $V_o$, the synaptic transistor will

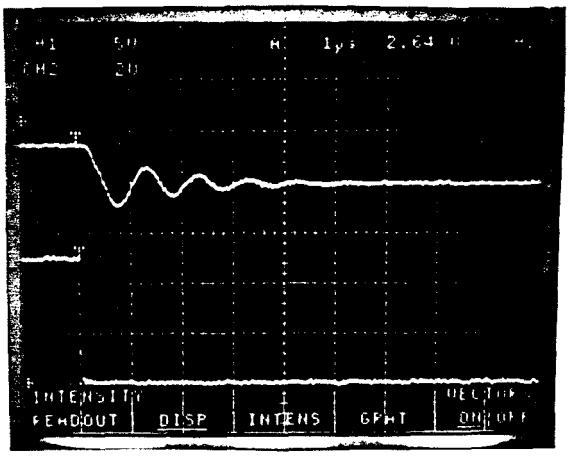



Fig. 4.16 - Synapse Reset Time

be in the linear regime for $V_o > V_{dd} - V_t$ and off for $V_o < V_{dd} - V_t$. We thus find the following equations for the synaptic current as a function of the synapse output voltage.

$$I_s(+1) = \begin{cases} \frac{\kappa_p}{2}(V_{dd} + 2V_{tp})^2 & V_0 < -2V_{tp} \\ \frac{\kappa_p}{2}\left[(V_{dd} + 2V_{tp})^2 - (V_o + 2V_{tp})^2\right] & V_0 > -2V_{tp} \end{cases} \tag{4.15}$$

$$I_s(-1) = \begin{cases} 0 & V_0 < -2V_{tp} \\ -\frac{\kappa_p}{2}(V_o + 2V_{tp})^2 & V_0 > -2V_{tp} \end{cases} \tag{4.16}$$

Notice that the forms of the two functions are identical with only a bias equal to $\kappa(V_{dd} + 2V_{tp})^2/2$ separating them. With all 15 neuron outputs at +1 or -1, the current will simply be 15 times the current through a single synapse. As we adjust the number of neuron outputs at $\pm 1$, the output current will be of the same form with a bias that varies linearly with the number of neuron outputs at +1. Fig. 4.17 shows the expected output current and Fig. 4.18 shows the measured output currents for 0, 7, 8, and 15 +1 neuron outputs. The measured curves appear similar in many respects to the expected curves; however, the measured curves are essentially flat over a smaller voltage range than expected when more neurons have

+1 outputs. Furthermore, the magnitude of the current in each direction is not equal. Nevertheless, we see that the magnitude of the synapse current is on the order of 20 $\mu A$.

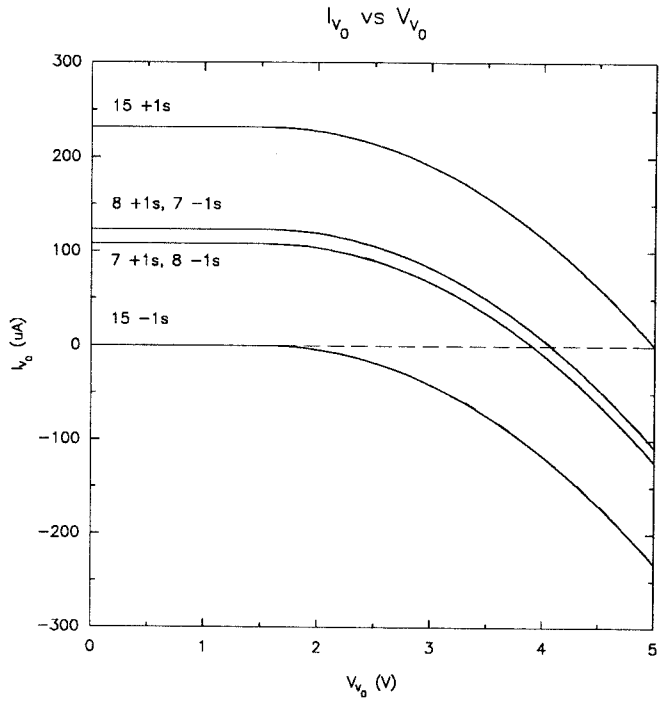



Fig. 4.17 - Expected Synapse Current vs $V_0$

By taking the curves for an opposite number of on and off neurons (15 +1s vs 15 -1s for example) we can estimate the neuron switching voltage $V_{ref}$ as follows. The neuron should switch at the neuron input voltage where the current through a group of synapses connected to $V_{dd}$ exactly matches the current through a like number of synapses connected to *Gnd*. The solid line in Fig. 4.18 represents the current through 15 synapses to +1 ($V_{dd}$) while the dashed line represents the current through 15 synapses to -1 (*Gnd*). The neuron switching voltage should be where the magnitude of the two currents are equal (intersection of solid and dotted lines) yielding a measured switching voltage of $3.93V$. This compares to the $3.3V$

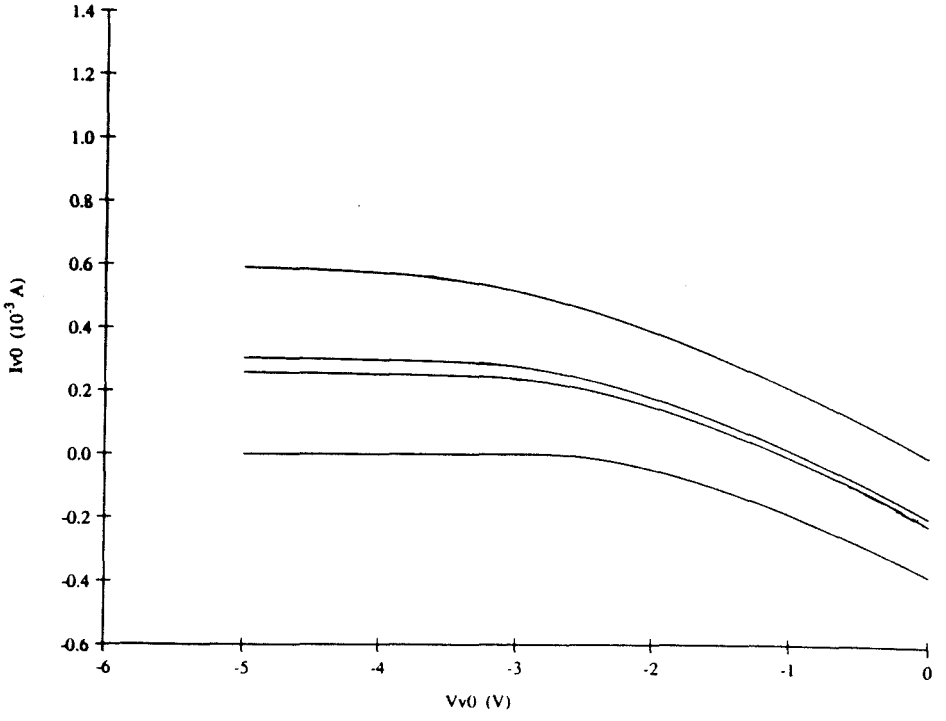



Fig. 4.18 - Measured Synapse Current vs $V_0$

switching voltage we estimated in Subsubsec. 4.2.1.6.

#### 4.2.2.6 Synapse Leakage Current

We also measured the amount of leakage current passing through the synapse with all synapses off and all neurons with either +1 or all -1 outputs. Figure 4.20 shows the measured synapse current as we sweep the synapse output voltage from 0 to $5V$. The leakage current was found to be in the $pA$ regime indicating that for arrays up to well over 100000 neurons per layer, leakage current is not a limiting factor.

#### 4.2.2.7 Synapse Uniformity

To measure the electrical uniformity of the array, we recorded the neuron input

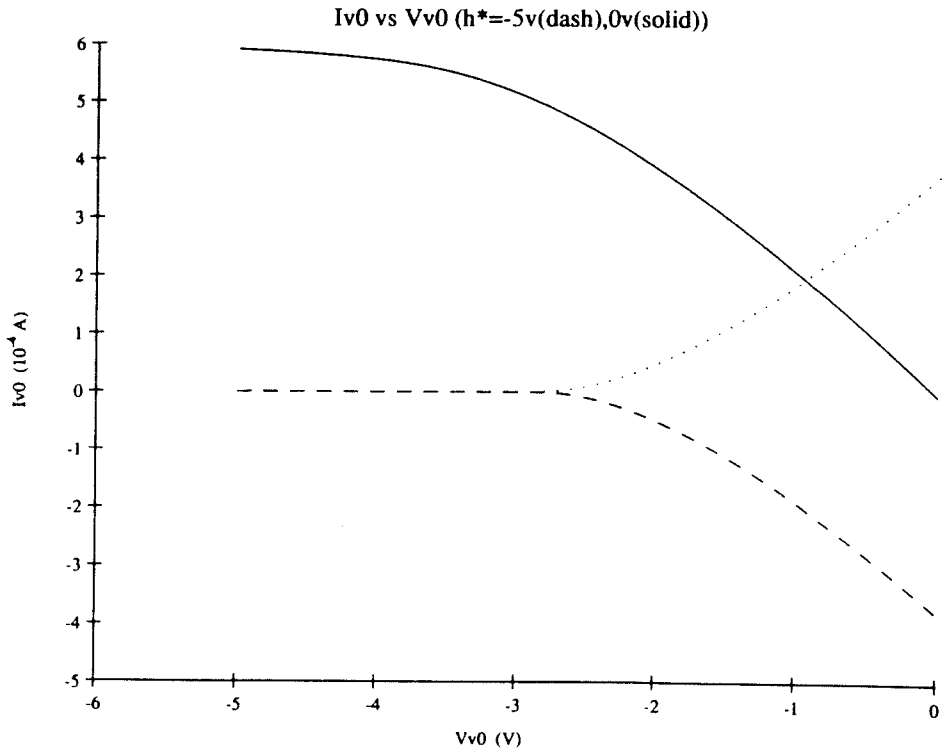



Fig. 4.19 - Measurement of $V_{ref}$

voltage as a function of the position of a single +1 neuron output in a field of -1s (Fig. 4.21), and a single -1 in a field of +1s (Fig. 4.22). Although the neuron input voltage should be independent of the position of the +1 and -1 inputs, we found a roughly 25 $mV$ variation in the neuron input voltage as we changed the position of a single +1 or -1.

Looking at Fig. 4.22, the lower voltage for -1s at positions 0, 1, 13, and 14 can be explained by the fact that these rows of synapses were driven by discrete devices off-chip instead of the neurons on-chip as shown in Fig. 4.23. These devices have a lower output impedance or greater drive capability, thus lowering the resulting neuron input voltage. The alternating pattern with even-numbered positions yielding lower neuron input voltages than odd-numbered positions is due to parasitic resistance

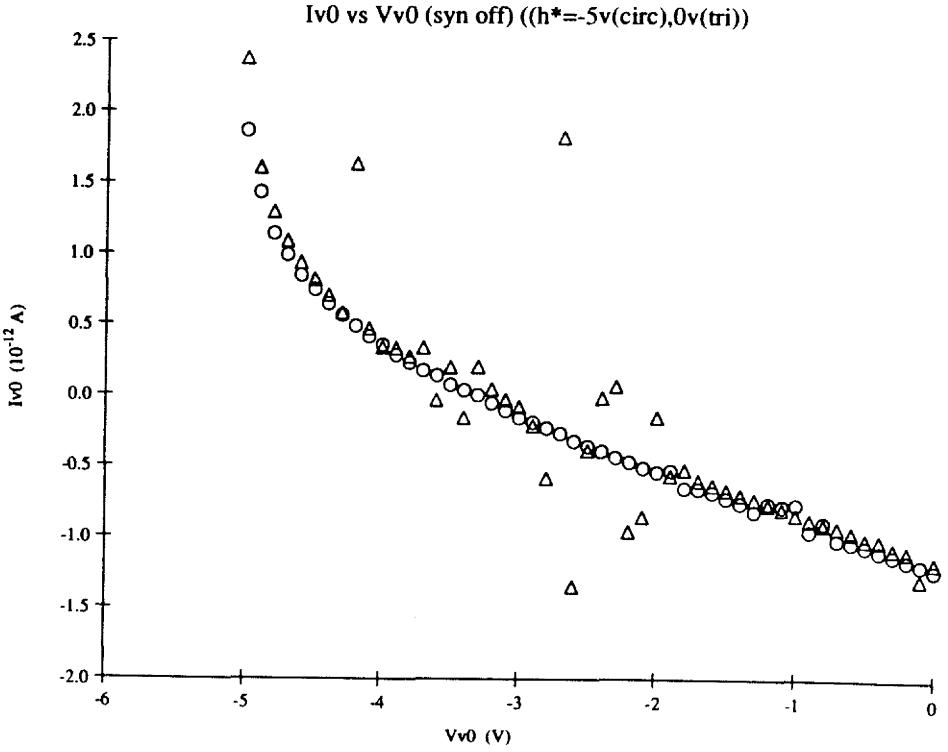



Fig. 4.20 - Synapse Leakage Current

and the layout of the network (Fig. 4.23). Figs. 4.21 and 4.22 measure the input voltage for a neuron at the edge of the array. Because we sacrifice a layer of metal for shielding, we are forced in places to run signals in polysilicon or diffusion. This means that there is less impedance between neurons that are close together and more impedance between neurons that are further apart. Because alternate neurons are placed on opposite sides of the synaptic array, the neuron input voltage bounces up and down as we shift the position of the single -1 input. Fig. 4.21 shows this parasitic resistance again in that not only does the neuron input voltage bounce up and down as we shift position due to the placement of alternate neurons on opposite sides of the array, but the input voltage generally decreases as we increase the distance between the neurons. It is unclear why this second manifestation of

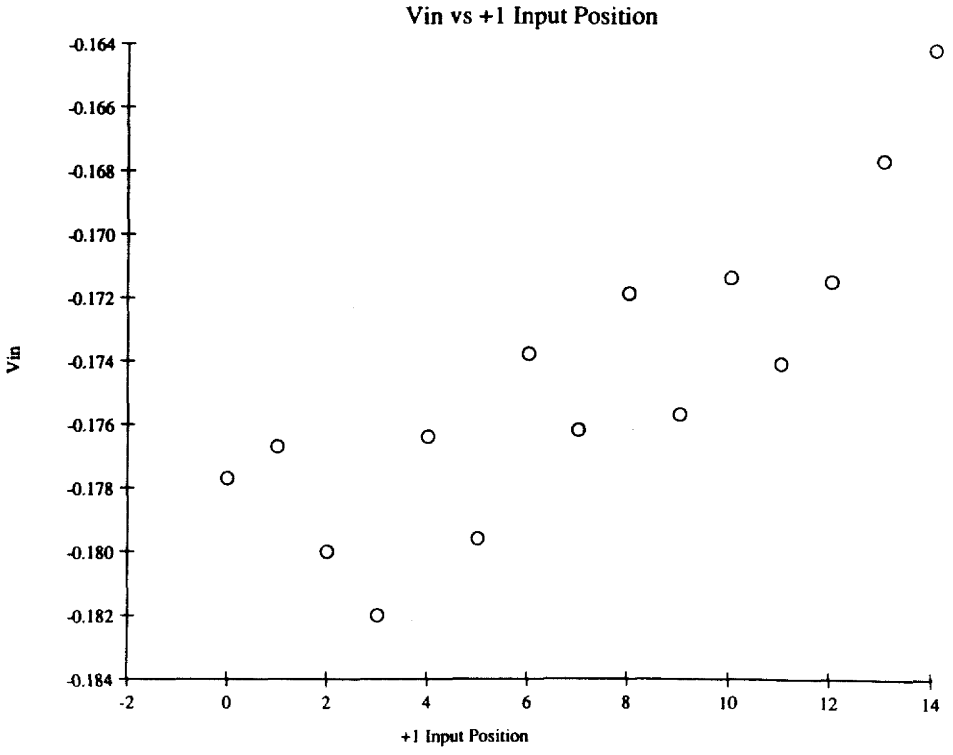



Fig. 4.21 - Neuron Input Voltage vs +1 Input Position

parasitic resistance is absent in Fig. 4.22. The effect of parasitic resistance is to introduce an uncertainty of about 25 $mV$ in the input voltage of a neuron. We found that the effect of other electrical nonuniformitities in the synapse were small compared to effect of the parasitic resistance.

#### 4.2.2.8 Neuron Input Voltage

In this experiment, we measure the input voltage seen by a neuron given different numbers of connections to high and low neuron outputs. Because we do not normally have access to this point in the circuit, we use a row of synapses not connected to a neuron to make this measurement. Figure 4.24 shows the optical setup for the experiment. A shuttered laser beam is collimated before it illuminates

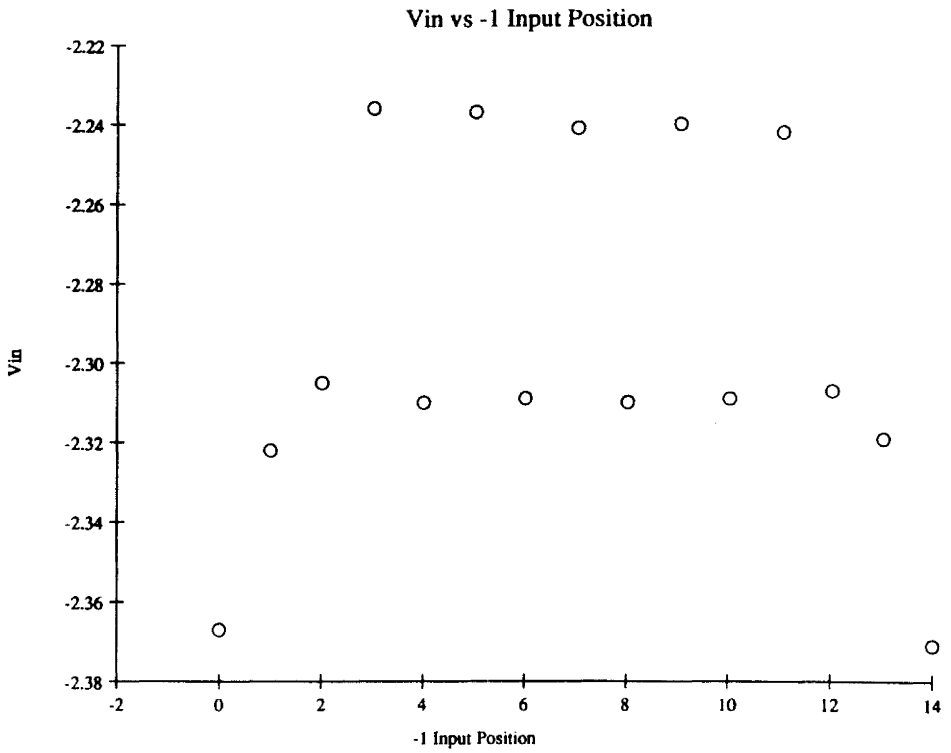



Fig. 4.22 - Neuron Input Voltage vs -1 Input Position

an aluminum foil mask (Fig. 4.25) which is then imaged with 50:1 demagnification through a beamsplitter onto the chip. The light reflected from the chip is then reimaged with 50:1 magnification (in effect, through a microscope) onto a CCD which is used to align the imaged pattern and the chip as shown in Fig. 4.26. The rectangles are the photodiodes and the bright circles are the imaged pinholes in the foil mask. By changing the foil mask, we can change the connection pattern implemented by the chip. We control the inputs to each synapse in the row by either loading the associated neurons with desired values or direct manipulation of inputs not connected to neurons. We use an oscilloscope to measure the resulting neuron input voltage.

Figure 4.27 shows the neuron input voltage versus the ratio of the number of

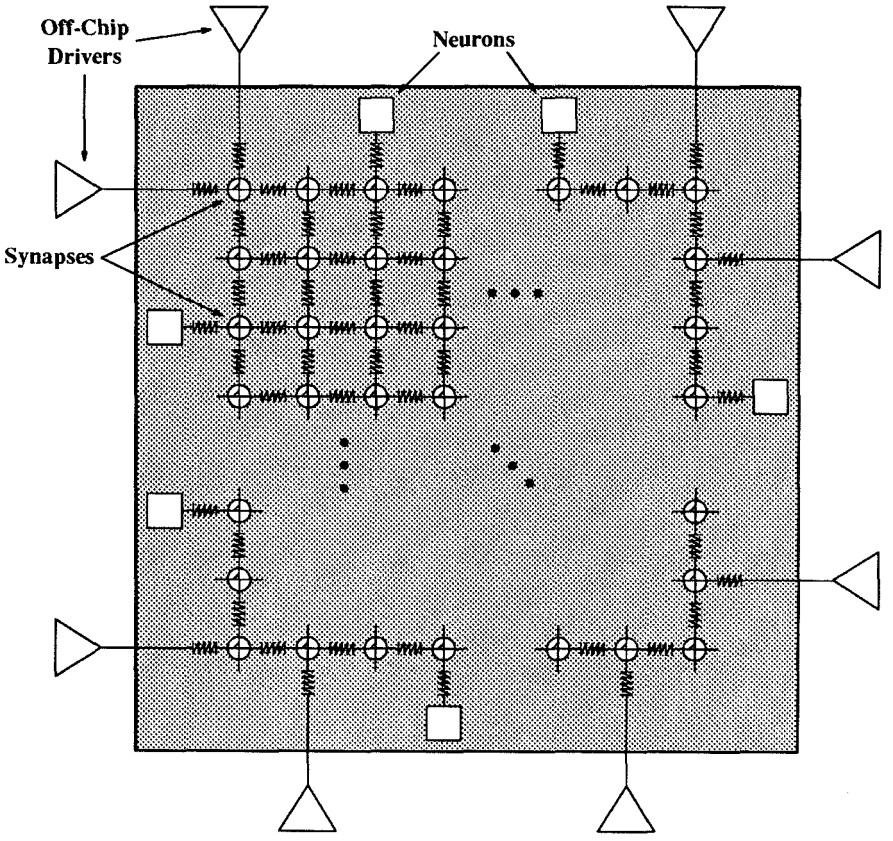



Fig. 4.23 - Parasitic Resistance in the ONNC

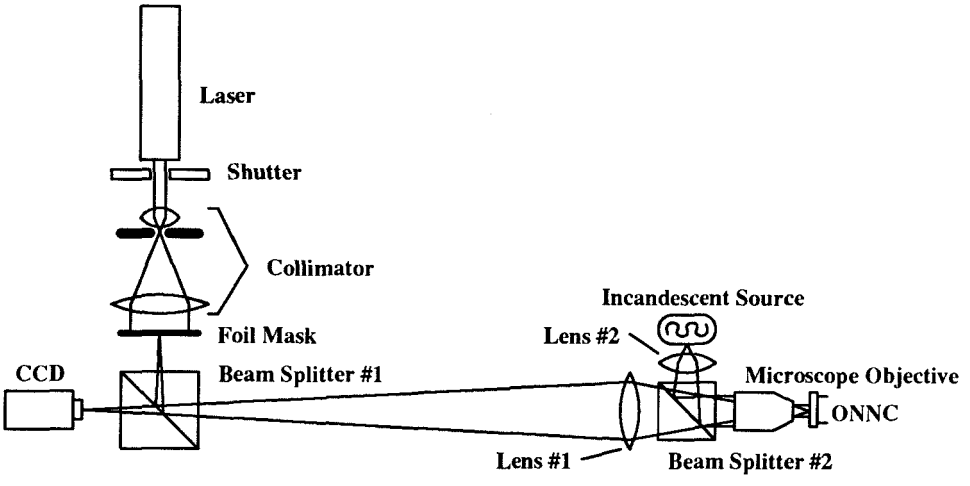



Fig. 4.24 - Foil Mask-ONNC Setup

connections to high and low neuron outputs. Remembering that the voltage actually depends on the ratio of high connections to low, we expect the curves to intersect only at the point where the number of high and low connections are equal

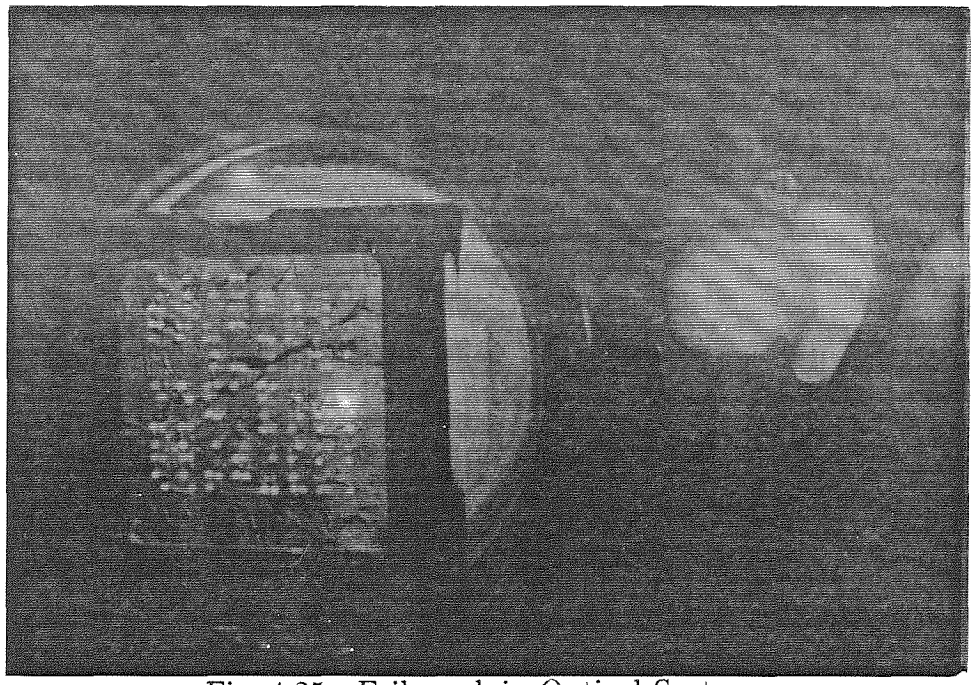


Fig. 4.25 - Foil mask in Optical System

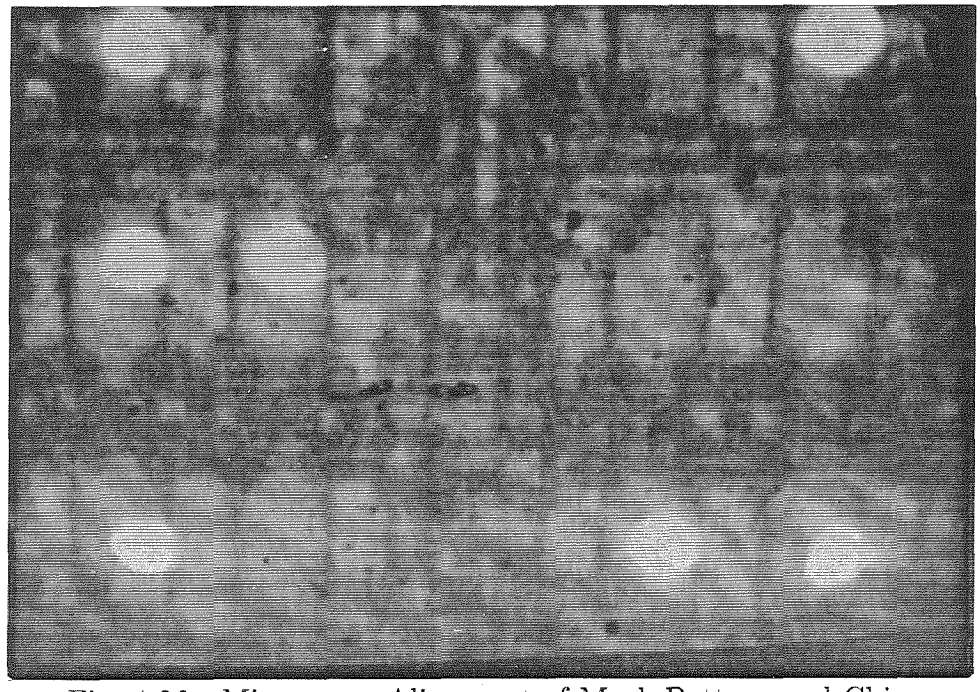


Fig. 4.26 - Microscope Alignment of Mask Pattern and Chip

which corresponds to a 1:1 ratio (normalized input equals 0). However, we notice that the measured neuron input voltage at this point depends on the total number of connections although it should be independent of this number. A leakage path from the neuron input to $V_{dd}$ could explain such behavior. The more connections

to $V_{dd}$ and *Gnd*, the higher the conductance. The higher the conductance, the less effect a leakage path will have on the input voltage. We suspect a leakage path from the neuron input to $V_{dd}$ since the neuron input voltage is drawn closer to $V_{dd}$ than expected with greater effect the smaller the total number of connections. We can estimate the magnitude of the impedance of the leakage path by using Eq. 4.2 to calculate the expected impedance of each connection. We use the following parameters $V_g = -V_{tp} \approx$ -.77$V$ and $\kappa_p \approx 5.16\mu A/V^2$ with $V_s = 5V$ and $V_d \approx V_0 \approx 3.3V$ for a connection to a high neuron output, and $V_s \approx V_0 \approx 3.3V$ and $V_d = 0V$ for a connection to a low neuron output. From Eq. 4.2, we find that connections to high neuron outputs are not saturated while connections to low neuron outputs are saturated. Using the above parameters in Eq. 4.2 and assuming an equal number of connections to $V_{dd}$ and *Gnd*, we find that connections to high neuron outputs have an impedance of about 74 $k\Omega$ while connections to low neuron outputs have an impedance of about 107 $k\Omega$. We are thus able to rule out the $10M\Omega$ oscilloscope probe as the source of leakage. Although we are unsure of the leakage path, we know that it has an effective impedance in the tens of kiloohms range.

#### 4.2.2.9 Neuron Switching Time

In the next experiment, we measured the switching time of the neuron. Figure 4.28 shows an example where we clock the neuron (exponential decay) and observe the output of the neuron through an external HCMOS buffer (sharp transition). Apparently, the capacitive load of the clock line was large compared to the drive capability of the external clock. Nevertheless, if we take the beginning of the decay as the beginning of the clocking signal, we find the neuron switching time to be on the order of 200 $\mu s$ or less. Theoretically, using a bistable comparator, it is possible for the neuron to become metastable with an arbitrarily long switching time; however such a situation was never observed.

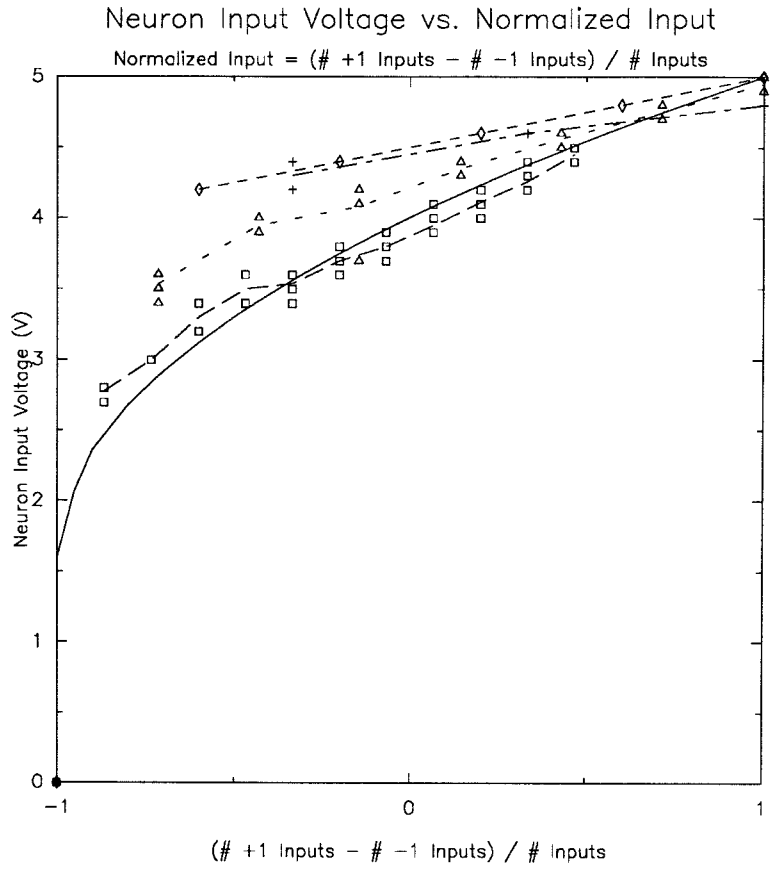



Fig. 4.27 - Neuron Input Voltage

#### 4.2.2.10 Neuron Tests

In the next experiment, we test the function of the neurons by precharging all the connections on and measuring the output of the neuron versus the threshold voltage $V_t$ and the ratio of connections to neurons with high outputs and low. First, we allow the voltage divider on-chip to set $V_t = V_0$ which we measure to be $3.9V$. Next, we turn various numbers of input neurons high and low. We find that all the neurons generate a low output with 7 or fewer outputs low and a high output with 8 or more neurons high. We then drive $V_t$ externally and find that we can adjust it so that neurons switch at any desired ratio of connections to neurons with high and low

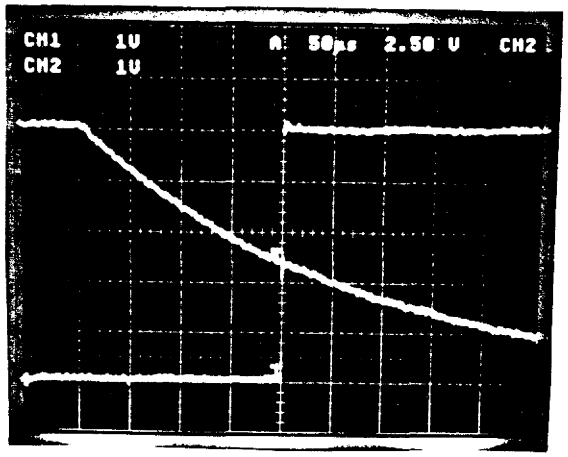



Fig. 4.28 - Neuron Switching Time

inputs. We also notice that as the threshold voltage approaches a voltage generated by a given ratio of connections to high and low neuron outputs, some neurons will generate an output high while others generate an output low given the same set of inputs. Moreover, we find that as the threshold voltage passes through one of these "boundary" voltages, there will be a pattern to the neurons that generate high and low outputs. This pattern can be explained by the parasitic resistance due to the higher resistance material (polysilicon, diffusion, and contact cuts) used to connect the synapses. As discussed earlier, this parasitic resistance results in every other neuron being connected more strongly together and a variation of about 25 $mV$ in the neuron input voltage. Once again, variations in the electrical characteristics of the neurons were small compared to the effects of parasitic resistance on the chip.

#### 4.2.2.11 Testing the Entire Chip

In the final experiment, we test the accuracy of the ONNC. There are many potential sources for error in such a system, some of which we list below.

**Transistor Nonuniformity** - Results in nonuniform neuron switching voltages

and nonuniform connection strengths.

**Parasitic Resistance** - Results in nonuniform connection strengths.

**Nonzero neuron output impedance** - The neuron output may drop below a full +1 or above a full -1 depending on the load that it drives.

**Electrical Crosstalk** - Electrical activity in neighboring neurons or synpases can effect the behavior of a given neuron or synapse. This problem may be especially bad in neurons where neuron switching may affect $V_{dd}$, $Gnd$, and/or $V_t$ voltage levels resulting in incorrect neuron switching.

**Spatially Nonuniform Illumination** - Depending on the quality of the beam that illuminates the spatial light modulator, the uniformity of the modulator elements, and for holographic storage the quality of the reconstruction, the intensity of light striking each detector may differ from the desired intensity. When using binary connections, this nonuniformity does not cause a significant problem as long as each synapse that is supposed to be turned off receives sufficient illumination.

**Optical Noise and Crosstalk** - Scattering in the optical system, waveguiding on the chip, and for computer generated holographic storage the deterministic encoding noise causes the actual illumination of each detector to deviate from the desired. With binary synapses, this may produce a partial connection when either a zero or full connection is desired.

**Nonuniform Illumination Time** - Variation in the illumination time from cycle to cycle can also cause the connection strengths to deviate from their desired values. If the illumination time is too short, a partial connection may be left where no connection is desired. If the illumination time is too long the strength of full connection may be degraded in the presence of optical scattering or crosstalk.

The net result of these and other potential sources of error is that the neuron will sometimes generate an output that deviates from the desired output based on the input and the desired connection pattern. In this experiment, we measured the

net switching error of the neurons using the same optical setup as in the experiment to measure neuron input voltage (Fig. 4.24). The measured neuron switching error after 3270 total trials using various connection and neuron input patterns are tabulated in Table 4.3. With up to 15 neurons, we found switching errors only for inputs that were just above or just below the switching threshold. With the greater noise that we expect to find in larger chips, we expect that neuron inputs further from the switching threshold will also result in switching errors. However, if we assume that the errors are independent and Gaussian in nature, we expect that the region of neuron switching error will become increasingly small in comparison to the total range of the input domain of the neuron.

Table 4.3 - Neuron Switching Error

| Dist. from Thresh. # | Switching Error (%) |
|---|---|
| -15 | 0 |
| -13 | 0 |
| -11 | 0 |
| -9 | 0 |
| -7 | 0 |
| -5 | 0 |
| -3 | 0 |
| -1 | 1.8 |
| 1 | 1.6 |
| 3 | 0 |
| 5 | 0 |
| 7 | 0 |
| 9 | 0 |
| 11 | 0 |
| 13 | 0 |
| 15 | 0 |

### 4.2.3 Optical Disk-ONNC Neural Network Implementations

In this subsection, we describe two experiments in which we use the optical disk in conjunction with the ONNC to implement multilayer feedforward neural networks. In the first experiment, we implement a two-layer heteroassociative memory.

In the second experiment, we implement a two-layer autoassociative memory.

#### 4.2.3.1 Heteroassociative Memory

For the first experiment, we implement a heteroassociative memory. Because we had not yet developed a learning algorithm for multilayer networks with binary connections, we implemented a two-layer network with a grandmother-cell representation in the hidden layer. In such a network, each cell in the hidden layer is responsible for recognizing a given input and generating the resulting output. We can choose the weights for such a network deterministically. Because each hidden-layer neuron must simply recognize a given input, we connect it to all the neurons in the input layer which should have low outputs when presented with that input. Using Fig. 4.27, we initially set the threshold to 4.88 $V$ such that if the hidden layer neuron is connected to any input units with low outputs, the hidden layer neuron will generate a low output itself. Thus when any of the training inputs is presented to the network, the specified hidden-layer neuron will generate a low output while the outputs of the other hidden layer neurons will be high.

Upon recognition of the input, each hidden layer neuron is responsible for driving the outputs to their associated value. We thus connect each hidden layer neuron to the output neurons corresponding to elements in the associated output which should have outputs low. For the second layer, we use Fig. 4.27 to set the threshold at 1.5 $V$ such that if the hidden layer neuron is connected to any low inputs, its output will be low.

In order to avoid possible variations in threshold voltage for different numbers of connections, we use only inputs with a fixed number of input neurons with high outputs and low. We choose 7 high and 8 low because we have the maximum number of combinations with half the inputs high and half low.

We see that with the threshold set as described above, we have actually built a

neural network PROM with input ORs and output ANDs. However, by adjusting the threshold voltage we can expand the domain of each hidden layer neuron such that inputs nearby in Hamming distance get mapped to the same output. This would then act to correct error or noise in the input.

The grandmother cell representation does not provide many of the touted advantages of neural networks. Its capacity is limited to the number of units in the hidden layer, and by concentrating information about each association in a single unit it is not very resistant to implementation errors.

Table 4.4 shows the input vectors, the hidden representation, and the associated outputs. Table 4.5 shows the weights required to implement the network. In our first implementation, with the same optical setup (Fig. 4.22) as in the experiment to measure the neuron input voltage, we used a foil mask instead of an optical disk to load the connection patterns. Fig. 4.29 shows the response of the neural network chip to the second training vector. The horizontal array of LEDs at the top show the states of the horizontal neurons, while the vertical array shows the states of the vertical neurons. Only the LEDs with white triangles are active. The objective used to image the connection pattern onto the ONNC is visible in the lower right hand side of each photo in front of the ONNC which is also in the lower right hand side. Initially, we set the states of the horizontal neurons (LEDs at the top of Fig. 4.29a) to the input (second training) vector, load the connection pattern by imaging the mask onto the chip, then propagate data through the array to the hidden (vertical layer) (LEDs at right side of Figs. 4.29a and b). The fact that the second LED from the top is off indicates that the second vector has been recognized. We then change masks, reset the synapses, and load the connection pattern for the second layer through the objective and before propagating data from the hidden layer to the output (horizontal) layer whose states are shown in the LEDs at the top of Fig. 4.29b. We were able to successfully recall all 11 vector heteroassociations

Table 4.4 - Vector Heteroassociations

| Neuron # | Input Vector | Output Vector |
|---|---|---|
| 1 | -- +++--+--+++ +- | -++---+---- |
| 2 | ++ ++----+-+-+ -+ | --++-+-++-- |
| 3 | +- -++--+++--- ++ | ++-++----+- |
| 4 | -+ --++++-+-+- +- | +++-++----- |
| 5 | ++ ++--++----+ +- | --++++--+-+ |
| 6 | ++ ++--+----++ +- | -+-+-+--++- |
| 7 | -+ +-+--++++-- -+ | +--+-++++++ |
| 8 | -+ --++++--+-+ -+ | --++--++-+- |
| 9 | ++ -++-----+-+ ++ | --+++---++- |
| 10 | -+ ++--+-++-+- -+ | -++---+-+-- |
| 11 | -+ --++---++++ -+ | +-+++--++++ |

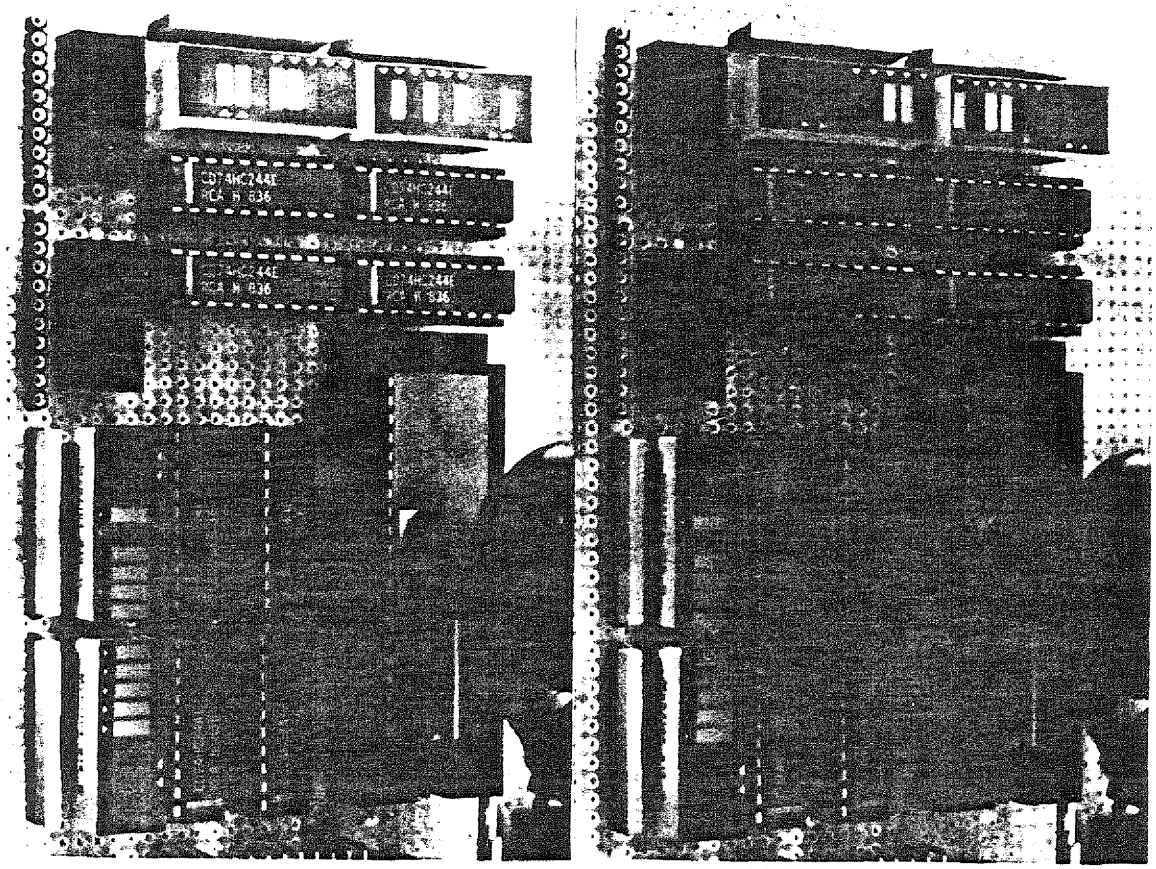
Table 4.5 - Heteroassociative Connection Matrices

$$\underline{w}^{(1)} = \begin{Bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{Bmatrix}$$

$$\underline{w}^{(2)} = \begin{Bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{Bmatrix}$$

(Fig. 4.30).

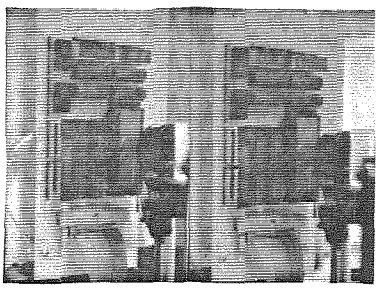
Next, we recorded images of the mask patterns on the optical disk and used the setup of Figs. 4.31 and 4.32. This time, we used a PC to control the entire



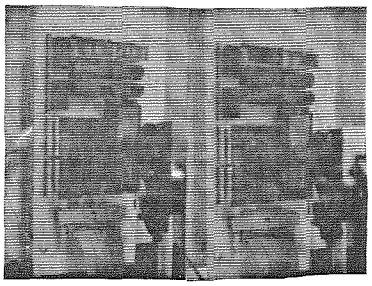
(a) First Layer (b) Second Layer

Fig. 4.29 - Recall of Second Vector Heteroassociation

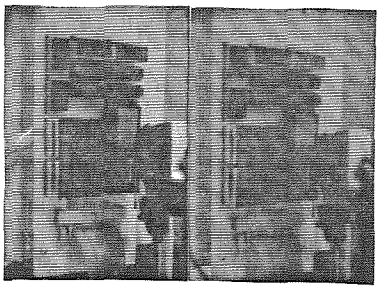
experiment: shutter, clock signals, input and output data, *etc.*. An optional microscope was built to align the mask pattern on the disk with the detectors on the chip. Fig. 4.33 shows such an alignment through the microscope. In this case, the light reflected from the mask on the disk selectively illuminates the synapses. Light reflected from the chip is imaged through the beamsplitter onto a CCD and only those synapses receiving illumination are seen on the video monitor. An alternative method for alignment involved using the chip in "imaging-mode." In this case, a single +1 neuron output is placed in a field of -1s in the vertical neurons. By adjusting the threshold voltage to detect the presence of a single +1, we are able to scan a single row of synapses and readout the mask pattern on the horizontal neurons. We



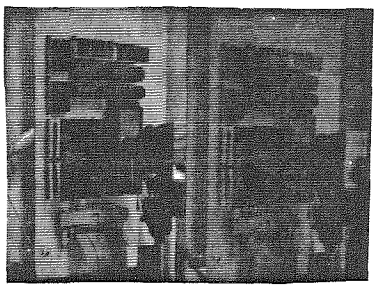
(1)



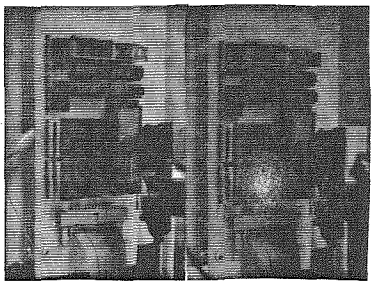
(2)



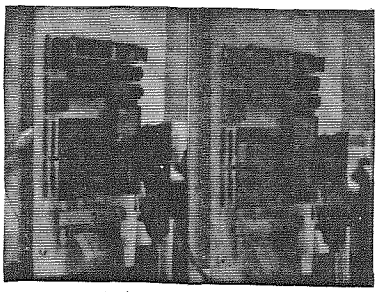
(3)



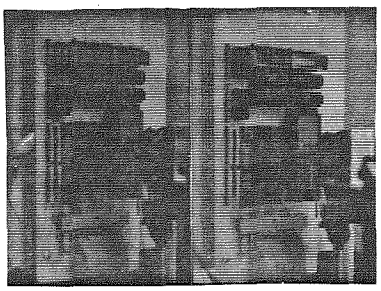
(4)



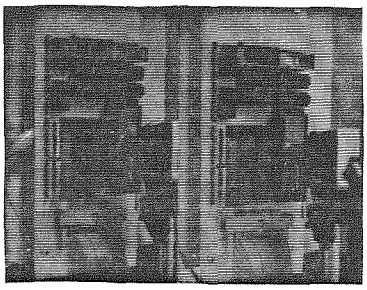
(5)



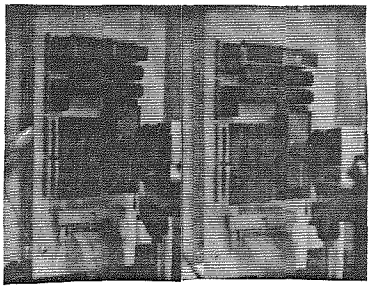
(6)



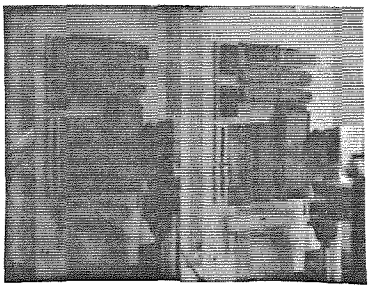
(7)



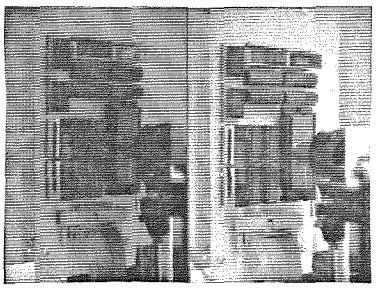
(8)



(9)



(10)



(11)

Fig. 4.30 - Recall of Vector Heteroassociations

are then able to scan the entire array by shifting the position of the +1. Fig. 4.34 shows the resulting computer reconstruction of the mask pattern detected by the chip when illuminated using the same mask as that used for Fig. 4.33.

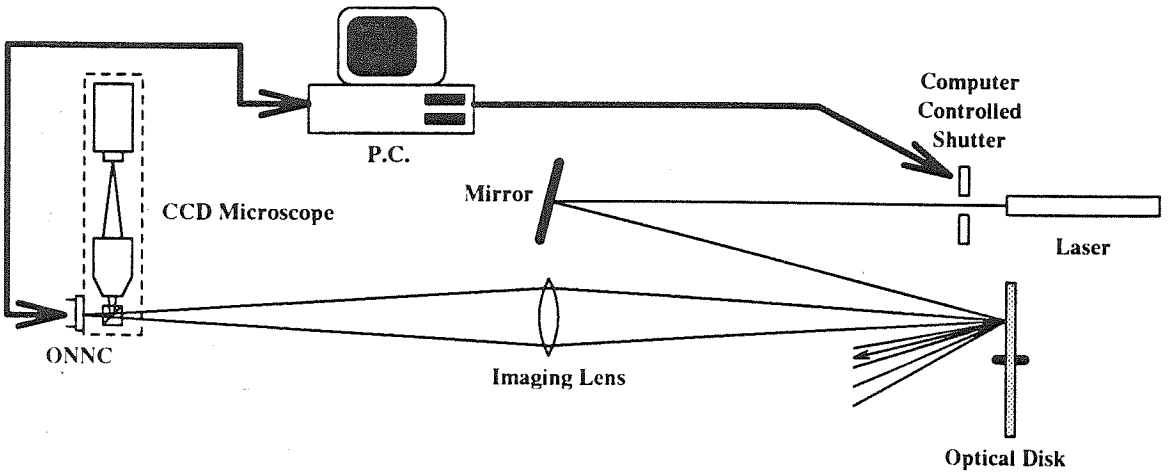



Fig. 4.31 - Optical Disk-ONNC Setup

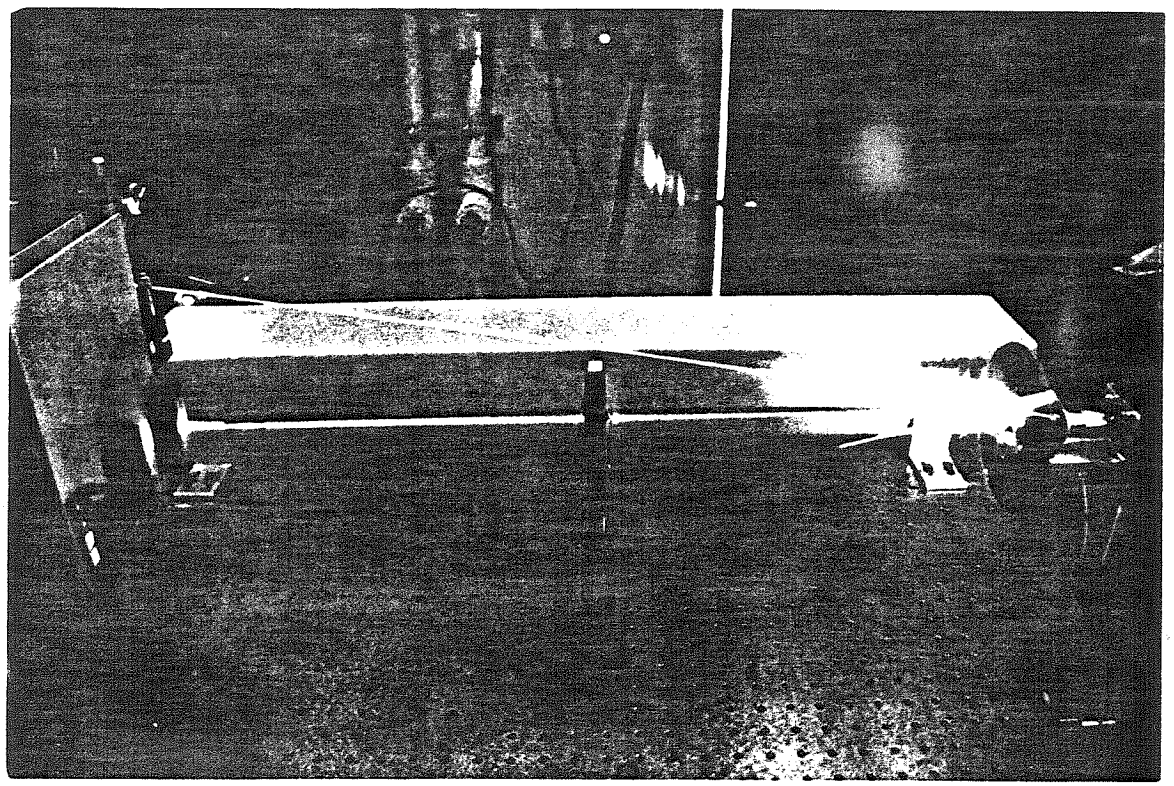


Fig. 4.32 - Photograph of Optical Disk-ONNC Setup

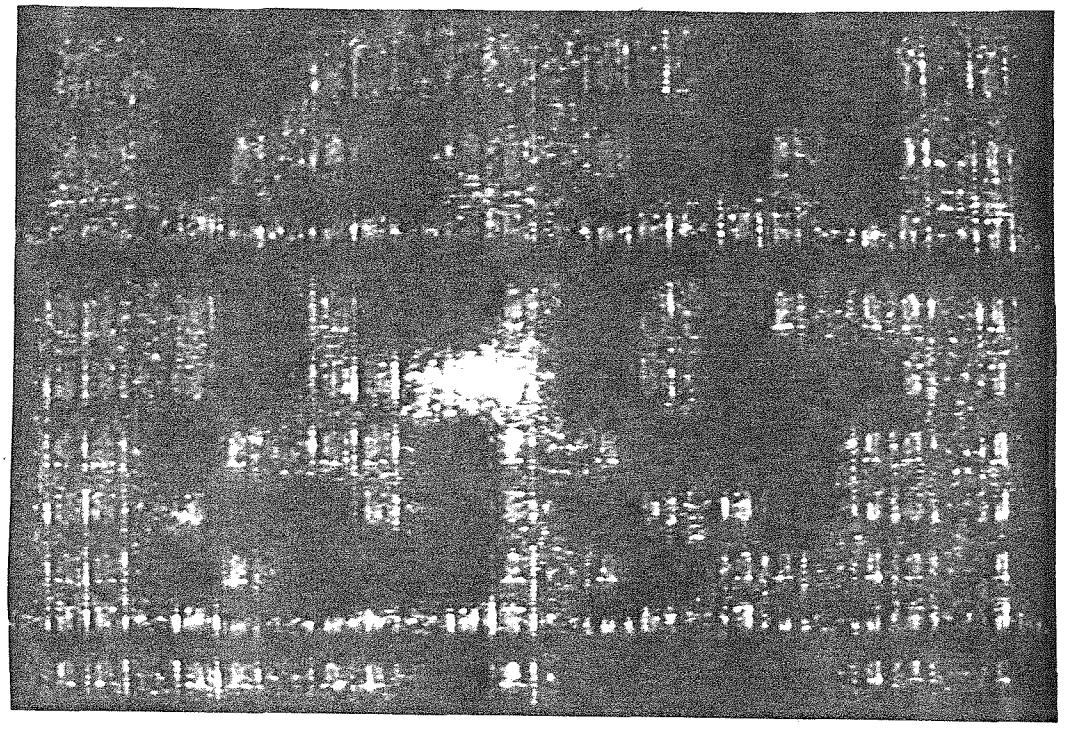


Fig. 4.33 - Microscope Alignment of Disk and Chip

Again, we found that we were able to successfully store and recall all 11 vector heteroassociations.

#### 4.2.3.2 Autoassociative Memory

For the second experiment, we implement an autoassociative memory. In this case, however, we use a more distributed representation in the hidden layer. We use an algorithm developed in our group (Subsec. 1.5.9) for training two-layer networks with binary connections. In our experiments with the algorithm, we found superior performance for bipolar binary synapses as opposed to the unipolar binary synapses implemented by the ONNC.

However, if we restrict all inputs to having the same number of high elements and the same number of low elements, we can implement a network with bipolar binary synapses using unipolar synapses. Let the following equation describe the

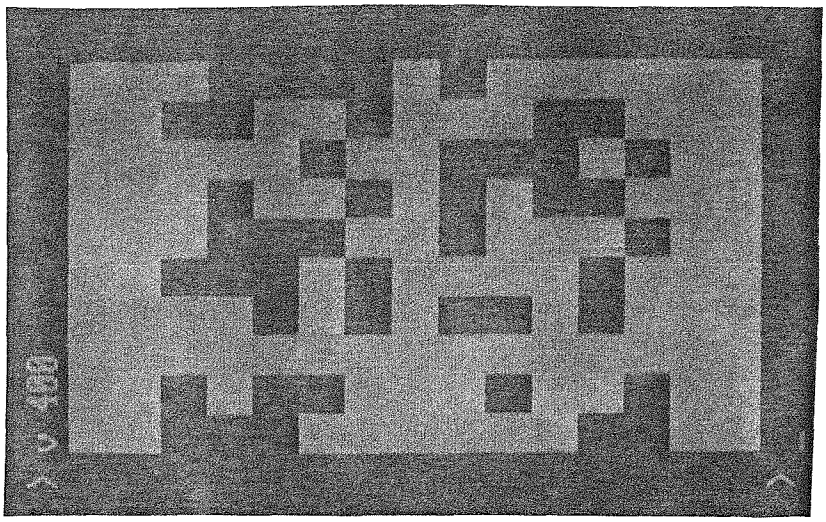


Fig. 4.34 - Imaging-Mode Alignment of Disk and Chip

switching threshold of the neuron:

$$\theta = \sum_i w_i^{(\pm 1)} x_i \tag{4.17}$$

where $\theta$ is the threshold, $\{w_i^{(\pm 1)}\}$ the set of weights connecting the neuron to its inputs, and $\{x_i\}$ the inputs (the outputs of other neurons). We wish to implement this network using a similar weighted sum of unipolar binary connections $\{w_i^{(0,1)}\}$. We can relate the individual bipolar and unipolar equations using the following equation:

$$w_i^{(\pm 1)} = 2 \times w_i^{(0,1)} - 1 \tag{4.18}$$

Substituting Eq. 4.18 into Eq. 4.17, we find that the neuron must now switch under the following condition for the network to act as if it had binary connections:

$$\frac{1}{2}\left(\theta + \sum_i x_i\right) = \sum_i w_i^{(0,1)} x_i \tag{4.19}$$

We see that if all the inputs have the same number of high elements and the same number of low elements, the sum over the elements of the inputs will be the same for all inputs, and every threshold will be adjusted by the same amount. In our experiment, we restricted ourselves to having only inputs with 5 elements high and 5 low. Furthermore, we restricted the algorithm to search for bipolar networks with neurons switching at zero threshold. We see from comparison of Eqs. 4.18 and 4.19, that a desired bipolar network with an equal number of high and low inputs can be implemented with a unipolar network.

Table 4.6 shows the stored vector autoassociations and the associated hidden representations. Table 4.7 shows the weights required to implement the network. We used the same two setups (foil and disk) as for the heteroassociative memory. Fig. 4.35 shows the propagation of all 6 vector autoassociations through the foil-ONNC setup (see Subsubsec. 4.2.3.1 for explanation of figure). In order to implement bipolar weights in the second layer, we detected the states of the five active neurons in the hidden layer, and set five additional neurons in the hidden layer to complementary values. We then implement positive weights by connecting the active neurons and negative weights by connecting to complementary neurons. In both the foil and disk cases, we were able to successfully store and recall all vector autoassociations.

Table 4.6 - Vector Autoassociations

| Vector # | Vector | Hidden Rep. |
|---|---|---|
| 1 | +++++----- | +-+-- |
| 2 | +---++++- | -+--- |
| 3 | +-+-+-+-+- | ++--- |
| 4 | ++-+-+-+-- | --+-- |
| 5 | ++--++---+ | ++++- |
| 6 | --++--++-+ | ---++ |

## 4.3 Future Directions

Table 4.7 - Weights

$$\underline{w}^{(1)} = \begin{Bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{Bmatrix}$$

$$\underline{w}^{(2)} = \begin{Bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{Bmatrix}$$

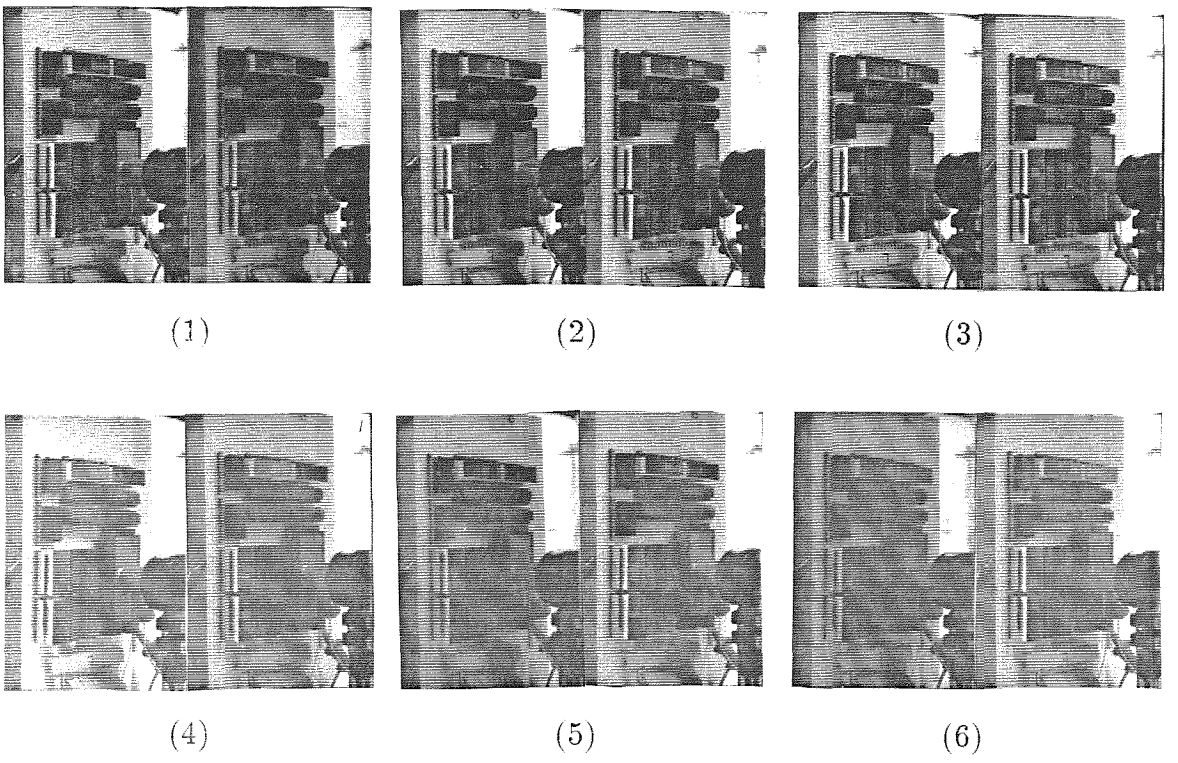



Fig. 4.35 - Recall of Vector Autoassociations

In the near term, it would be interesting to experiment with some alternative

synapse designs. In the long term, optical connections instead of electronic will be developed.

### 4.3.1 Alternative Optically Controlled Electronic Connections

Many potential alternative architectures exist for implementing analog connections. For example, we could use the photodiode to provide the bias current for an analog multiplier as shown in Fig. 4.35a. In this case, the output current would be given as follows:

$$I = I_p \tanh\left[\frac{q(V_i - V)}{kT}\right] \tag{4.20}$$

If we remain in the linear regime, this synapse would allow us to use analog neurons and connections. The synapse of Fig. 4.36b uses unipolar binary neurons to gate analog weighted currents. We can also make use of a multiplying digital-to-analog converter (MDAC) style neuron such as those used by Alspector[21]. As shown in Fig. 4.36c, the photodiodes would control the gates of a series of transistors with exponentially scaled width/length ratios. $N$ photodiodes per synapse would give each synapse $N$-bits of dynamic range.

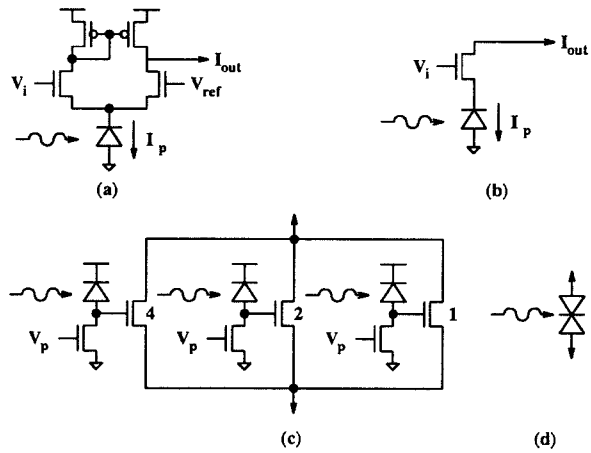



Fig. 4.36 - Alternative Optoelectronic Synapse Designs

Boyd has proposed a synapse (Fig. 4.36d) consisting of back-to-back photodi-

odes.[22] The I-V characteristic of this synapse is as follows:

$$I = (I_p + I_0)\tanh\left(\frac{qV}{2kT}\right) \tag{4.21}$$

For $I_p >> I_0$ and small $V$, this synapse can implement analog connections for bipolar analog neurons.

In order to implement bipolar weights, we can use a pair of lines to each row and column of synapses corresponding to positive and negative currents (excitation and inhibition). For example, the MDACs pictured in Fig. 4.37 could be used to implement a 5-bit signed discrete connection. We actually fabricated a synapse of type (a) (Fig. 4.38) to test its ability to implement discrete weights and found that it worked properly. However, the large size of this synapse ($120\mu m \times 120\mu m$) led us to abandon this technique in favor of the single photodetector implementation.

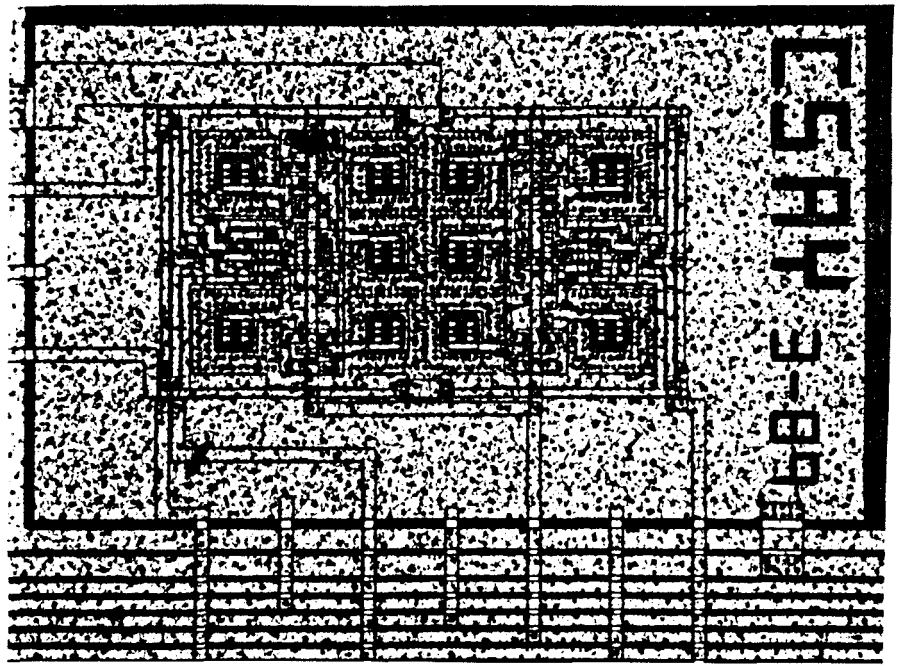


Fig. 4.38 - Photograph of 5-Bit Bipolar Synapse

### 4.3.2 Optical Connections

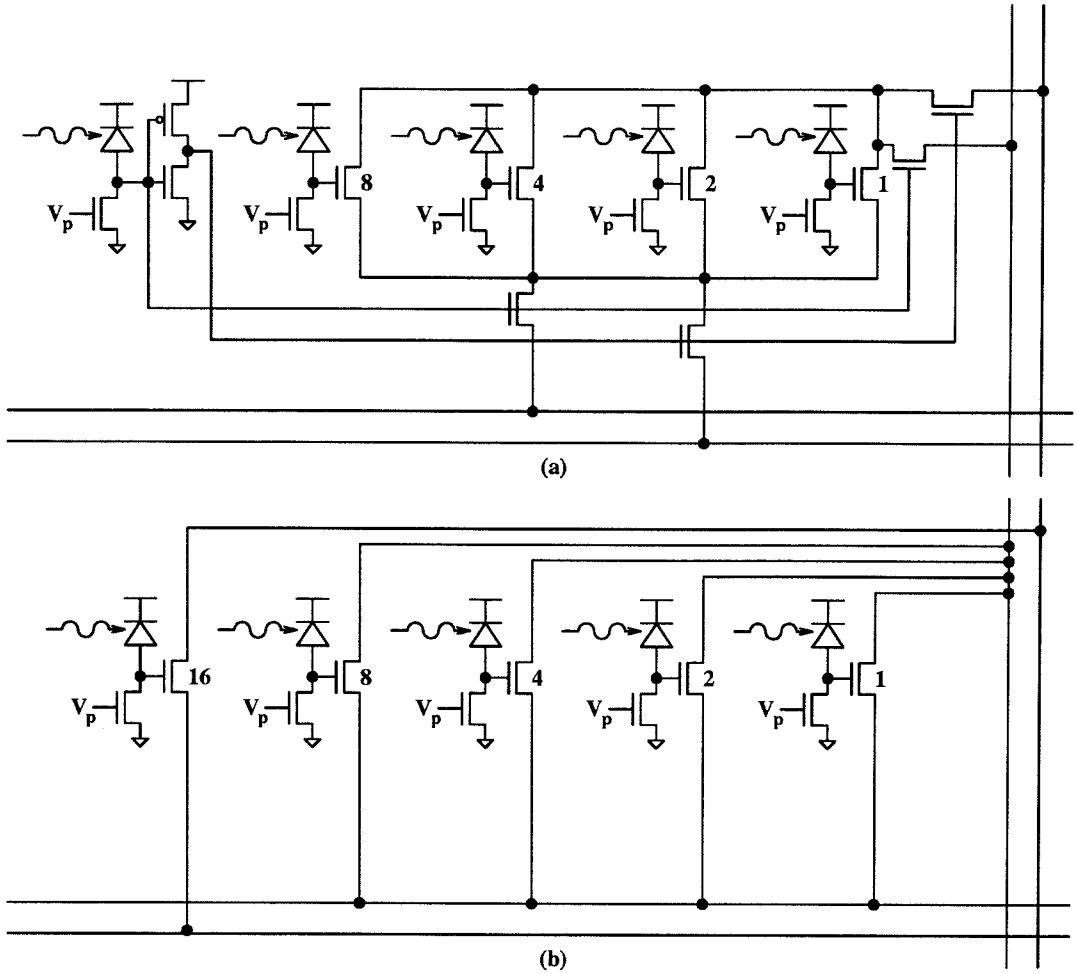



Fig. 4.37 - Bipolar Synapses with 5-Bits of Dynamic Range

Although we were able to successfully demonstrate an optoelectronic neural network, the main contribution of this work is the demonstration that optical storage and parallel readout from an optical disk can be effectively exploited with processing elements fabricated on a custom chip.

Most of the chip area in this design however is occupied by connections. This is a direct result of the fact that we provided no means for the neurons to optically broadcast their outputs. In the future, with the possibility of neurons with optical detectors and either sources[23] or modulators,[24] it should be possible to fill the chip area with processing elements and use the vertical or third dimension to provide connections. Because the optical disk is a planar medium, the number of degrees of

freedom it provides grows as $O(L^2)$ where $L$ represents the linear scale of the system. Although the number of neurons can grow as $O(L^2)$, full interconnection requires $O(L^4)$ degrees of freedom. We can use a canonical vector-matrix architecture in an imaging configuration to provide connections between a row and a column of neurons as shown in Fig. 4.39a. We can also use holographic techniques between fractal planes[25] to provide connections between two planes containing $O(l_1)$ and $O(l_2)$ neurons respectively where $l_1 l_2 = L^2$ (Fig.4.39b).

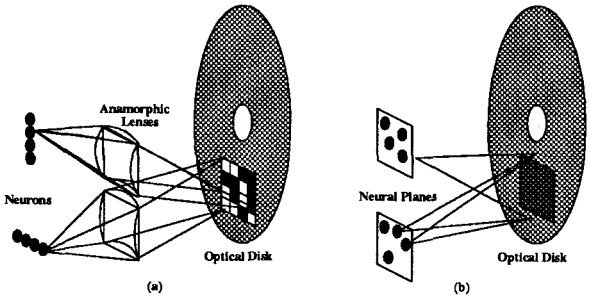



Fig. 4.39 - Optical Disk Implementation of Neural Connections

## 4 References

[1] C. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley Publishing Co., Menlo Park, CA, 1989.

[2] D. Pomerleau, G. Gusciora, D. Touretzky, and H. Kung, "Neural Network Simulation at Warp Speed: How we got 17 Million Connections Per Second," *Proc. of IJCNN*, pp. 143-150, July, 1988.

[3] U. Ramacher, J. Beichter, and N. Brüls, "Architecture of a General-Purpose Neural Signal Processor," *Proc. of IJCNN*, pp. 443-446, July, 1991.

[4] J. Elias, M. Fisher, M. Monemi, "A Multiprocessor Machine for Large-Scale Neural Network Simulation," *Proc. of IJCNN*, pp. 469-474, July, 1991.

[5] M. Witbrock, M. Zasha, "An Implementation of Back-Propagation Learning on GF11-A Large SIMD Parallel Computer," *Tech. Rept. CMU-CS-89-208*, Carnegie-Mellon University, 1989.

[6] X. Zhang, *et al.*, "An Efficient Implementation of the Back Propagation Algorithm on the Connection Machine CM-2," *Advances in Neural Information Processing Systems*, Vol. 2, D. Touretzky, ed., Morgan-Kaufmann, 1990.

[7] M.A. Holler, "VLSI Implementations of Learning and Memory Systems: A Review," *Advances in Neural Information Processing Systems*, Vol. 3, R. Lippman, *et al.*, eds., Morgan-Kaufmann, San Mateo, CA, pp. 993-1000, 1991.

[8] H.P. Graf, *et al.*, "Reconfigurable Neural Net Chip with 32K Connections," *Advances in Neural Information Processing Systems*, Vol. 3, R. Lippman, *et al.*, eds., Morgan-Kaufmann, San Mateo, CA, pp. 1032-1038, 1991.

[9] H. McCartor, "Back Propagation Implementation on the Adaptive Solutions CNAPS Neurocomputer Chip," *Advances in Neural Information Processing Systems*, Vol. 3, R. Lippman, *et al.*, eds., Morgan-Kaufmann, San Mateo, CA, pp. 1028-1031, 1991.

[10] M. A. Holler, *et al.*, "An Electrically Trainable Artificial Neural Network

(ETANN) with 10240 'Floating Gate' Synapses," *Proc. IJCNN*, 1989.

[11] D. Psaltis and N. Farhat, "Optical Information Processing Based on an Associative Memory Model of Neural Nets with Thresholding and Feedback," *Optics Letters*, Vol. 10, pp. 98-100, January 1985.

[12] K. Hsu, D. Brady, and D. Psaltis, *Proceedings of the IEEE Conference on Neural Information Processing Systems*, Denver, November 1987.

[13] W.P. Bleha, *et al.*, *Optical Engineering*, Vol. 17, No. 371, 1978.

[14] A.B. VanderLugt, *IEEE Trans. Inform. Theory*, Vol. IT-10, No. 139, 1964.

[15] D. Psaltis, *et al.*, *Second Topical Meeting on Optical Computing*, Incline Village, Nevada, March 1987.

[16] D. Brady, Ph.D. Thesis, California Institute of Technology, 1990.

[17] K. Wagner and D. Psaltis, "Multilayer Optical Learning Networks," *Applied Optics*, Vol. 26, No. 23, pp. 5061,5076, 1 December 1987.

[18] R. Agranat, C. Neugebauer, and A. Yariv, "A CCD Based Neural Network Integrated Circuit with 64K Analog Programmable Synapses," *Proc. IJCNN*, 1990.

[19] J. Ohta, *et al.*, "GaAs/AlGaAs Optical Synaptic Interconnection Device for Neural Networks," *Optics Letters*, Vol. 14, No. 16, pp. 844-846, 1989.

[20] E.A. Rietman, *et al.*, "Amorphous Silicon Photoconductor Arrays for Artificial Neural Networks," *Applied Optics*, Vol. 28, No. 15, pp. 3474-3478, 15 August 1989.

[21] J. Alspector and R.B. Allen, "A Neuromorphic VLSI Learning System," *Advanced Research in VLSI Proc. 1987 Stanford Research Conf.*, pp. 313-349, MIT Press, 1987.

[22] G.D. Boyd, "Optically Excited Synapse for Neural Networks," *Applied Optics*, Vol. 26, No. 14, 15 July 1987.

[23] S. Lin, Ph.D. Thesis, California Institute of Technology, 1991.

[24] T. Drabik, Ph.D. Thesis, Georgia Institute of Technology, 1990.

[25] X. Gu, Ph.D. Thesis, California Institute of Technology, 1990.

# 5 Conclusions

In Chap. 2, we noted that a control signal can be divided into feedforward and feedback componenets. We stated the belief that the greater the role of feedforward control, the better we can fully exploit the capabilities of the plant. We then considered the application of multilayer feedforward neural networks to feedforward control. We feel that neural networks have a role to play in control because learning and adaptation of neural networks could be used to advantage for the control of plants that are uncertain, complex, nonlinear, time-varying, or otherwise hard to model. Plants with one or more of these characteristics can be difficult to control using conventional techniques.

The problem with using the supervised techniques we considered to learn feedforward control is that we must have an estimate of the error at the output of the (control) network in order to modify the connection weights in the network. However, we assume that we have incomplete knowledge about the plant input (network output) required to generate a desired plant otuput. We thus considered two architectures/algorithms, generalized and specialized learning, for training feedforward controllers. Generalized learning provides a direct measurement of error but not the ability to completely specify the region in which we train the network to operate the plant. On the other hand, provided that we can estimate the error at the output of the network given the error at the output of the plant, specialized learning allows us to specify the region in which we train the network.

We derived a specialized learning algorithm called BEPing through the plant

by considering the plant to be an unmodifiable part of the network and applying gradient descent. In order to do this, we require estimates of the plant Jacobian. In simulations, we accomplish this by perturbing plant inputs and observing the effect on the outputs of the plant. We also proposed alternative techniques such as least squares estimation of the Jacobian, and noted the parallel between our problem of estimating the plant Jacobian and the problem of parameter estimation for self-tuning regulators in adaptive control. Other authors have proposed using either fixed gains or a neural network model of the plant to estimate the control network output error given the plant output error.

We then performed a series of experiments. In the coordinate conversion problem, we found that both generalized and specialized learning (in the form of BEPing through the plant) were effective in inverting a static mapping. In the one-link manipulator problem, we found BEPing through the plant effective and noted that random selection of training points from an infinite set provided better generalization than the use of a fixed finite training set. In light of the VC-dimension results, we note that the random training set attempts to provide a number of training samples far in excess of the VC-dimension. In the two-link manipulator problem, we found that even random training sets were unable to provide good generalization given *ad hoc* selection of network architecture. In the two-link manipulator problem revisited, we found that by restriction of the task (and thus the region of operation) and by using improved algorithms that dynamically adjust the network architecture, we were able to train an effective controller for the two-link manipulator. Furthermore, we demonstrated techniques to incrementally increase the region of operation so that a difficult task could be learned, one piece at a time.

We also noted that a tapped delay line could be used to convert a dynamic input signal into a static vector of samples so that a feedforward network could respond to input dynamics. We found that as expected, feedback is required for

effective control of the plant. We incorporated feedback in operation by modifying the desired trajectory given to the feedforward controller based on both desired and actual trajectories. Finally, we noted that one could not train a network online to control a plant without the an initial ability to control the plant. We proposed and demonstrated a technique called feedback learning that allows us to train online using a plant that we have only the most rudimentary ability to control.

In Chap. 3, we noted that current commercial optical disk systems do not exploit a major advantage of optical storage over its magnetic counterparts–the ability to access data in parallel. We then characterized an existing prototype disk system in terms of its suitability for parallel access. Next, we proposed and demonstrated several techniques, both imaging and holographic, for the parallel readout of information from optical disks. We noted that if nothing else, parallel access of optical disks can be used to significantly increase the bandwidth of information readout from the optical disk. We concluded by proposing several applications such as character recognition, database processing, and neural network weight storage that further exploit the advantgaes of optical disk. These applications typically involve the rapid search through a large library of information.

In Chap. 4, we described an optoelectronic implementation of a multilayer feedforward neural network using a custom VLSI chip with neurons and recoonfigurable connections and an optical disk for the storage of connection weights. In order to provide the maximum possible number of neurons and connections, we chose to minimize the circuitry in the synapse by using a pulldown FET and photodiode to set the gate voltage of a single synaptic FET connecting a pair of neurons. We described in detail the design and operation of the chip, and found that except in the case of unipolar binary neurons, the single FET could be used only for binary connections. We then experimentally demonstrated that the successful transfer of connections from the disk to chip, and the successful operation of the chip itself. Although

this technique may not be optimal for the implementation of neural networks, the experiments in this chapter demonstrated that parallel readout from optical disks can be exploited successfully using special purpose optoelectronic readout elements.

Neural networks provide a computational alternative to conventional computers based on the Von Neumann machine. With their emphasis on learning and adaptation instead of algorithmic solutions, they may provide solutions to problems such as the control of complex, nonlinear systems, that are currently difficult to solve using conventional techniques, especially if those problems involve a large amount of sensory or input information best handled through the parallel processing inherent in neural architectures. Although there is much work to be done on theoretical development of neural networks, this must take place concurrently with experimental implementation. Experimentation not only serves to test theoretical results, but also to generate new questions as unexpected behavior comes to light. Although conventional computers continue to increase in power, they can only be used to effectively simulate hundreds or at the very most tens-of-thousands of neurons. Special purpose hardware must be developed for experimental testing of systems involving even greater numbers of neurons that may be required for real-time sensory information processing or the emergence of new and interesting behavior in collective computation.