HAC REF G9346



12

# STIMULATED PHOTOREFRACTIVE OPTICAL NEURAL NETWORKS

Y. Owechko, G. Dunning, G. Nordin, and B.H. Soffer

Hughes Research Laboratories
3011 Malibu Canyon Road
Malibu, California 90265



December 1992

Final Report
Contract N00014-89-C-0274
September 1989 through October 1992



OFFICE OF NAVAL RESEARCH
Department of the Navy
800 N. Quincy Street
Arlington VA 22217-5000

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION / AVAILABILITY OF REPORT |
| 2b. DECLASSIFICATION / DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>N00014-89-C-0274 | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Hughes Research Laboratories<br>6b. OFFICE SYMBOL *(If applicable)* | 7a. NAME OF MONITORING ORGANIZATION<br>Office of Naval Research |
| 6c. ADDRESS *(City, State, and ZIP Code)*<br>3011 Malibu Canyon Road<br>Malibu, CA 90265 | 7b. ADDRESS *(City, State, and ZIP Code)*<br>Department of the Navy<br>800 N. Quincy Street<br>Arlington, VA 22217-5000 |
| 8a. NAME OF FUNDING / SPONSORING ORGANIZATION<br>8b. OFFICE SYMBOL *(If applicable)* | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS *(City, State, and ZIP Code)* | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO. |

11. TITLE *(Include Security Classification)*

STIMULATED PHOTOREFRACTIVE OPTICAL NEURAL NETWORKS

12. PERSONAL AUTHOR(S)

Owechko, Y., Dunning, G., Nordin, G., and Soffer, B.H.

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT *(Year, Month, Day)* | 15. PAGE COUNT |
|---|---|---|---|
| Final | FROM 9/29/89 TO 10/31/92 | 92/12/15 | 131 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)* |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | |
| | | | |
| | | | |

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This final report describes research in optical neural networks performed under DARPA sponsorship at Hughes Aircraft Company during the period 1989-1992. The objective of demonstrating a programmable optical computer for flexible implementation of multi-layer neural network models was successfully achieved. The advantages of optics for neural network implementations include large storage capacity, high connectivity, and massive parallelism which result in high computation rates. The optical neurocomputer developed on this program is based on a new type of holography, cascaded grating holography (CGH), in which the neural network weights are distributed among angularly- and spatially-multiplexed gratings generated by stimulated processes in photorefractive crystals. This approach reduces crosstalk and improves the utilization of the optical input device. Successfully implemented neural networks include the Perceptron, Bidirectional Associative Memory, and multi-layer backpropagation networks. Up to $10^4$ neurons, $2\times10^7$ weights, and processing rates of $2\times10^7$ connection updates per second were achieved. Packaging concepts for future versions of the neurocomputer were also studied.

| | |
|---|---|
| 20. DISTRIBUTION / AVAILABILITY OF ABSTRACT<br>☐ UNCLASSIFIED / UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>William Miceli | 22b. TELEPHONE *(Include Area Code)*<br>(617) 457-4484<br>22c. OFFICE SYMBOL |

DD Form 1473, JUN 86 *Previous editions are obsolete*

## PREFACE

This report presents a developmental prototype knowledge-based expert system (ES) RETAININGEARTH for the selection and design of earth retaining structures. This study will provide guidance for the selection of an appropriate retaining structure, based on a given set of input conditions and the subsequent detailed design of the selected structure. Funding for this study was provided by Headquarters, US Army Corps of Engineers, and the study was monitored by the Information Technology Laboratory (ITL), US Army Engineer Waterways Experiment Station (WES), Vicksburg, Mississippi. Partial graduate assistantships were provided by the Department of Ocean Engineering, Florida Atlantic University, Boca Raton, Florida.

The report was written by Dr. M. Arockiasamy as Principal Investigator with able assistance from graduate students. The computer code of the ES was developed by Giri Sreenivasan and Keling Shen who worked in this project as Graduate Research Assistants. Ms. Barbara Steinberg typed the report coordinating the text and tables layout.

The work was monitored at WES by Mr. Michael E. Pace, under the general supervision of Mr. H. Wayne Jones, Chief, Scientific and Engineering Applications Center, and Dr. N. Radhakrishnan, Director, ITL.

At the time of publication of this report, Director of WES was Dr. Robert W. Whalin. Commander and Deputy Director was COL Leonard G. Hassell, EN.

**ACKNOWLEDGEMENT**

The authors would like to thank Dr. N. Radhakrishnan, Chief, Information Technology Laboratory (ITL), US Army Engineer Waterways Experiment Station (WES), Vicksburg, Mississippi (the sponsor of this project), for his valuable guidance, constructive criticism and cooperation. The authors sincerely appreciate the constructive and critical comments of Mr. Michael Pace, Civil Engineer, ITL at WES which were very valuable in revising the draft final report. Grateful acknowledgments are due to several researchers and organizations whose references have been adapted, used or reproduced freely in the preparation of this report. They wish to express their appreciation to Dr. S.E. Dunn, Professor and Chairman, Department of Ocean Engineering and Dr. Craig S. Hartley, Dean of Engineering, Florida Atlantic University for their support, continued interest and encouragement.

Finally, Ms. Barbara Steinberg is acknowledged for her excellent and patient typing.

TABLE OF CONTENTS

# TABLE OF CONTENTS

# ILLUSTRATIONS

# ILLUSTRATIONS

# ILLUSTRATIONS

# ILLUSTRATIONS

# SECTION 1
# EXECUTIVE SUMMARY

This final report describes research in optical neural networks performed at Hughes Aircraft Company under a three-year DARPA sponsored contract. The objective of demonstrating a programmable optical computer for flexible implementation of neural network models was successfully achieved. The advantages of optics for neural network implementations include high storage capacity, connectivity, and very fine-grained parallelism which results in high computation rates. The optical neurocomputer developed under this program is based on a new type of holography, cascaded grating holography, in which the neural network weights are distributed among cascaded angularly- and spatially-multiplexed gratings. This approach reduces crosstalk and improves the utilization of the optical input device. Successfully implemented neural networks include the Perceptron, Bidirectional Associative Memory, and backpropagation neural networks. Up to $10^4$ neurons, $2x10^7$ weights, and processing rates of $2x10^7$ connection updates per second were achieved on this program.

The organization of this final report is as follows. First, we briefly describe the nature of neural network models and the types of problems they are meant to address. We describe real-time holography and its advantages and disadvantages for neural network implementations in terms of storage capacity, connectivity, and parallel processing. A new holographic technique, cascaded grating holography (CGH), was developed by us to overcome an important source of distortion in holographic neural networks. We demonstrated CGH in photorefractive $BaTiO_3$ crystals using both visible light (514 nm) from an argon laser and infrared light (830 nm) from a laser diode. The infrared experiments are especially significant because very compact systems can be built using laser diode light sources.

We then discuss the design and construction of the optical neurocomputer based on CGH. The number of neurons, number of layers, and the neuron activation function can all be programmed without hardware adjustments. We believe ours is the first truly programmable optical neurocomputer. In addition, the simple system design requires only a single crystal, input spatial light modulator, and output detector regardless of the network configuration. The entire system was built from presently available off-the-shelf components. Although the present research system occupies a relatively large volume, we discuss packaging concepts in which the entire system is contained within a volume smaller than a shoe box.

Three widely differing neural network algorithms (Perceptron, bidirectional associative memory, and backpropagation) were successfully implemented on the neurocomputer. As of this

writing, up to 10,000 neurons, $2x10^7$ weights, and learning rates of $2x10^7$ connection updates per second have been achieved. The potential exists for improvement factors of $10^3$ in the number of weights and $10^6$ in the processing rate. Finally, we discuss our experimental efforts in "fixing" or permanently recording connection weights in the holographic medium.

# SECTION 2
# NEURAL NETWORK MODELS OF COMPUTATION

Computational problems can be described in terms of many attributes. One of the most fundamental measures is the "randomness" or algorithmic complexity of the problem. It is generally agreed that nonrandom highly structured problems with known algorithmic solutions are best solved using traditional computer programs. In contrast, many problems involving natural data have a high degree of randomness, especially pattern recognition problems involving noisy data. These are problems that biological organisms excel at compared with classical rule-based algorithms in which the rules are assumed *a priori*. Biologically inspired neural network models are useful for solving such high entropy problems in which the underlying algorithm is unknown and the required transformations must be learned from examples. They are more powerful than standard statistical techniques because a larger range of solutions can be represented using their multi-layer nonlinear structure. In fact, it has been shown that any function can be approximated by a sufficiently large neural network.[1]

Neural network models of computation consist of many simple processing nodes or "neurons" which communicate with each other via interconnection weights. The nodes are anthropomorphically called neurons here in acknowledgment of the vastly more sophisticated biological neurons which inspire connectionist models of computing. (However, any resemblance of the nodes to actual neurons is, at best, superficial and grossly simplified.) A diagram of the logical structure of a multi-layer feed-forward neural network is shown in Fig. 1. Associated with each neuron is an activation level which is calculated from a weighted sum of the activity levels of other neurons.

$$y_i^{(n)} = f\left(x_i^{(n)}\right)$$
$$x_i^{(n)} = \sum_j w_{ij}^{(n)} y_j^{(n-1)}$$

Patterns which are input to the network through the bottom layer are transformed into patterns which represent the answer to the problem the network is trained to solve. Many neural net architectures have been designed and demonstrated for various computing tasks. Techniques such as backpropagation as well as many others have been developed to "train" or adjust the interconnection weights to solve problems in pattern recognition, vision, and robotic control. Although the problem of learning an arbitrary transformation for a completely random problem has

been shown by Judd[2] to be NP-complete (therefore probably requiring exponential increases in learning time as the problem size increases), most if not all problems in pattern recognition involve at least partially structured data for which the learning time will be a polynomial function of the problem size.

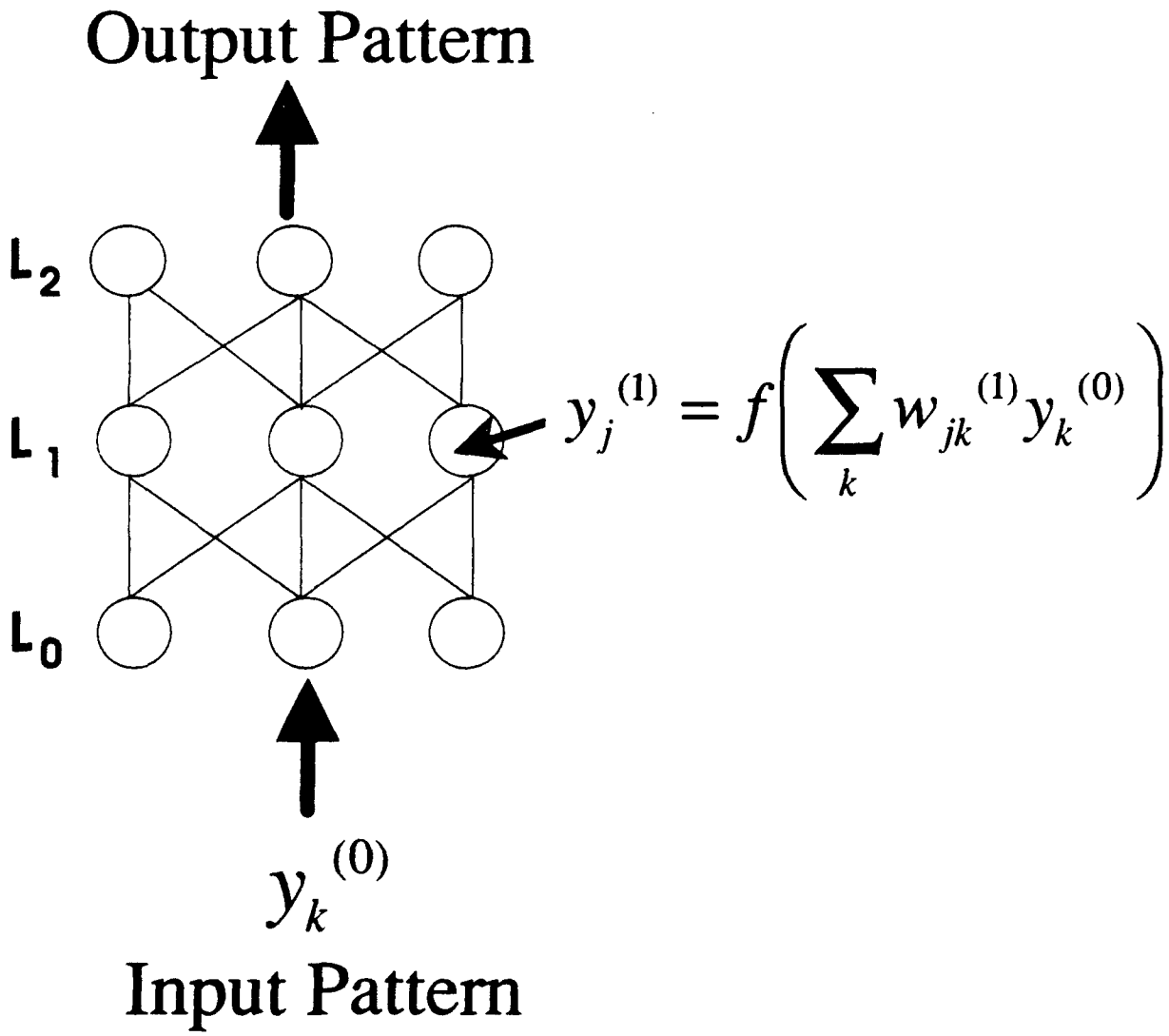



**Fig. 1. Structure of feed-forward neural networks.**

Two important parameters which characterize a neural network are the number of neurons, N, and their connectivity, K, where K is the number of synapses or weights connected to a neuron. We argue that both N and K should be as large as possible in any general purpose neurocomputer. Pattern recognition problems involving natural data, especially vision or target recognition, require large numbers of input neurons to handle the large raw input data rate. In addition, large numbers of neurons in the hidden layer are required in order to solve non-trivial

problems. As for the connectivity, Abu-Mostafa[3] has shown that it must be large for at least two reasons. First, since the neurons essentially implement K-input threshold functions, one K-input neuron with K associated weights is equivalent to order $K^2$ two-input neurons with $2K^2$ associated weights. Thus far fewer weights are required if the neurons have high fan-in and fan-out. Second, Abu-Mostafa also showed that in order for a neural network to learn, the connectivity K must exceed the entropy H of the environment where H is the $\log_2$ of the number of input patterns typically generated by the environment. H increases as the randomness of the problem increases. Since neural networks are most useful for partially random problems, we argue that both the number of neurons and their connectivity should be as large as possible in a neuro-computer. Increasing N and K while maintaining computational parallelism is a daunting task for electronic architectures due to the 2-D nature of electronic interconnects. Most of the area on analog or digital electronic neuro-chips is taken up by the interconnects while implementing only modest numbers of neurons ($N=O(10^2)$).

Optical implementations of neural networks are attractive because of the large storage capacity and, most importantly, the parallel access and processing capabilities of optics. Optical architectures can exploit 3-D free space interconnects, allowing the input and output planes to be fully populated with highly interconnected neurons ($N=O(10^5)$). Moreover, an entire weight layer can be updated in one time step, unlike electronic approaches which must use some form of time multiplexing when the number of neurons is large. Optical neural networks divide naturally into two classes distinguished by the dimensionality of the weight storage medium, e.g. weights can be stored in 2-D or 3-D formats. Example 2-D weight storage media include film, SLMs, and optical disks while almost all of the 3-D formats use photorefractive crystals as the storage medium. The optical processor described here uses 3-D weight storage based on volume holography. The primary motivation for considering volume holograms as a storage medium for neural networks is the potential for extremely high storage capacity and fully parallel processing of the weights during both the learning and reading phases.[4] Motivation for maximizing these parameters can be found in the potential application areas for neural networks.

Figure 2 is an adaptation of a figure which originally appeared in the final report of the 1988 DARPA Neural Network Study. It shows the potential application areas mapped onto a 2-D space in which one axis is the storage required in terms of the number of weighted connections and the other axis is the processing rate required in connections per second. The corresponding estimated parameters of some biological systems are included. A variety of electronic implementations, denoted by open squares, have also been added to Fig. 2. (These performance points were taken from an article by Alspector.) The electronic implementations are, apart from the Sun and Cray computers, specialized analog and digital chips. Some implement learning and some do not. For those that don't, weight values must be learned off-chip and subsequently loaded into the chip.

Finally, diagonal lines of constant network update time were added to the figure. The update time is the time required for information to pass from the input of the network to the output. Such a plot is necessarily a highly folded and simplified projection of a high dimensional reality onto a low dimensional representation. For example, the degree of local vs global connectivity is ignored as well as the algorithmic complexity of the application. Nevertheless, certain trends can be deduced.

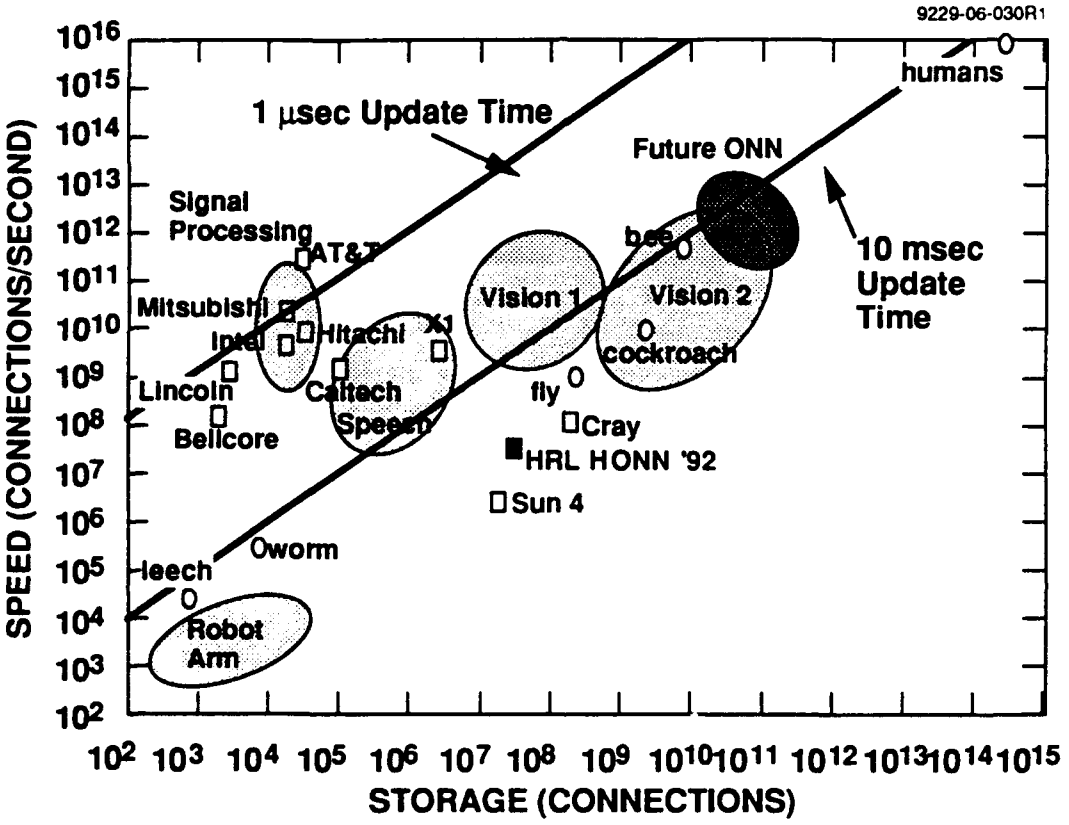



**Fig. 2. Relative performance of present holographic neural network (HONN) compared to application requirements and other hardware implementations.**

It is interesting to note that potential application areas in robotics, speech, and vision cluster around the 10 msec network update time line. These applications require large numbers of weights in order to achieve the complexity required to solve the problem, but the problem does not need to be solved in less than a few milliseconds. The large number of weights, however, requires very high processing rates to achieve this update time. Significantly, the biological systems also cluster around the 10 msec update line, perhaps because they must also solve problems in speech and vision. With the exception of the general purpose computers, the electronic implementations have relatively modest storage capacity, although their processing rates are high. (Ignoring the fact that many of the chips do not have on-chip learning.) This is due to the 2-D nature of VLSI which

limits the number of connections that are practical. They therefore appear most suited to the signal processing applications in which fast update times and modest storage are required.

It is our opinion that optical neurocomputers are complementary to specialized electronics in that the 3-D connectivity and parallelism of optics permits the implementation of very large networks with high processing rates and relatively modest network update times suitable to applications in vision. The projected future performance of our optical neurocomputer is indicated by the large filled circle labeled ONN. The present performance of our optical neurocomputer is also indicated by the filled square. Despite being a proof-of-principle first prototype system using non-optimized components, its performance is quite good.

In order to fulfill a practical role complementary to the strengths of electronics, optical neurocomputers should include, first, large numbers of neurons and weights; second, distortionless software mapping of a variety of neural network algorithms onto the hardware with no hardware reconfiguration; third, co-processor-type interfacing to a host computer; and fourth, hardware simplicity for low cost and compact packaging. In this final report we describe an experimental optical neurocomputer which serves as a testbed for meeting these requirements. The neurocomputer is a nonlinear, highly interconnected, parallel, and analog opto-electronic computer based on real time holography. In the next few sections we discuss some aspects of holography relevant to neural network implementation.

# SECTION 3

# REAL-TIME HOLOGRAPHY FOR NEURAL NETWORKS

In holographic optical neural networks, neurons are represented by pixels on SLMs. The brightness of a pixel corresponds to the activation level of the neuron. By placing the SLM in the back focal plane of a lens and using coherent readout, as shown in Fig. 3, the pixels are converted to coherent beams which illuminate a real-time holographic medium. Weights between neurons are formed when a pair of light beams interfere in the holographic medium, forming a volume sinusoidal light intensity pattern. The photorefractive effect is a suitable physical mechanism for converting this light intensity pattern into a semipermanent deformation of the optical properties of the material, thereby recording the weight values.

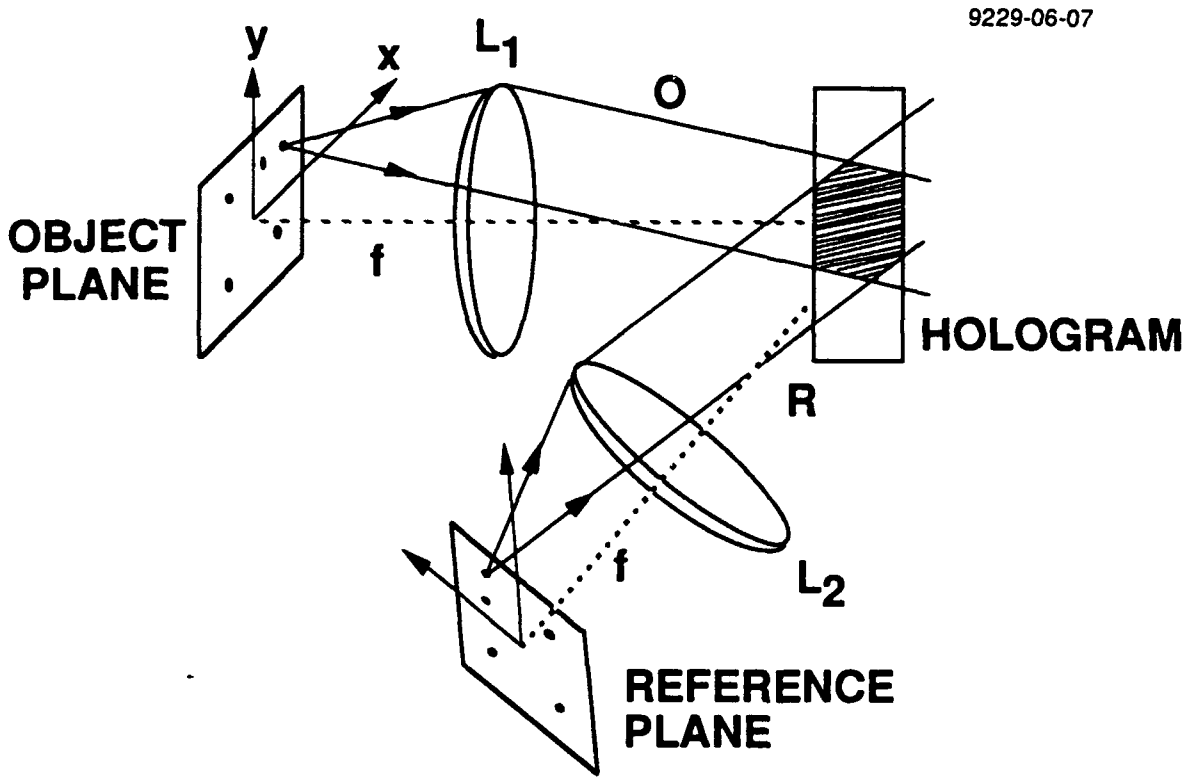



**Fig. 3. Geometry for holographically recording connection weights between neurons.**

In the photorefractive effect incident light excites carriers (electrons and/or holes) from traps into the conduction or valence band. These carriers then are transported by diffusion and drift until they fall into empty traps, creating an internal space-charge field which in turn modulates the

birefringence of the material through the electro-optic effect. This results in a phase grating in the material. Because of the long dark decay times of some of these materials, the phase gratings can be stored with a time constant of many hours. (Storage for longer periods is also possible in some materials using various fixing methods.) When one of the original two beams subsequently addresses the grating, the other beam is reconstructed with a diffraction efficiency that represents the weight value between those two neurons.

The diffraction efficiency of the semi-permanent phase grating represents the connection weight formed between the neurons. It is proportional to the outer-product of the amplitudes of the writing beams and has a fortuitous similarity to the Hebbian learning rules of many neural network models. This equivalence between the outer-product form of the diffraction efficiency and Hebbian learning forms the basis for implementing the weights directly using the analog laws of physics rather than digital representations as in conventional computers. Learning can be implemented in photorefractive optical neural networks since the weights can be selectively increased or decreased. Reading out the grating partially erases it unless the readout beam is much weaker than the original writing light or the crystal is fixed using special techniques.

The angular or Bragg selectivity of a volume photorefractive grating can be very high which results in large storage capacities. (For example, 500 high quality images consisting of 70,000 pixels each have been stored in a single photorefractive crystal.[5]) The Bragg condition states that a beam will be reconstructed only if the angle of incidence of the incident beam relative to the grating is equal to that of the original writing beam. The angular selectivity for reconstruction can be calculated from coupled mode theory[6] and is given by

$$\Delta\theta = \frac{\lambda}{nT_z \sin\varphi}$$

where $T_z$ is the grating thickness, $\lambda$ is the optical wavelength, n is the index of refraction, and $\phi$ is the angle between the two writing beams. Note that the selectivity is greater for thicker crystals. Each individual light beam can be represented by a momentum- or k-vector. (The direction of the k-vector corresponds to the direction of propagation and the magnitude of the k-vector is the inverse of the wavelength.) By using phase matching arguments, the Bragg condition can be described geometrically as a vector sum: $\mathbf{K_j} + \mathbf{K_g} = \mathbf{K_i}$, where $\mathbf{K_j}$ and $\mathbf{K_i}$ are the wavevectors of the incident and diffracted beams, respectively, and $\mathbf{K_g}$ is the grating wavevector.

A geometrical construction for the theoretical storage capacity of a volume hologram can be drawn in k-space, as shown in Fig. 4. If one writing beam varies over solid angle $\theta_O$ while the second writing beam varies over angle $\theta_r$, then the vector difference between the two beams (the grating wavevector $K_g$) will trace out a three-dimensional region in k-space (shaded gray in Fig. 4).

This volume represents the region of k-space that is accessible for storage of information. It will be further limited by the resolution or modulation transfer function (MTF) of the holographic medium. This limit is represented by a sphere centered on the origin whose radius is equal to the largest spatial frequency that can be resolved by the medium. The volume of the accessible region depends on such geometrical factors as the focal lengths of the optics and the spacing of the neurons on the SLMs.

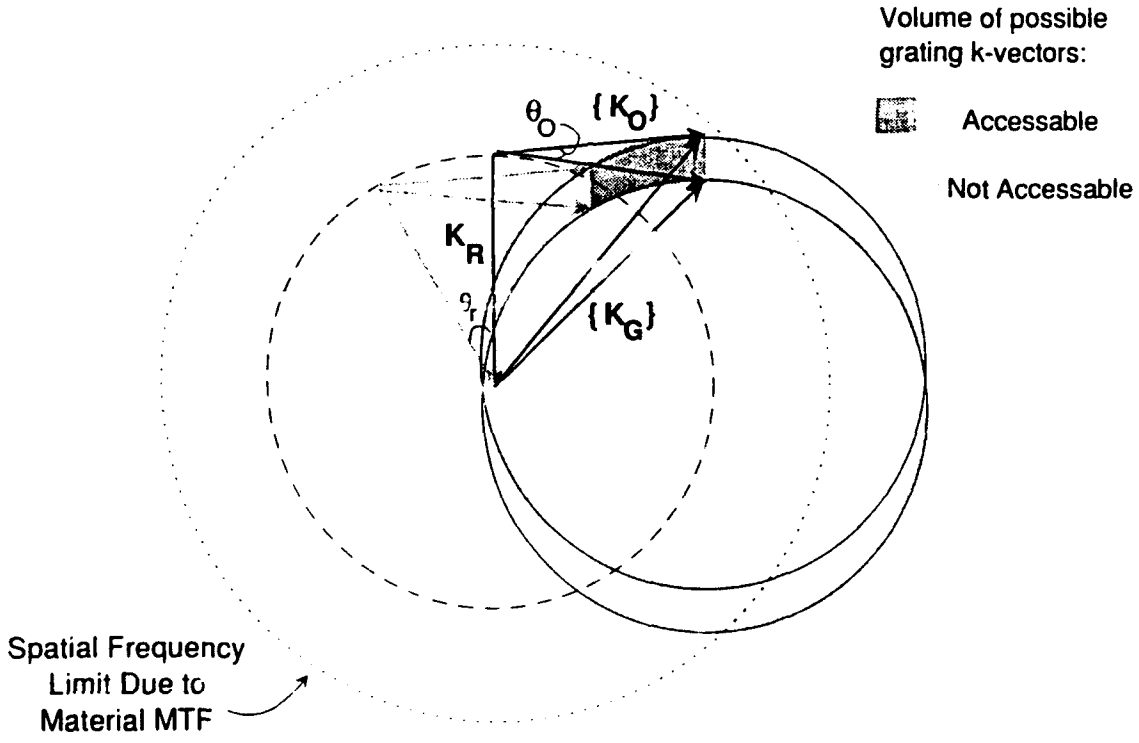



**Fig. 4. K-space diagram of grating k-vectors accessible using recording geometry of Fig. 2.**

The grating wavevector $\mathbf{K_g}$ has an uncertainty volume associated with it due to the finite physical size of the hologram and the nonzero size of the SLM pixels. Dividing the accessible volume of k-space by the volume of the uncertainty volume results in the maximum theoretical number of resolvable gratings which can be stored in the photorefractive crystal. Without going into details here it can be shown that the storage capacity is limited by two upper bounds due to the hologram and neuron dimensions:

$$No.\ of\ gratings \leq \begin{cases} \omega_O \left(\sin\alpha_1^R - \sin\alpha_2^R\right)\left(\dfrac{n_1 V_{holo}}{\lambda}\right)^3 \\ \dfrac{\omega_O}{\sin\theta}\left(\sin\alpha_1^R - \sin\alpha_2^R\right)\left(\dfrac{n_1 f}{d_{neuron}}\right)^3 \end{cases}$$

where $V_{holo}$ is the hologram volume, f is the lens focal length, $d_{neuron}$ is the SLM pixel diameter, $\omega_O$ is the object solid angle, $\alpha_1{}^R$ and $\alpha_2{}^R$ are the angular limits of excursion of the reference beam, $\theta$ is the mean object-reference beam angle, and $n_1$ is the crystal index of refraction. It was assumed in the calculations that led to the above result that the hologram MTF limit is high enough to be ignored. For an active crystal volume of a few cubic mm and reasonable optical parameters the values of the two upper bounds range from $10^{10}$ to $10^{12}$ gratings. This is sufficient to form a fully interconnected network of $10^5$ to $10^6$ neurons. Moreover, each grating can be read out or updated in parallel without the time multiplexing, data contention, or bottleneck problems common in electronic architectures.

The high theoretical storage capacity and parallel access to weights is a direct result of the three-dimensional nature of optical holographic storage. However, other nonfundamental factors also limit the practical performance. The challenge is to construct optical neurocomputers that reduce the difference between actual performance and theoretical limits by attacking these factors.

# SECTION 4
# LIMITATIONS OF CONVENTIONAL HOLOGRAPHY

A variety of mechanisms limit the practical holographic storage capacity of photorefractive crystals to values below the theoretical limit. They include hologram dynamic range, Bragg degeneracy, beam coupling, and laser, hologram, and detector noise. In this section we discuss limitations due to Bragg degeneracy and beam coupling, factors which can be alleviated using cascaded-grating holography.

## 4.1 BRAGG DEGENERACY

One of the most important impediments to holographic neural networks is crosstalk which arises from an effect known as "Bragg degeneracy." The Bragg condition states that the angle of incidence of a light beam relative to a volume grating must match one of the original writing beams in order to form an optical connection. However, even if the angular selectivity is high, crosstalk can still occur. Given a particular grating, it is possible for many light beams to satisfy the Bragg condition for that grating, in addition to the beams which originally wrote the grating. As shown in the momentum space diagram of Fig. 5, a set of beam pairs which define the surfaces of two end-to-end cones all form the same angle with respect to the grating. All of the neuron pairs defined by the cones are connected by that grating even though it was written by only one grating pair. Therefore, a large set of beams other than the original writing beam can scatter constructively from the grating, forming erroneous reconstructions and crosstalk.

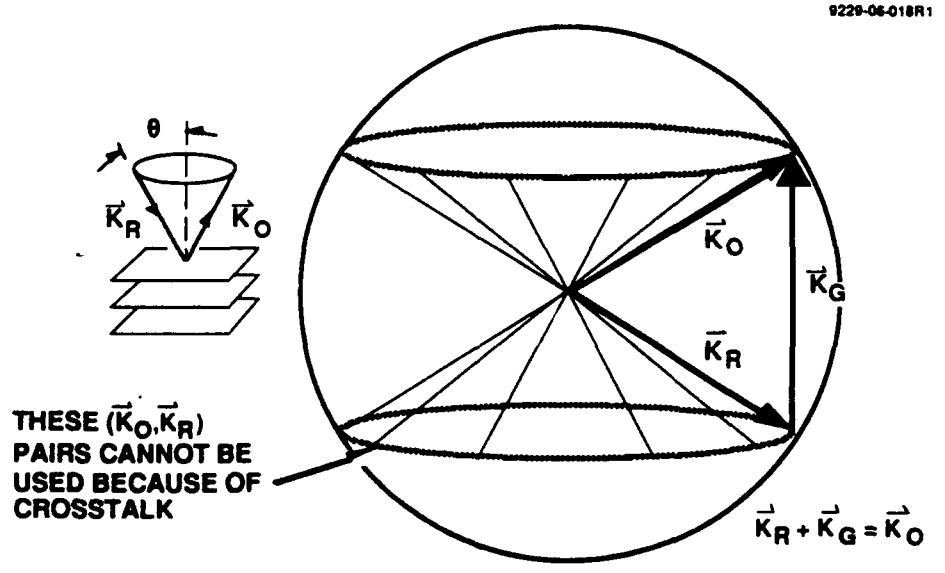
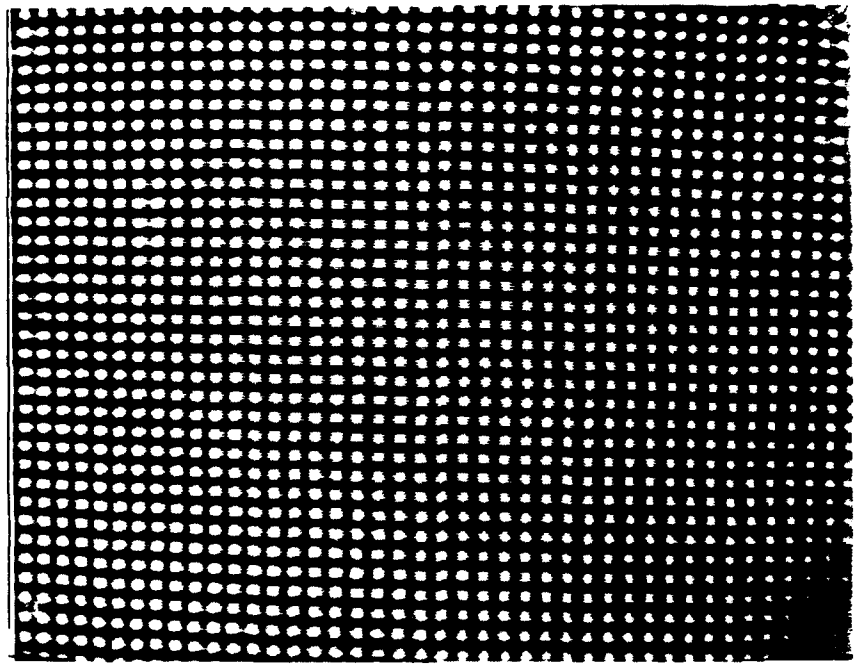



**Fig. 5. K-space diagram of cross-talk resulting from Bragg degeneracy.**

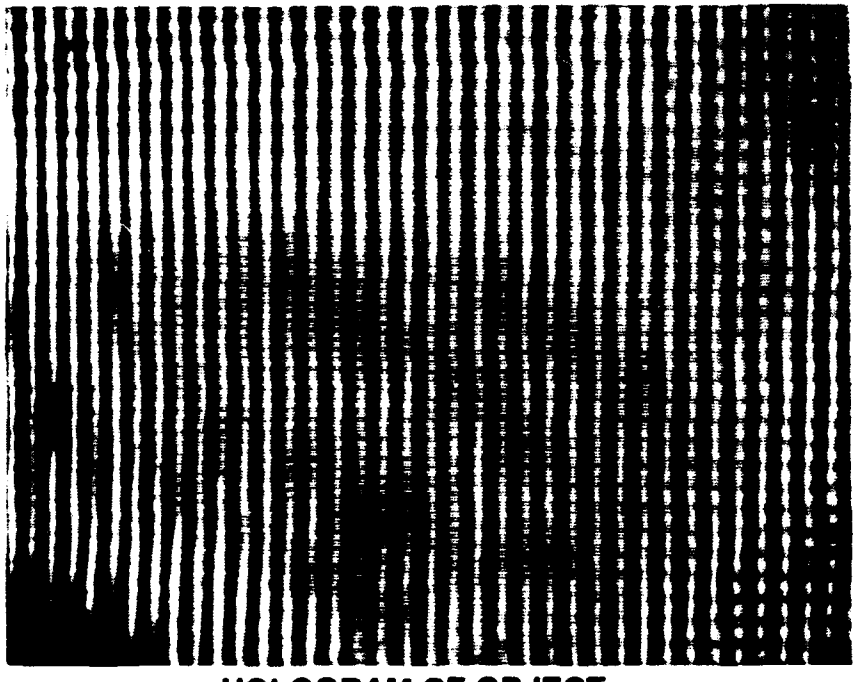
An experimental demonstration of Bragg degeneracy is shown in Fig. 6. We recorded a hologram in a c-cut $BaTiO_3$ crystal using a laser diode light source with a wavelength of 830 nm. (The wavelength and crystal geometry were chosen for low two-wave mixing gain in order to eliminate beam fanning which, as will be shown later in this report, can be used to eliminate Bragg degeneracy.) The experimental configuration of Fig. 3 was used. During recording the object plane consisted of a regular rectangular grid pattern and the reference plane was a uniformly filled rectangle (all pixels on). When the hologram was read out using the uniform reference, the reconstructed object was smeared in the vertical direction. No horizontal smearing can be observed. This can be easily explained by considering the k-space diagram of Fig. 7. In Fig. 7(a). it can be seen that the grating k-vector created by beams R1 and O1 also connects all beams above and below R1 in the R plane. These beams, such as, for example, R2, reconstruct extraneous beams such as O2 which appear above and below the original beam O1. This results in vertical smearing. In Fig. 7(b) we see that horizontally displaced beams are not connected by the same grating k-vector, therefore horizontal smearing is much less. Note that Bragg degeneracy cannot be eliminated by simply phase aberrating the reference beam since the above construction would still hold for the individual plane wave components.

Possible approaches for avoiding Bragg degeneracy are subsampling of the SLMs[7] and spatial multiplexing of holograms.[8] In the subsampling approach, neurons are arranged in sparse nonredundant patterns on the SLMs, and output planes are similarly sparsely sampled; thus although false reconstructions still occur, they occur at unused positions and do not contribute to the output. The special patterns can consist of so-called "fractal" lattices or other sparse patterns. If the SLMs are capable of displaying NxN neurons, then this approach can implement a total of $N^{3/2}$ neurons and $N^3$ weights. This has the pleasing quality that the storage capacity of the crystal and the number of weights required to fully interconnect the neurons on the sampled SLMs have the same dimensional scaling. In many practical cases, however, it also has the drawback that the storage capacity may be limited by the number of neurons that can be displayed in sparse patterns on the SLM, rather than by the potentially large capacity of the crystal. As discussed above, the storage capacity of a 1 $cm^3$ crystal should be sufficient to store the interconnections for a NxN array of neurons where N=500, which matches the capabilities of present-day SLMs. However, because of the subsampling, only $N^{3/2}$ neurons can be implemented even though the SLM is capable of displaying $N^2$ neurons. Since N=500, the neuron and weight storage capacities are reduced by factors of 22 and 500, respectively, from the theoretical maximums. The light efficiency is also lowered because some of the light is diffracted to unused pixels as a result of the Bragg degeneracy.



OBJECT

(a)



HOLOGRAM OF OBJECT

(b)

Fig. 6. Experimental demonstration of Bragg degeneracy using infrared hologram recorded in c-cut $BaTiO_3$. Vertical smearing is a sign of Bragg degeneracy.

The spatial multiplexing approach avoids the Bragg degeneracy problem by physically dividing the crystal into separate volumes for each weight. However, this reduces the coupling efficiency of gratings, reduces parallelism because sequential exposures must be used, and increases hardware complexity.

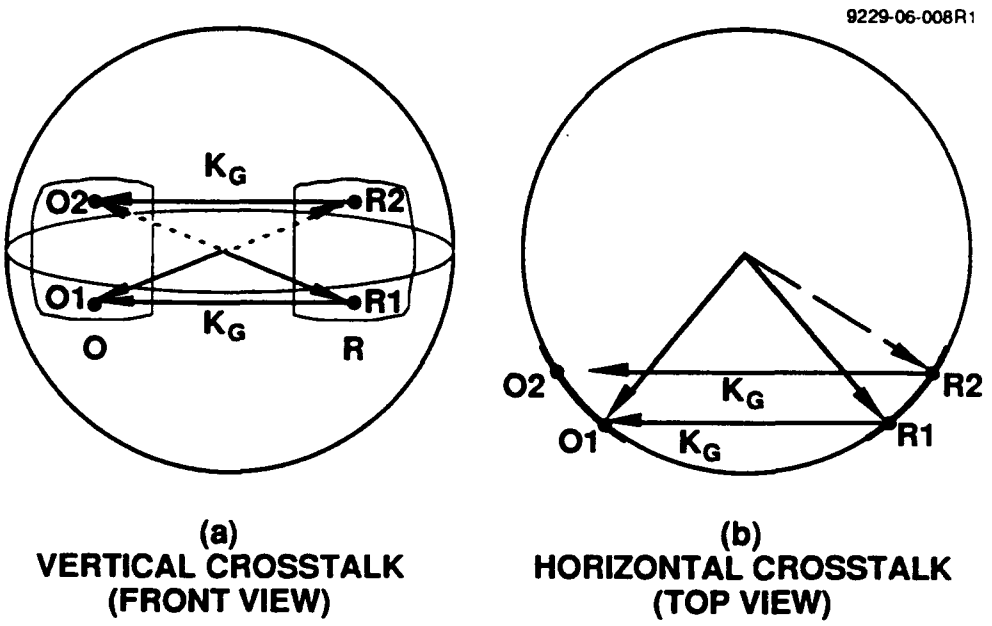



**Fig. 7. K-space diagram showing origin of vertical smearing effect in Bragg degeneracy. Vertically-displaced pairs of reference-object points are all connected by the same grating k-vector.**

## 4.2 ANGULAR WIDTH BRAGG DEGENERACY

In this section we discuss a previously unidentified effect related to the Bragg degeneracy that can occur in holographic interconnection systems. This effect results in further unintended Bragg degeneracies due to the angular width of the Bragg diffraction responses of volume holographic gratings. We hereafter refer to this effect as *angular width Bragg degeneracy*. We define it more precisely in Section 1.3. The main practical result of our investigations is that one of two design rules for developing fractal sampling grids (that are used to avoid the effects of Bragg degeneracy) does not always work; i.e., while this rule can be used to avoid direct Bragg degeneracy crosstalk, it does not necessarily eliminate angular width Bragg degeneracy and its associated crosstalk. For holographic interconnection systems that utilize fractal sampling grids to avoid Bragg degeneracy it is thus important to be aware of the conditions under which angular width Bragg degeneracy occurs and to use the correct fractal sampling grid design rule to avoid its presence in the desired interconnections.

In Section 4.2.1 we briefly review Bragg degeneracy and its effects on interconnection systems. We discuss the two design rules that have been proposed to develop fractal sampling grids. In Section 4.2.2 we describe angular width Bragg degeneracy in more detail and illustrate its effects using numerical modeling examples based on a three dimensional, multiple grating version[9] of the optical beam propagation method (BPM).[10] Our findings are summarized in Section 4.2.3, and the relationship of angular width Bragg degeneracy to the fundamental interconnection capacity of holographic media is briefly addressed.

### 4.2.1 Bragg Degeneracy and Fractal Sampling Grids

As discussed in Section 4.1, Bragg degeneracy occurs when a grating that connects a desired pair of pixels also inadvertently connects one or more additional pairs of pixels due to the two cones of angles that are Bragg matched to that particular grating.[11] One method of visualizing which pixels in the interconnection system's input and output planes have degenerate interconnections is to project the cones of Bragg-matched angles onto the pixel planes.[12] As is illustrated in Fig. 8, for a given grating this projection results in two lines, one on each pixel plane. The two degeneracy lines shown in the figure correspond to the grating connecting the top-most pixel pair (joined by a solid line). Due to Bragg degeneracy, this grating also connects the pair of pixels joined by the dashed line since these pixels are on the degeneracy lines.

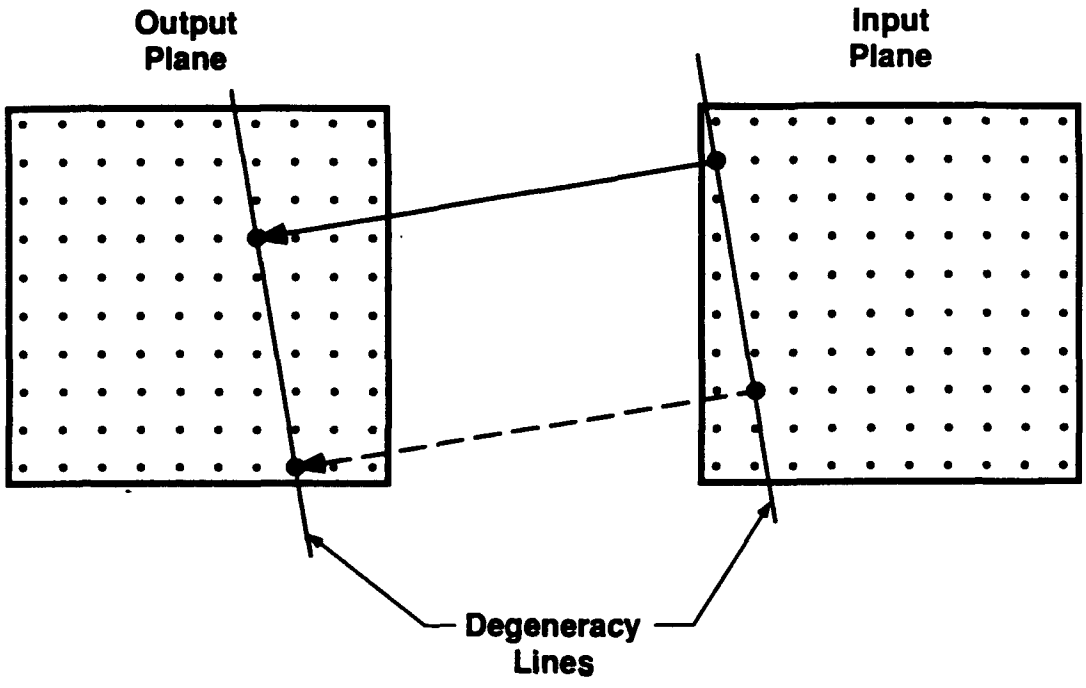



**Fig. 8. Bragg degeneracy causes a desired interconnection (such as between the pixels joined by the solid line) to inadvertently connect other pixel pairs (such as the pixels joined by the dashed line) that are on the degeneracy lines. Such pixel pairs, together with the original pixel pair, form the vertices of a rectangle.**

As has been generally recognized, Bragg degeneracy can be a significant source of crosstalk in holographic interconnection systems that superimpose all of the desired interconnection gratings within the same volume of material. Bragg degeneracy can seriously affect the fidelity of holographic interconnections in at least two ways: 1) the recording of a desired grating inadvertently connects undesired pairs of pixels, and 2) if several pairs of pixels are connected by gratings that are degenerate then the actual weighted interconnection between each of the pairs is the sum of the complex (magnitude and phase) grating amplitudes. In the first case, completely undesired interconnections are formed, and in the second case desired interconnections have incorrect weights. For holographic interconnection systems in which Bragg degeneracy can occur, it is thus important to eliminate its effects. The use of so-called fractal sampling grids for the input and output planes has often been suggested for this purpose.

The use of fractal sampling grids involves creating patterns of pixels on the input and output planes that can be interconnected only by non-degenerate gratings. The direct crosstalk due to Bragg degeneracy is thus avoided, but at the cost of reduced spatial sampling of the input and output plane pixelation. For example, if an interconnection system is implemented using two-dimensional pixel planes that are each composed of $M^2$ pixels arranged as an evenly spaced M by M array, the largest number of usable pixels in each plane is $M^{3/2}$ for an N-to-N interconnection.

Several design rules have been proposed[13] to generate fractal sampling grids. These can be summarized as the "no rectangle rule" and the "unequal row spacing rule." Each rule is claimed to be a sufficient condition for avoiding undesired crosstalk due to Bragg degeneracy.

The rectangle rule can be understood as follows. Note in Fig. 8 that when two pairs of pixels (each pixel pair consisting of one pixel in the input plane and one pixel in the output plane) are connected by degenerate gratings, the pixel pairs form the vertices of a rectangle. Use of the no rectangle rule consists of generating patterns of pixels for the input and output planes such that no rectangles can be formed having two pixels in the input plane and two pixels in the output plane as vertices. An example of a set of input and output plane grid patterns that satisfies the no rectangle rule is shown in Fig. 9(a) for a 27-to-27 interconnection system utilizing 9 by 9 pixel arrays. The large and small dots refer to used and unused pixel sites, respectively.

An example set of grid patterns generated using the unequal row spacing rule is shown in Fig. 9(b). The operating principle here is to select rows of pixels that are vertically separated by unequal numbers of pixels in the input and output planes. In the example shown in Fig. 9(b), the rows in the input plane are separated by either 4 or 8 pixels, while the row separation in the output plane is 3 or 6 pixels. As we shall see below, fractal sampling grids generated with the unequal row spacing rule automatically avoid angular width Bragg degeneracy as well as normal Bragg degeneracy. This, however, is not true for grids generated with the no rectangle rule.

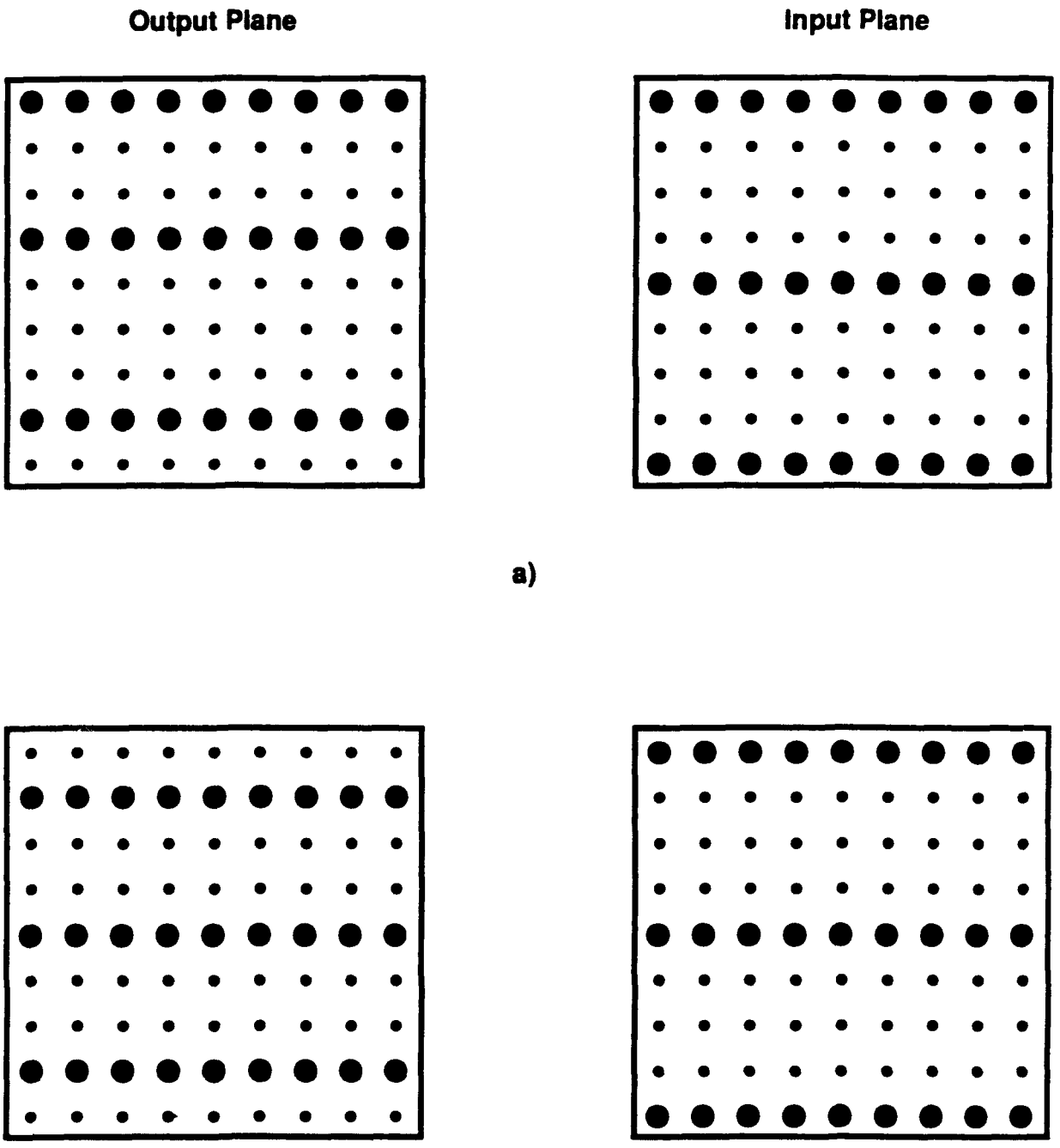



Fig 9. Two possible fractal sampling grids for a 27-27 interconnection system. (a) A fractal sampling grid that satisfies the no rectangle rule, but not the unequal row separation rule. (b) A fractal sampling grid designed using the unequal row separation rule.

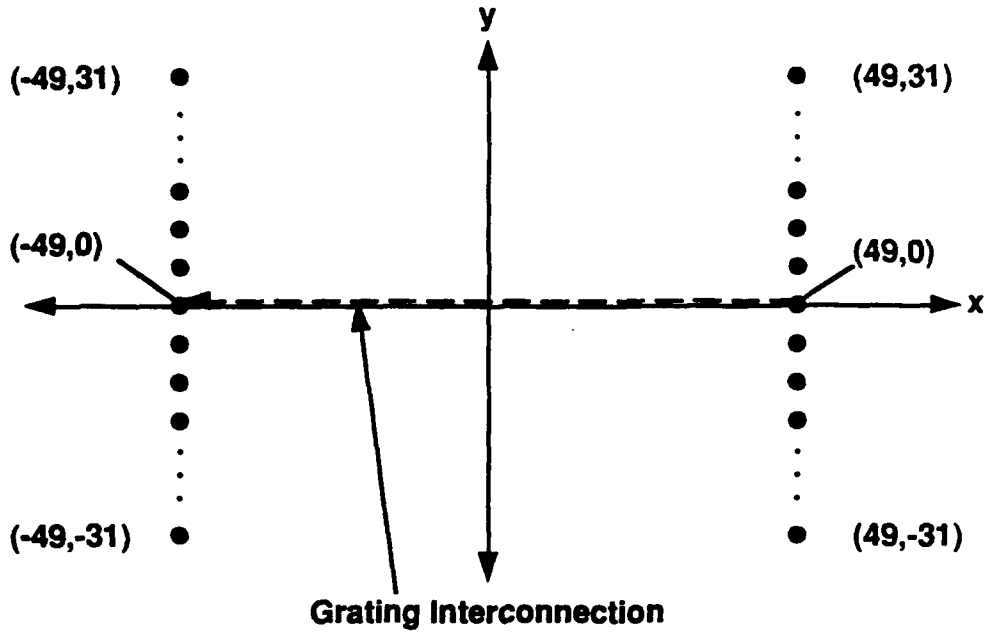
### 4.2.2 Definition of Angular Width Bragg Degeneracy

In this section we explain more clearly the nature of angular width Bragg degeneracy, how it arises, its effect on weighted interconnections, and how it may be avoided. In the literature, Bragg degeneracy is described only in terms of exact angular Bragg-alignment of undesired pixel pairs to a degenerate grating. Such a description neglects an important point: the angular sensitivity of a volume grating's diffraction response has a finite angular width about its Bragg angles. The angular FWHM of this diffraction response, $\Delta\theta$, is approximately $\Lambda/D$ (measured in the holographic medium) in which $\Lambda$ is the period of the grating and $D$ is the optical path length of the readout beam through the medium. The possibility therefore exists for an interconnection grating to inadvertently connect an unintended pair of pixels if each pixel is within an angular range of approximately $\Delta\theta$ of the grating's Bragg angles. This is what we mean by angular width Bragg degeneracy.

We illustrate this effect and compare it to normal (Bragg-matched) Bragg degeneracy using the following pair of examples. Consider two columns of pixels as shown in Figs. 10(a) and 10(b). Let the right column in each figure be in the input plane of a holographic interconnection system and the left column in the output plane. In each case, a single sinusoidal phase grating is present. In Fig. 10a, the grating is intended to connect the pixels labeled (49,0) and (-49,0) in the input and output planes, respectively. In Fig. 10(b), the grating connects pixels (49,1) and (-49,0). By turning on the 63 pixels in the right hand column and monitoring the diffracted light at each of the 63 pixels in the left hand column, we can evaluate the amount of crosstalk in each case caused by the presence of a single grating.

In the case of Fig. 10a, we naturally expect each pixel in the input plane to be connected to its opposite pixel in the output plane since the interconnection grating is degenerate to all of these interconnections. This is illustrated in Fig. 11(a) in which the expected diffracted outputs of the output plane pixels are shown as a function of the pixel index [each diffracted output is assumed to have 1% diffraction efficiency for consistency with the numerical simulation results discussed below and shown in Fig. 11(b)]. As shown in Fig. 11(c), we expect the grating of Fig. 10(b) to yield a diffracted output only at pixel (49,0) in the output plane when all 63 input plane pixels are turned on. Since no other pairs of pixels lie on the lines of degeneracy for this interconnection grating, no other diffracted outputs should be present.

We tested the above predictions by using a three dimensional version of BPM to propagate the 63 readout beams through a holographic medium in which the desired interconnection grating was recorded. We assumed an angular separation of 18 degrees between the pixel columns, a readout wavelength of 514 nm, and a 5.0 mm thick holographic medium having a refractive index of 2.48. The pixel separation in the input and output planes was taken to be 0.97 mm. The focal length of the collimating lens between each pixel plane and the holographic medium was assumed




a) Grating interconnection between pixels (49, 0) and (-49,0)

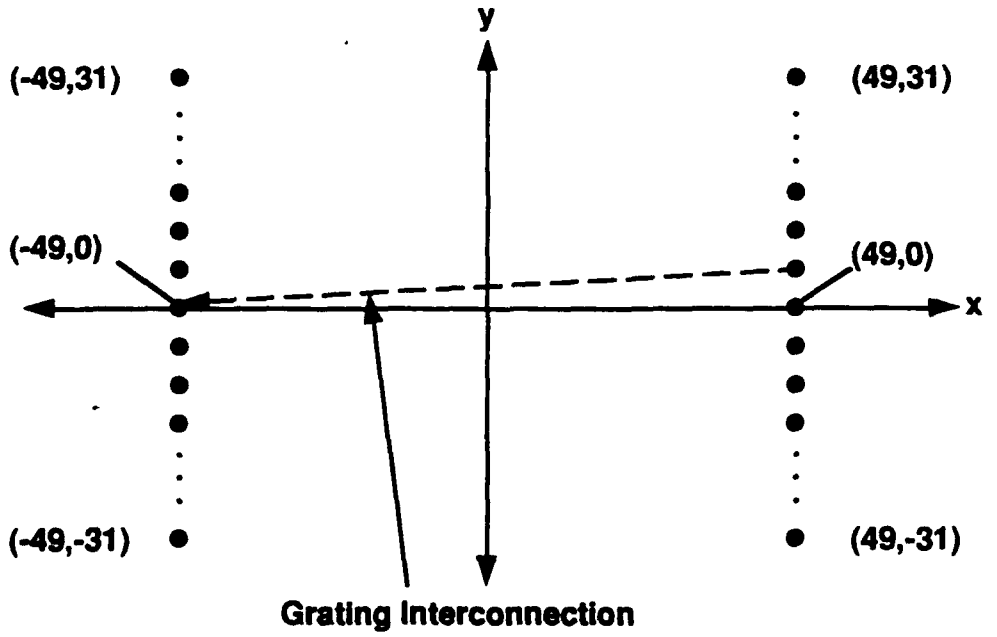



Fig. 10. (a) Grating interconnection between pixels (49,0) and (–49,0). (b) Grating interconnection between pixels (49,1) and (–49,0).

to be 300 mm. The grating strength of the interconnection grating was designed to yield 1% diffraction efficiency for a single input to a single output.

Simulation results are shown in Figs. 11(b) and 11(d). As seen in Fig. 11(b), Bragg degeneracy (the case of Fig. 10a) does indeed cause each input plane pixel to be connected to its opposite output plane pixel, and the simulation matches our expected results [Fig. 11(a)]. However, as seen by comparing Figs. 11(c) and 11(d), there are more diffracted outputs in the simulation results than expected for the interconnection grating shown in Fig. 10b. This is due to angular width Bragg degeneracy, which causes pixel (49,i) in the input plane to be connected to pixel (-49,i-1) in the output plane. For example, even though pixels (49,0) and (-49,-1) are not exactly on the degeneracy lines of the interconnection grating, they are close enough to be well within $\Delta\theta$ of the corresponding Bragg angles of the grating and therefore are connected by the grating. The efficiency of the interconnection depends on how angularly detuned a given pixel pair is from the grating. This is seen in Fig. 11(d) in which the light detected at the output pixels is shaped like the angular sensitivity of the interconnection grating.

One can summarize angular width Bragg degeneracy by noting that the degeneracy lines shown in Fig. 8 are not infinitesimally thin. Rather (as seen in Fig. 12), they have a finite thickness that is dependent on the angular width of the interconnection grating to which they correspond. This can result in undesired interconnections for fractal sampling grids designed with the no rectangle rule. In the next section we illustrate the effects of angular width Bragg degeneracy on the reconstruction fidelity of weighted interconnections.

### 4.2.3 Effects on Weighted Interconnections

Angular width Bragg degeneracy can have just as severe an effect on the reconstruction fidelity of weighted interconnections as normal (Bragg-matched) Bragg degeneracy. In this section we discuss a set of numerical simulations of 9-to-9 weighted interconnections performed using BPM. By comparing the RMS error of the normalized diffracted outputs, we quantitatively examine the fidelity performance of the 9-to-9 holographic interconnection system in the presence (among the desired interconnections) of normal Bragg degeneracy (Case A), angular width Bragg degeneracy (Case B), and no form of Bragg degeneracy (Case C).

In our BPM simulations, the same assumptions apply as for the simulations in Section 1.3.1 except for the following additions. First, the holographic medium is assumed to have a linear recording characteristic relative to the incident write beam intensity. Second, only the desired interconnection gratings are assumed to be present in the medium; no cross gratings exist between pixels within the input or output planes (as would be created using a simultaneous recording method, for example).

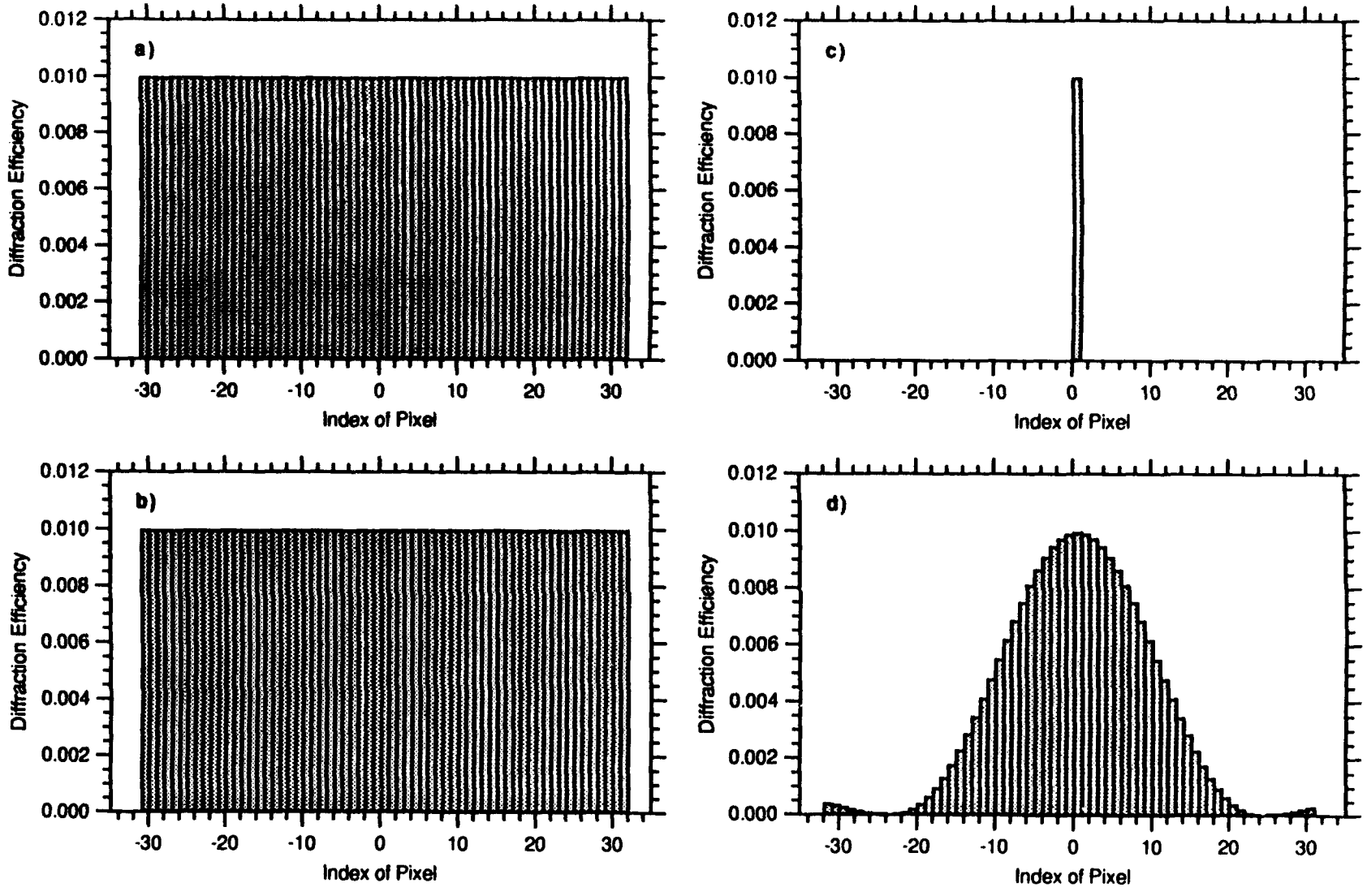



Fig. 11. Simulation results for single grating interconnections. (a) Expected diffracted outputs, Case 1. (b) Obtained diffracted outputs, Case 1. (c) Expected diffracted outputs, Case 2. (d) Obtained diffracted outputs, Case 2.

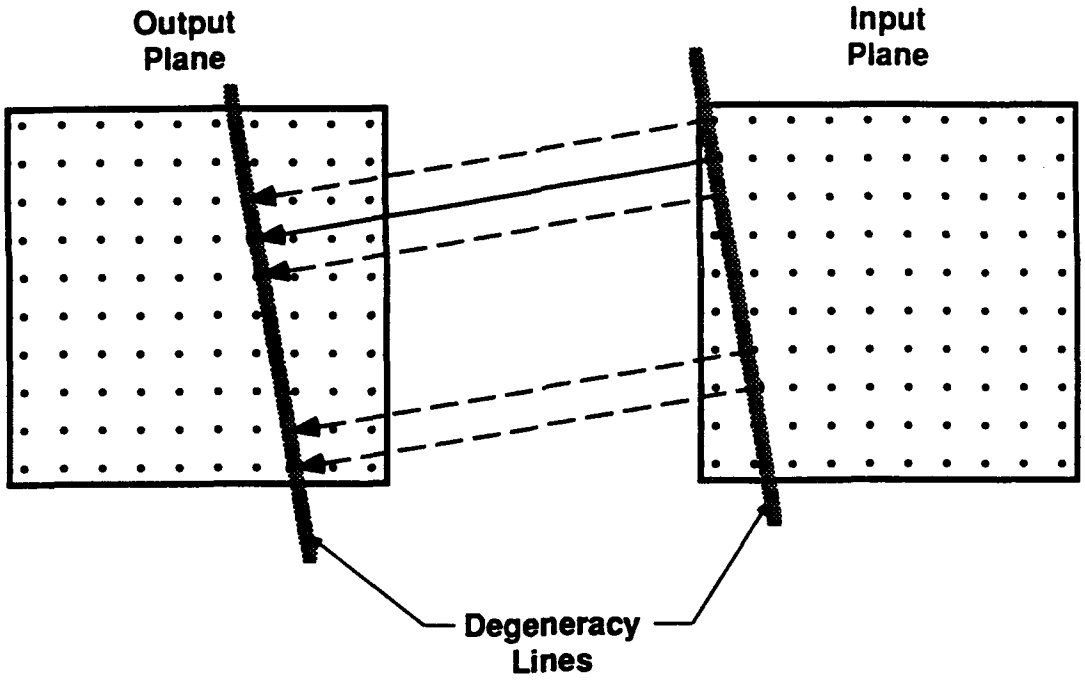



**Fig. 12. Degeneracy lines have finite thickness, corresponding to the Bragg width of a given interconnection grating. This results in additional degenerate interconnections (compare dashed interconnection lines above with Fig. 1).**

The pixel geometry for Case A is shown in Fig. 13(a). The pixels in both the input and output planes are each arranged on a regular 3 by 3 grid, and thus significant Bragg degeneracy is present in the interconnection system. The weights are the same between all of the input plane pixels and a given pixel in the output plane. The relative weights for the top row of pixels in the output plane (left to right) are 0.8, 0.5, and 0.2. Similarly, the relative weights for the second and third rows are 0.9, 0.6, 0.3, and 1.0, 0.7, 0.4, respectively. All nine input plane pixels are used during readout, each generating a unit intensity plane wave at the front face of the holographic medium. In the absence of fidelity errors, the diffracted outputs at each pixel should therefore have the same relative detected power as the weight corresponding to that pixel. The ideal diffracted outputs for Case A are shown in Fig. 14(a).

Simulation results for Case A are shown in Fig. 14(b). Since presumably only the diffracted outputs at the nine original output plane pixels are of interest, these are shown in Fig. 14(c). Fig. 14c therefore represents the actual detected outputs (assuming that detectors are placed only at the desired output pixels), while Fig. 14(b) represents all of the diffracted outputs generated by the interconnection system. Those outputs that are present in Fig. 14(b) but not in 14(c) (for example,

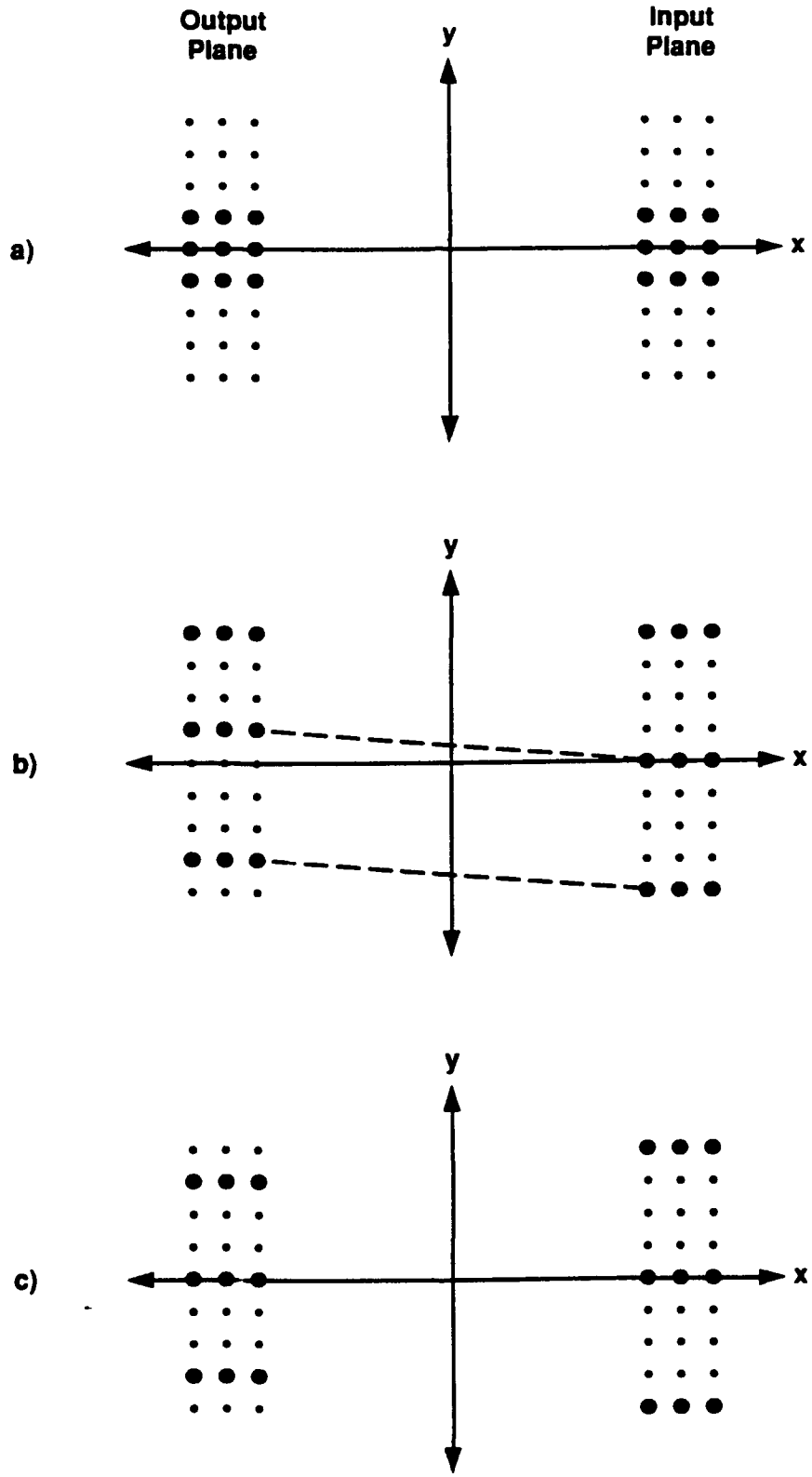



Fig. 13. Fractal sampling grids used for weighted 9-to-9 interconnection simulations. The grids for Cases A, B, and C, and shown in (a), (b), and (c), respectively.

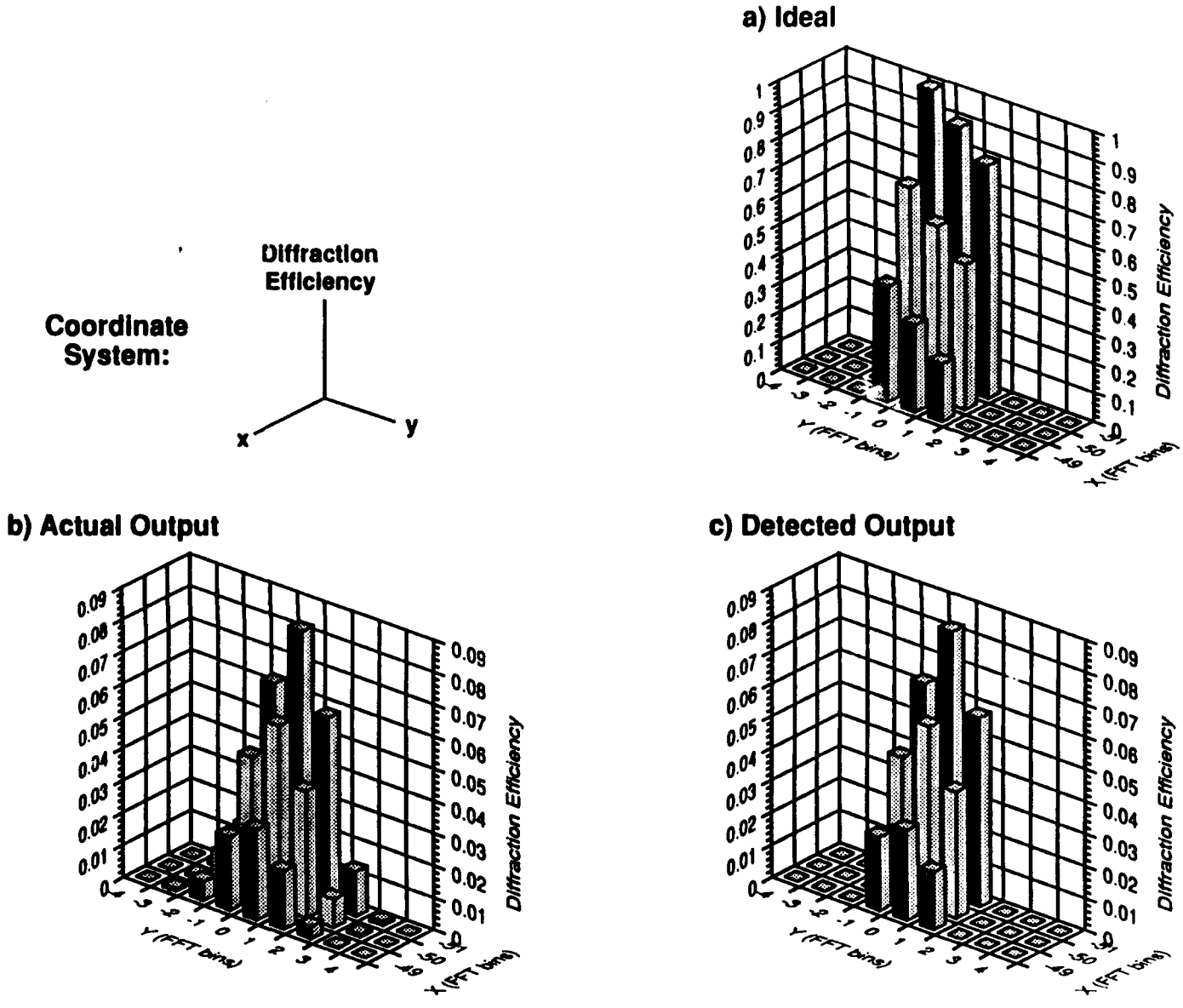



Fig. 14. Simulation results for Case A (see text).

at pixels (-49, 2), (-50,3), and (-51,2)) are due to Bragg degeneracy. They naturally do not directly affect the detected outputs.

As seen by comparing in Figs. 14(a) and 14(c), there are significant visible differences between the ideal and actual outputs of the interconnection system. The relative magnitudes of the detected outputs are obviously incorrect. One quantitative measure of the amount of error in the detected outputs is the RMS error of the normalized outputs. If the ideal outputs are represented by the vector $\boldsymbol{y}$ (each component of which is the ideal magnitude of one of the nine pixels in the output plane) and the actual diffracted outputs by the vector $\boldsymbol{y}'$ (each component being the detected output at the pixel corresponding to the same component in $\boldsymbol{y}$), the RMS error of the normalized outputs, $\varepsilon$, is

$$\varepsilon = \sqrt{\sum_{i=1}^{N}\left(\frac{y_i}{|\mathbf{y}|} - \frac{y_i'}{|\mathbf{y}'|}\right)^2}$$

in which $N = 9$. The maximum value of this error is $\sqrt{2}$. We have found that values of 0.1 correspond to large errors in the relative magnitude of individual outputs (up to 25% or more). The RMS error for Case A is 0.173, which [as seen in Fig. 14(c)] represents a significant deviation from the desired outputs. As will be seen upon examining Case C (in which there is no Bragg degeneracy), most of the error in Case A is due to Bragg degeneracy.

The pixel geometry for Case B is shown in Fig. 13(b). Note that it is a subset of the fractal sampling grid shown in Fig. 9(a) that was generated using the no rectangle rule. The same relative weights and readout intensities as for Case A are assumed for the BPM simulation of Case B. The ideal outputs are shown in Fig. 15(a), the diffracted outputs in Fig. 15(b), and the detected outputs in Fig. 15(c). At first glance, the detected outputs appear to be similar to the ideal outputs. However, closer examination reveals that the pixels having a y-axis index of 4 are nearly a factor of two too small compared to the other pixels. Indeed, the RMS error for this case is 0.182–slightly larger than Case A's!

This poor fidelity performance is due to angular width Bragg degeneracy. As an example, consider the two interconnections indicated by the dashed lines in Fig. 13(b). Although the four pixels involved do not form the vertices of a rectangle, a simple calculation shows that the upper set of pixels is well within $\Delta\theta$ of the grating connecting the lower set of pixels, and vice versa. Thus, angular width Bragg degeneracy is present among these interconnections. Similarly, the interconnections between each pixel in the middle row of the input plane to each pixel of the middle row of the output plane are degenerate with the corresponding interconnections between pixels in

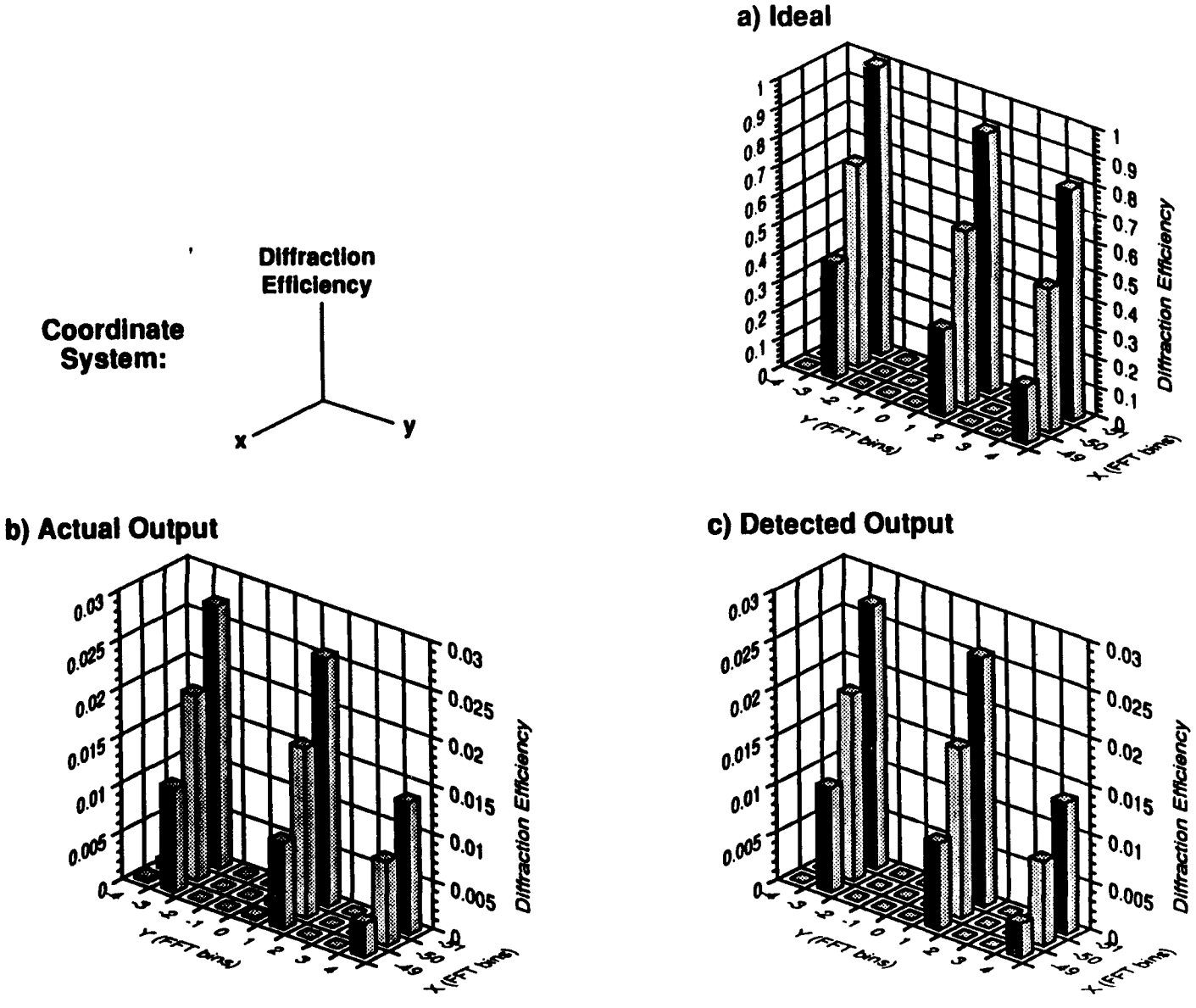



Fig. 15. Simulation results for Case B (see text).

the bottom rows. This angular width Bragg degeneracy is the reason that the detected outputs of the pixels in Fig. 14(c) along rows 1 and -3 are nearly twice as large as those along row 4.

Note that angular width Bragg degeneracy occurs in a pixel geometry that satisfies the no rectangle rule. This result indicates that the no rectangle rule is not a sufficient condition for the complete avoidance of any form of Bragg degeneracy. This is in direct contradiction to References 12 and 13.

The pixel geometry for Case C is shown in Fig. 13(c). It is a subset of the fractal sampling grid generated using the unequal row separation rule shown in Fig. 9(b). Again, the relative weights and readout beam intensities are the same as above. The ideal outputs are shown in Fig. 16(a). The diffracted actual outputs from the BPM simulation are shown in Fig. 16(b). Note that extraneous diffraction outputs are visible in rows -4, -1, 1, and 4. Angular width Bragg degeneracy alone is responsible for the outputs in rows -1 and 1, while a combination of angular width and normal (Bragg-matched) Bragg degeneracy causes the outputs in rows -4 and 4. Since the outputs in these rows are not detected they naturally do not directly affect the fidelity of the detected outputs.

The detected outputs are shown in Fig. 16(c). A visual comparison with the ideal outputs in Fig. 16(a) shows significantly improved fidelity over the previous cases. This is verified by comparing the RMS error of the normalized outputs, which for Case C is 0.0129. This is over an order of magnitude lower than the errors obtained for Cases A and B. Selection of a pixel geometry that eliminates all forms of Bragg degeneracy among the desired interconnections thus significantly improves the holographic interconnection system's reconstruction fidelity. The unequal row separation rule allows the design of such pixel sampling grids.

In summary, we have demonstrated a previously unrecognized form of Bragg degeneracy that is due to the angular width of the Bragg diffraction responses of volume holographic gratings. As shown by numerical simulation, this angular width Bragg degeneracy can degrade interconnection fidelity as much as normal (Bragg-matched) Bragg degeneracy. Use of the no rectangle rule to design fractal sampling grids does not guarantee the avoidance of angular width Bragg degeneracy among the desired interconnections when implementing a given interconnection system. However, all forms of Bragg degeneracy among the desired interconnections can be avoided if the unequal row separation rule is used in the design of the fractal sampling grids. This is therefore the preferred design rule to use with holographic interconnection systems in which Bragg degeneracy can occur.

Having explored some of the practical aspects of angular width Bragg degeneracy, we turn now to a brief discussion of a more fundamental issue; namely, does this form of Bragg degeneracy impose any additional limitations on the interconnection capacity of volume holograms? The answer at this point appears to be no. As discussed in Section 3, one can calculate the amount of

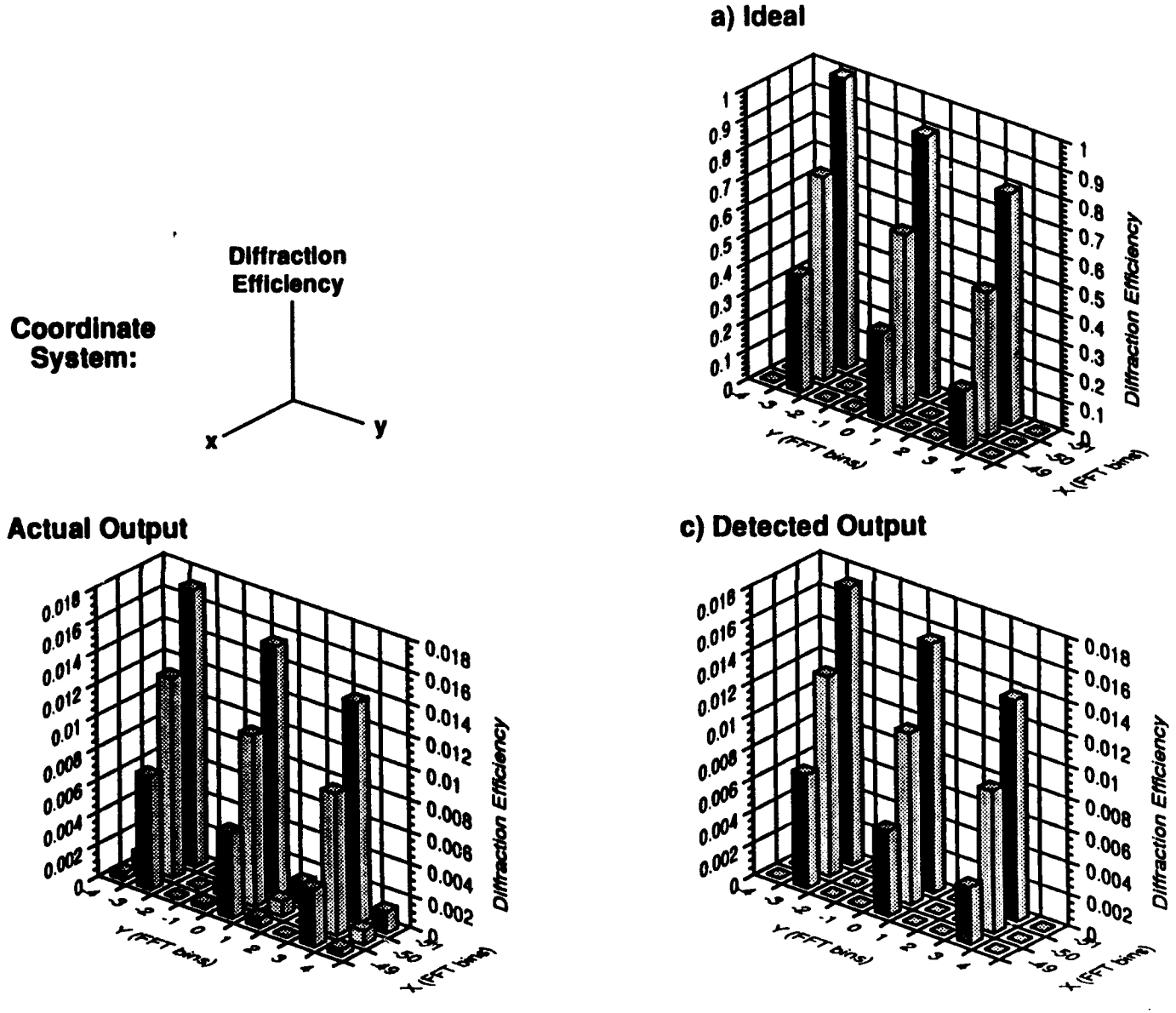



Fig. 16. Simulation results for Case C (see text).

grating k-space that is accessible from two finite pixel planes. Based on the dimensions of the holographic medium, a given grating occupies a fixed volume of this k-space. The size of each grating's k-space volume corresponds inversely to the angular width of the grating's Bragg response. One can calculate the number of interconnection gratings that can be stored in the accessible volume of k-space by simply dividing by the volume occupied by an individual grating. Angular width Bragg degeneracy does not alter this fundamental fact. Rather, it relates to how specific sampling grids are chosen such that the resultant interconnection gratings do not overlap in the accessible grating k-space. Awareness of angular width Bragg degeneracy is thus important in the *implementation* of an interconnection geometry, but does not affect the fundamental number of interconnections that the geometry can intrinsically access.

## 4.3 BEAM COUPLING

A second major source of distortion in holograms is energy transfer between Bragg-matched beams via two-wave mixing. This occurs because when an input beam is Bragg-matched to a grating, the reconstructed beam is automatically Bragg-matched. This beam can then itself read out the same grating and reconstruct the input beam. If the gain-length product of the grating is sufficiently high, energy transfer or beam coupling can occur between the beams, resulting in distortions. Several workers have analyzed the effects of beam coupling on holographic image quality.[14,15] They found that distortion increases rapidly as the gain-length product of the grating increases, as illustrated in Fig. 17. Therefore, a straightforward technique for reducing beam-coupling distortion is to reduce the gain-length product. However, this also results in low diffraction efficiencies and low signal-to-noise ratios.

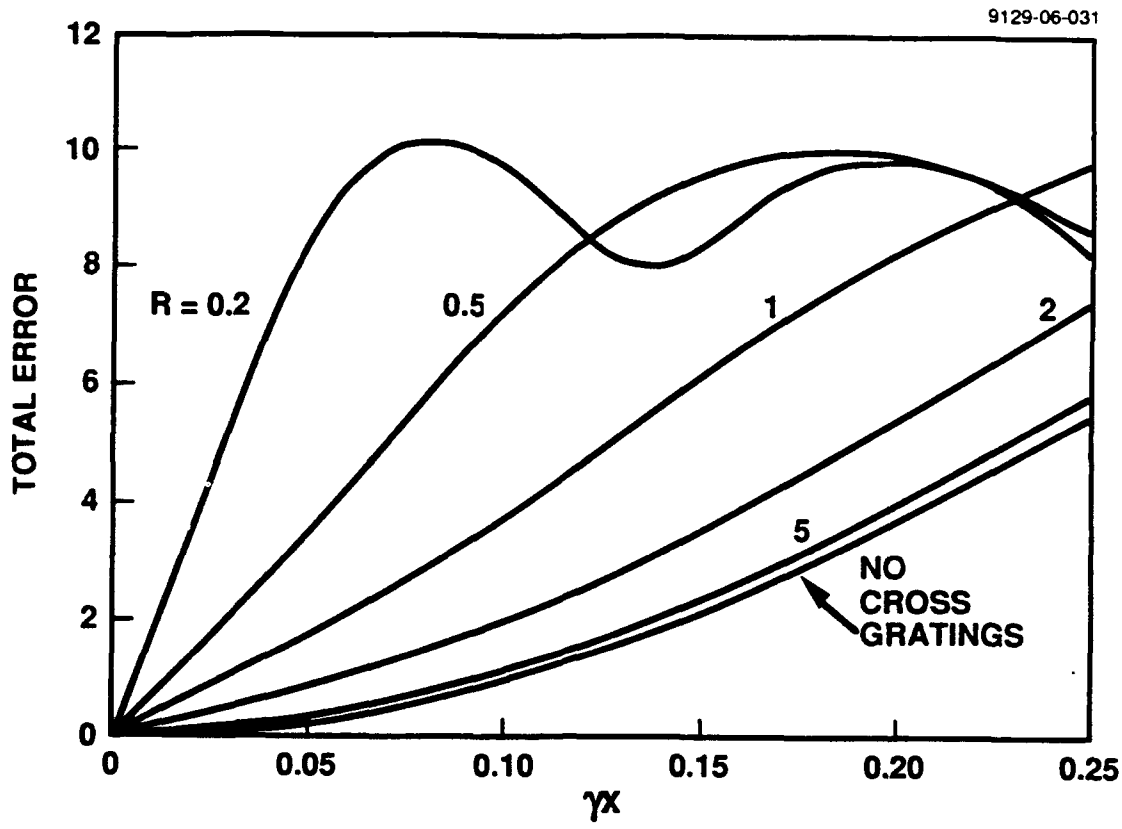


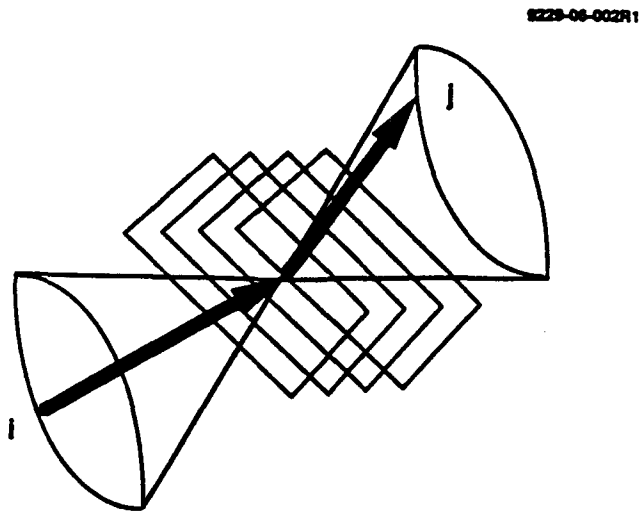

Figure 17. Cross-coupling error in superimposed volume gratings as a function of grating-strength x interaction length.

# SECTION 5
# CASCADED-GRATING HOLOGRAPHY

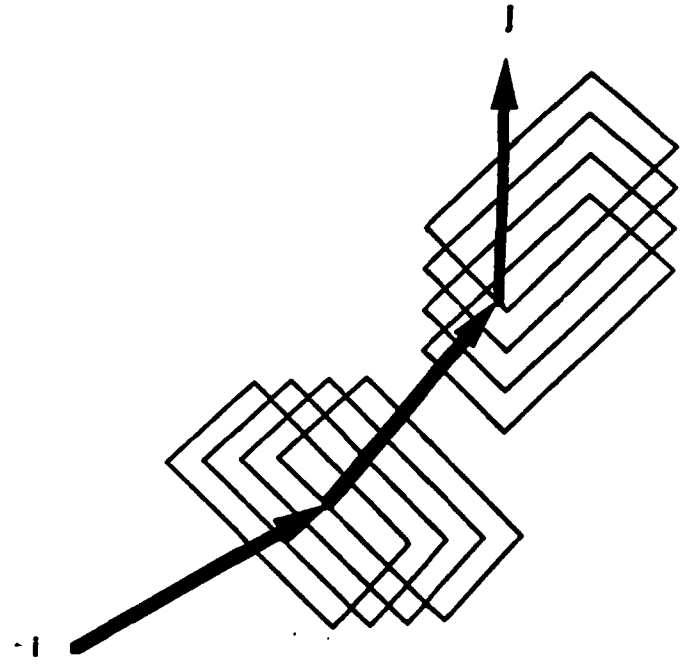
## 5.1 CASCADED-GRATING WEIGHT STORAGE

We have developed a holographic recording technique called cascaded grating holography (CGH) which greatly reduces the distortions discussed in the previous section. The essence of the CGH idea is to use a set of angularly and spatially multiplexed gratings to store each weight rather than a single grating. By forcing a light beam to match the Bragg condition at each of a cascaded series of spatially and angularly distributed gratings (Fig. 18), crosstalk due to Bragg degeneracy is greatly reduced. From the k-space construction of Fig. 19, it is clear that two gratings in series will connect only a single input/output pair of beams via an intermediary diffracted beam. All other beam triplets (input, intermediary, and output beams) will not be able to match the Bragg conditions at both gratings because the intermediary diffracted beam will not lay on the Bragg degeneracy cone of the second grating. An undesired beam on the Bragg degeneracy cone of one grating is therefore rejected by the remaining gratings. This allows the neurons to be arranged in arbitrary patterns on the SLM, increasing the storage capacity (since all pixels can be used) and throughput.

We have investigated two techniques for generating such multiple-grating connection weights in photorefractive materials: the self-pumped phase conjugate mirror (PCM) and forward-scattering beam fanning. A self-pumped PCM[16] generates the "phase conjugate" or time-reversed version of an image-bearing coherent input beam.[17] In the early stages of our research we conjectured that the temporal evolution of the phase conjugation process in self-pumped photorefractive PCMs, which include dynamic multiple-wave mixing terms, will be similar to the single grating case in that the amplitude diffraction efficiency of the conjugate beams will increase with the outer-product of the writing beam amplitudes, thereby emulating the Hebbian learning rule. In addition, multiple cascaded gratings are generated which should eliminate Bragg degeneracy. This in turn implies that the multiple grating per weight approach can be implemented using self-pumped PCMs and used to form neural networks which obey Hebbian learning rules. Moreover, the observed distributions of beams within a self-pumped PCM, which are determined by the high coupling gain of $BaTiO_3$, scattering centers, reflections from crystal faces, and the geometry of the PCM configuration,[18] suggest that light beams from the entire input plane mix in the crystal. This results in the global interconnection of input pixels by a self-pumped PCM, especially if the PCM is in the Fourier plane of the input spatial light modulator. [19,20] Since a beam from one pixel must diffract from a large set of spatially distributed gratings in order to form the




SINGLE-GRATING WEIGHTED
CONNECTION BETWEEN I AND J
(SUFFERS FROM BRAGG DEGENERACY)




MULTIPLE-GRATING WEIGHTED
CONNECTION BETWEEN BEAMS I AND J
(NO BRAGG DEGENERACY)

**Fig. 18. Optical connections made by scattering from multiple cascaded gratings reduce cross-talk due to Bragg degeneracy and allow full utilization of input and output planes.**

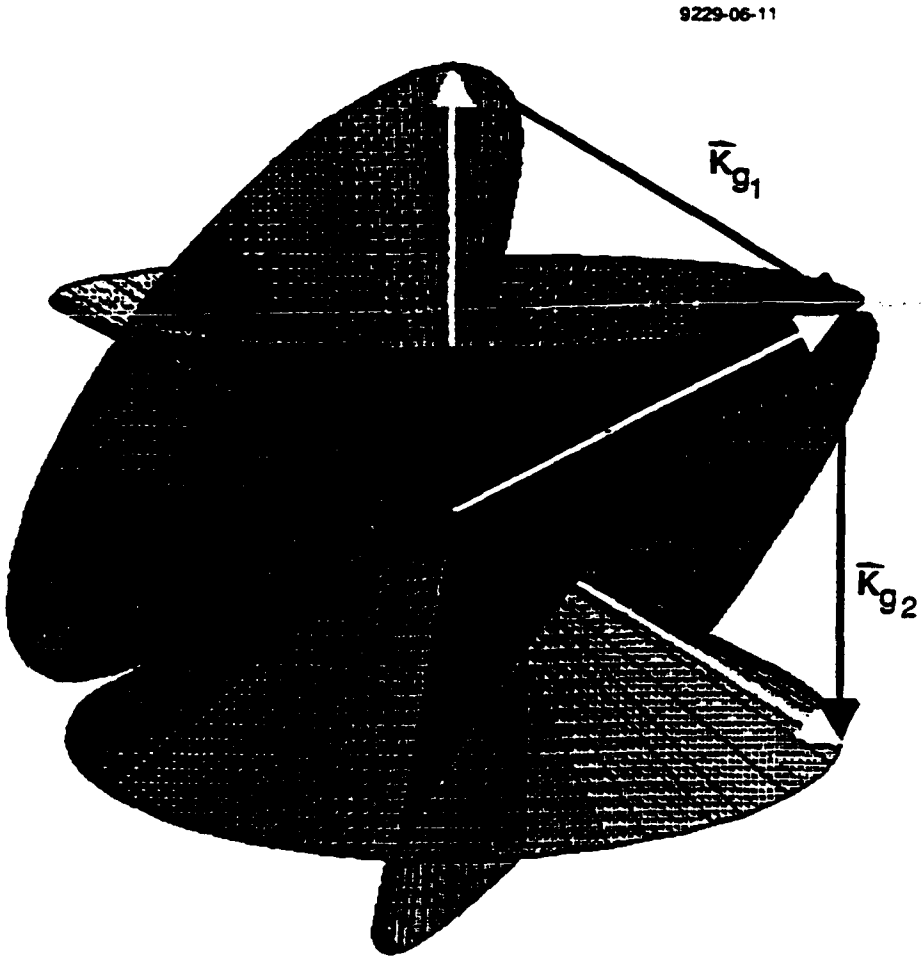



**Fig. 19. Three-dimensional k-space diagram for satisfaction of Bragg conditions at two gratings simultaneously. Only one triplet of light beams (colored white) satisfy the Bragg conditions of both gratings simultaneously.**

conjugate of a second pixel, the Bragg degeneracy crosstalk should be low according to the arguments presented previously.

We performed a series of experiments to test these conjectures for the grating selectivity and global connectivity of the PCM approach. The setup illustrated in Fig. 20 was used to test the vertical and horizontal angular selectivity of a self-pumped PCM. A $BaTiO_3$ crystal was placed in the back focal plane of a lens in the correct orientation for self-pumped phase conjugation. To measure the horizontal selectivity, the input laser beam passed through a Ronchi ruling oriented with the stripes vertical. The beam was focused into the PCM. The phase conjugate or "time-reversed" beam was then reflected from a beam splitter and focused into a photodiode detector.

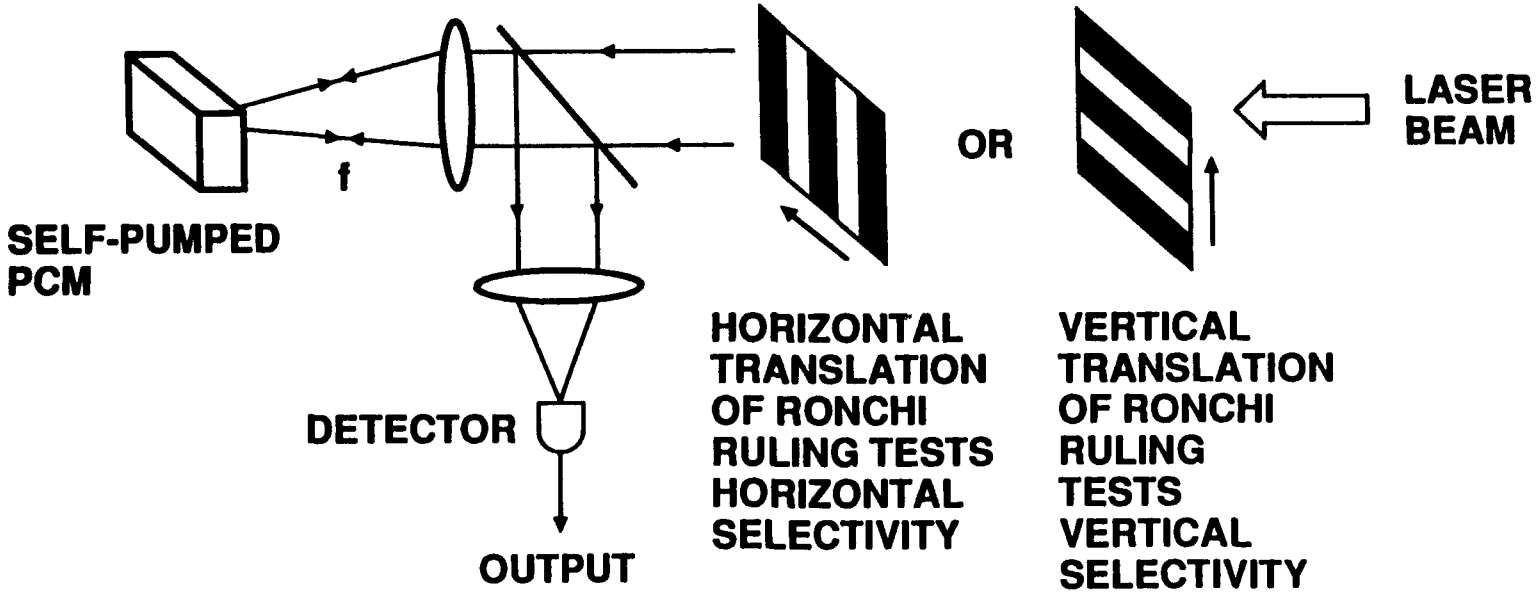



Fig. 20. Experimental configuration for testing Bragg selectivity of self-pumped phase conjugate mirror in readout phase.

After steady state was reached, the laser power was lowered to reduce the rate of formation of new gratings and a recording was made of the conjugate signal as the Ronchi ruling was translated horizontally. Assuming sufficient angular selectivity, the conjugate signal should oscillate as the ruling is translated. Specifically, for zero crosstalk the output vs time should be the autocorrelation function of the Ronchi ruling. The smaller the depth of the oscillation, the greater the crosstalk. Similarly, to measure vertical selectivity the Ronchi ruling was rotated 90° and translated vertically. The experimental results are shown in Fig. 21. For this case the tested angular separation was $1.7 \times 10^{-3}$ radians, which is sufficient for $10^4$ to $10^5$ neurons assuming reasonable lens focal lengths and SLM dimensions. As can be seen in the figure, the extinction in both directions is complete, proving that Bragg degeneracy was eliminated.

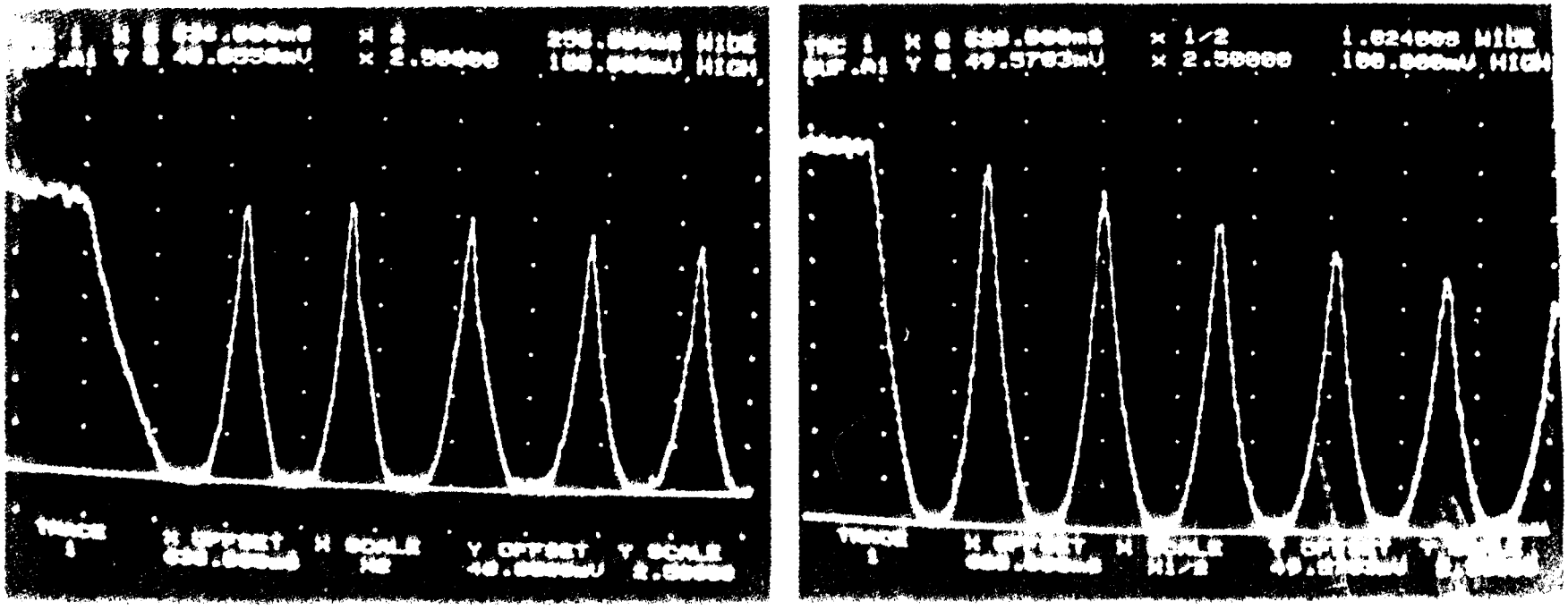
Global connectivity was demonstrated by reading out a complete stored image using less than 3% of the original image as input. In short we found that holograms with good image quality, minimal Bragg degeneracy, and global connectivity could be stored in the PCM configuration. We repeated these experiments using a mutually-pumped PCM and obtained similar results.

Despite the positive results we obtained for the PCM method in terms of Bragg degeneracy and global connectivity, we found that we could not implement neural networks. Investigating further, we found that the *dynamic* characteristics of self- and mutually-pumped photorefractive PCMs resulted in inter-hologram crosstalk when multiple exposures were recorded in the crystal. In particular, we found that previously recorded holograms influenced the formation of new holograms in such a way that extraneous connections were made. For example, if two orthogonal images A and B (e.g., with no common pixels) were sequentially recorded in the PCM, subsequent readout with either image would recall the superposed phase conjugates of *both* images.

This effect is illustrated in Fig. 22 which plots the conjugate signals for the two images as the input was repeatedly switched between A and B. Note that when the input is switch to B from A, *both* $A^*$ and B* appear. (As the exposure continues $B^*$ eventually decreases to zero as B's gratings are erased.) This effect is analogous to the "hologram sharing" effect in mutually-pumped PCMs in which one hologram formed by an input beam affects the formation of a second hologram formed by a second mutually incoherent input beam.[21] Gratings which can be "shared" between two holograms have higher gains over other gratings and tend to be enhanced. This in fact is the principle behind mutually-pumped PCMs in which two mutually incoherent beams can read out each other's conjugate beam. In the case considered here the exposures of the two holograms are separated in time rather than simultaneous, but the principle is the same.

The hologram sharing effect results in very strong inter-hologram crosstalk and is detrimental to optical neural networks. This led us to consider a second non-phase-conjugate method which uses the same multi-grating principle to avoid Bragg degeneracy but does not suffer

$\lambda = 514$ nm, $f = 15$ cm, 50 LP/INCH RONCHI RULING



HORIZONTAL SELECTIVITY

VERTICAL SELECTIVITY

**Fig. 21. Bragg selectivity of self-pumped PCM ($BaTiO_3$ #69G) $\Delta\Theta = 1.7\times10^{-3}$ Rad.**

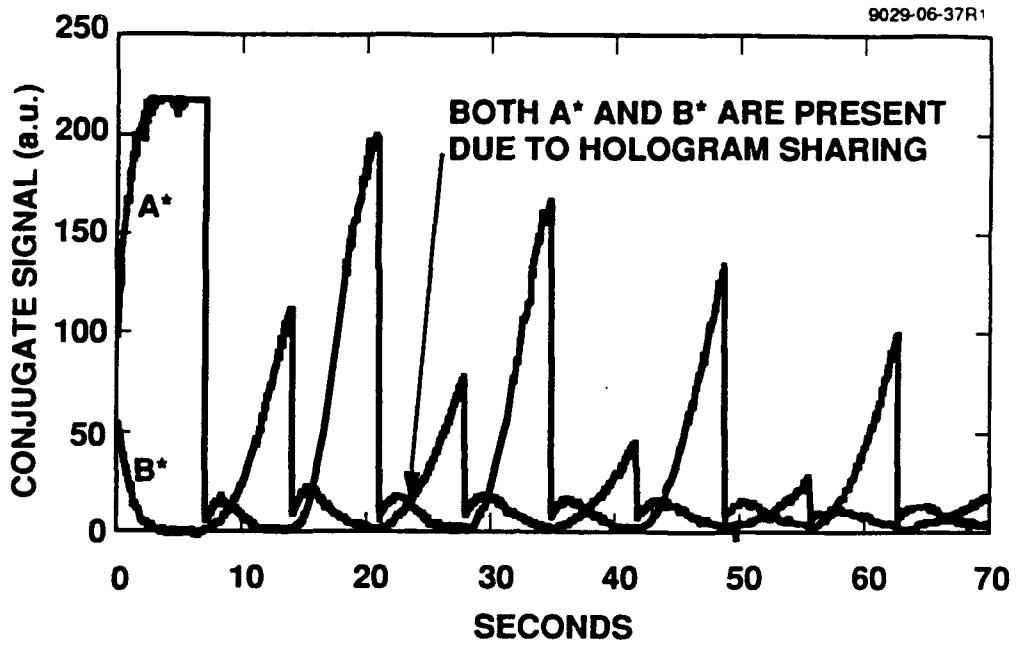



**Fig. 22. Time-sequenced exposure of self-pumped phase conjugate mirror with two orthogonal inputs A and B. Presence of both conjugates $A^*$ and $B^*$ simultaneously is due to effects of hologram sharing and indicates inter-hologram crosstalk.**

from hologram sharing. This method is based on the observed phenomena of beam fanning in photorefractive crystals.

"Beam fanning" is a well-known effect in high gain photorefractive crystals in which an input beam is initially scattered by small inhomogeneities in the crystal, resulting in low amplitude scattered optical noise. The noise beams then interfere with the original input beam and write gratings. Scattering of the input beam by these gratings selectively amplifies some of the noise beams by the process of energy transfer in photorefractive two-wave mixing. The amplified beams then write new gratings and the process cascades. Which beams are amplified most is determined by the electrooptic tensor of the crystal and the orientation of the input beam. The net effect is that the input beam literally fans in the crystal as it writes a set of spatially and angularly distributed gratings.

As shown in Fig. 23, this fanned light can be used as a reference beam when it is interfered with a second, unfanned object beam to form a holographic connection which suffers neither from Bragg degeneracy (because multiple cascaded gratings store each connection) nor from hologram sharing (because the conjugate beam is not used for readout). An unfanned object beam is used because object beam fanning would degrade the quality of the reconstructed object image. Fanning can be controlled so that the reference beam fans and the object beam doesn't by adjusting the orientation of the beams relative to the crystal. The fanning process generates high gain-length

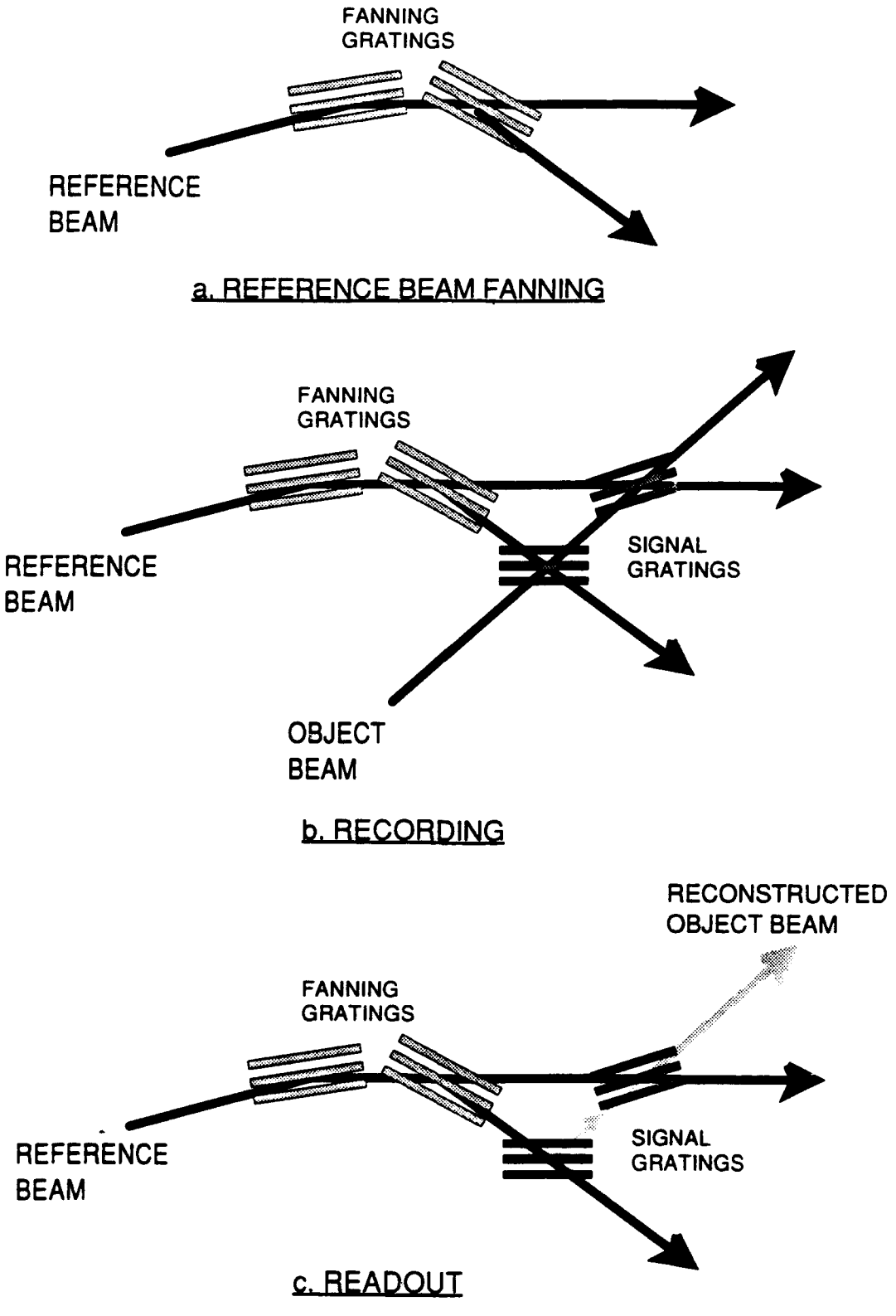



**Fig. 23. Recording of connection using fanned reference beam.**

product "fanning gratings" which divide each reference beam into a set of beams at different orientations and locations. During recording the object beams form a set of "signal gratings" in which the connection weights are stored. Both the signal and fanning gratings are angularly and spatially multiplexed. Upon readout each reference beam must match the Bragg condition at a multitude of fanning gratings, which breaks the Bragg degeneracy. In addition, the beamlets reconstructed by the signal gratings are all in phase and sum coherently, hence the aggregate diffraction efficiency can be high even though the diffraction efficiency of any individual signal grating is small. In addition, the low gain-length product of the individual signal gratings greatly reduces distortions due to beam coupling.

We have applied crosstalk tests to the fanning method as described above for the PCM method and found that the Bragg degeneracy advantages of PCM are maintained but hologram sharing is eliminated. Recordings of holograms in a fanning crystal of $BaTiO_3$ do not show Bragg degeneracy, hologram sharing crosstalk, or distortions due to beam coupling. An example of holographic recording using an arbitrary 2-D gray-scale reference image is shown in Fig. 24.

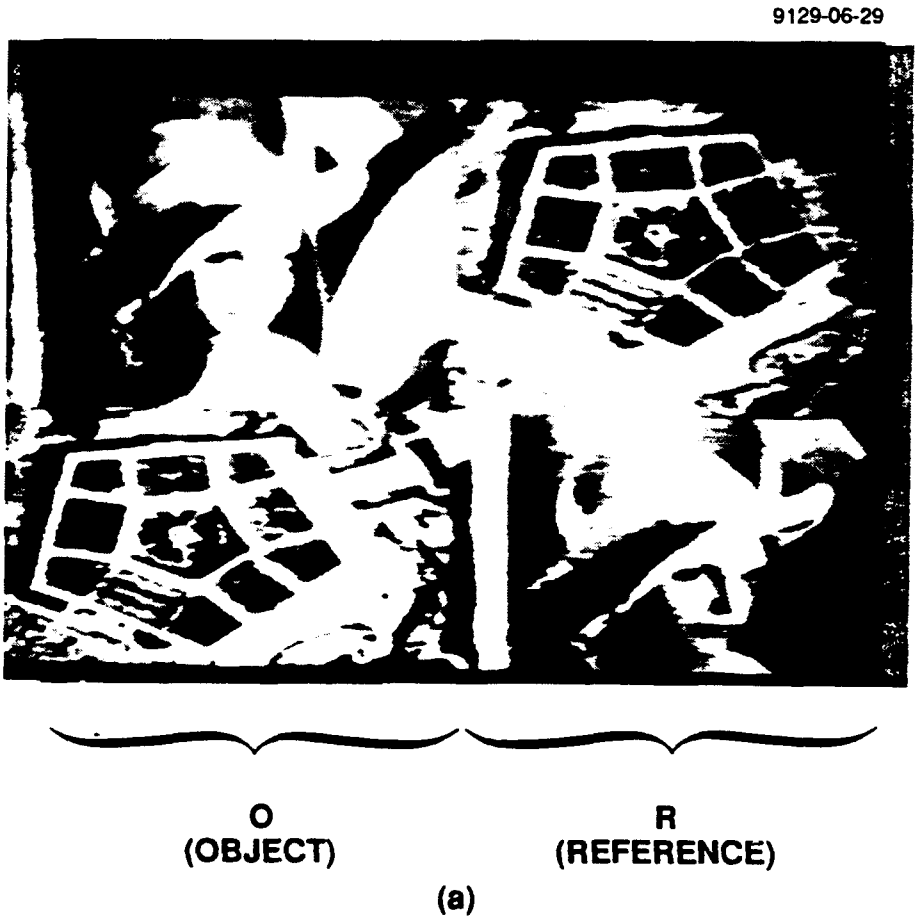



**Fig. 24. Experimental demonstration of recorded image quality using fanned reference beam and 2-D non-subsampled reference. (a) Object and reference used for recording.**

9129-06-27

TRANSMITTED OBJECT
(b)

OBJECT RECONSTRUCTED
USING 50% OF ORIGINAL
REFERENCE
(c)

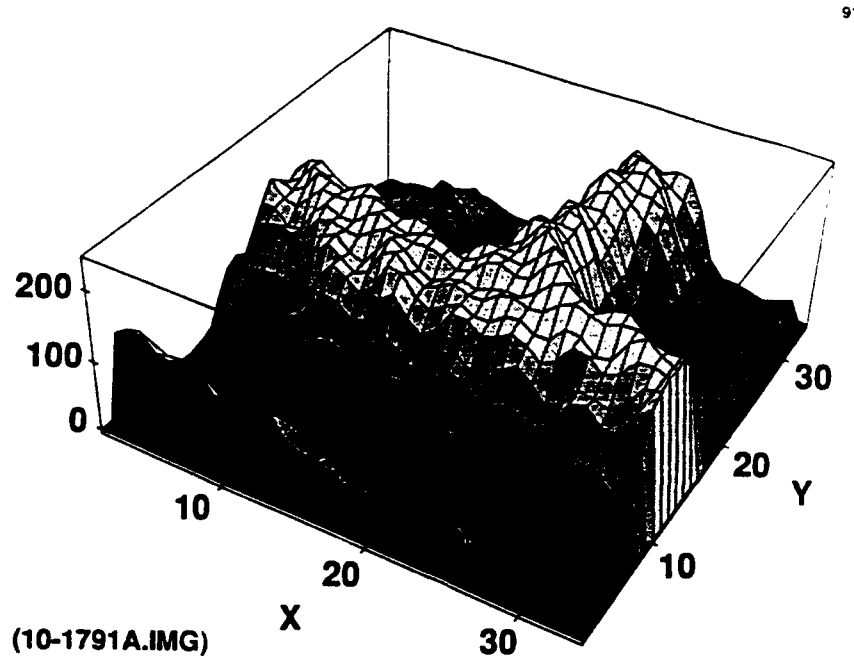
Fig. 24. Experimental demonstration of recorded image quality using fanned reference beam and 2-D non-subsampled reference. (b) Original object for comparison. (c) Hologram of object reconstructed using 50% of original gray-scale reference.

The object and reference both consisted of combinations of Pentagon and woman images displayed on a LCTV with 30,000 gray-scale pixels (90,000 pixels if the individual red, green, and blue pixels of the color LCTV are counted). The original object (showing the system's optical quality) and reconstructed hologram are shown in Figs. 24(b) and 24(c). Note that only 50% of the original reference was used in reading out the hologram. The reconstructed image is virtually identical to the original, as shown in Fig. 25 which is a 3-D plot of a detail in the original and holographic images. Upon magnification of the reconstructed hologram, each of the 45,000 LCTV pixels in the original object was clearly discernible.

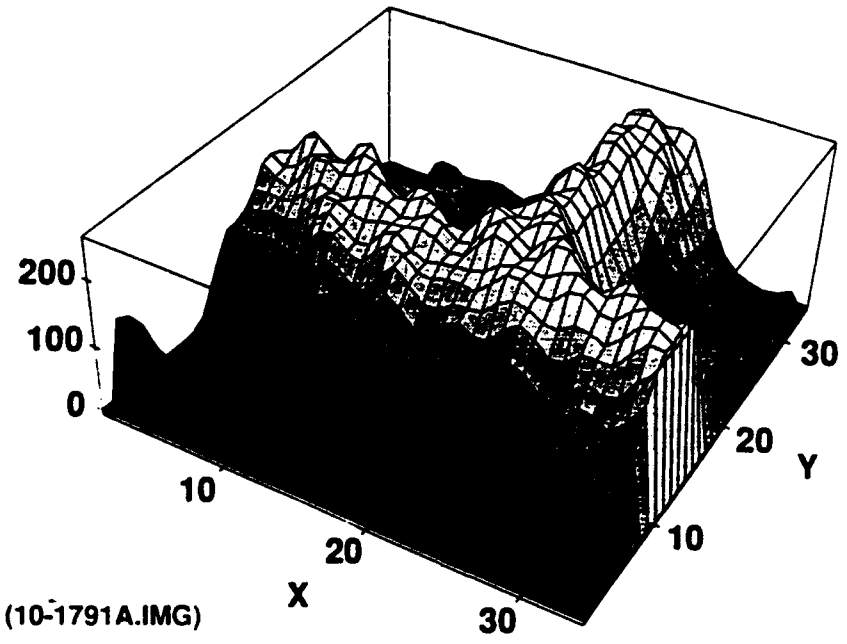
A measurement of the holographic fan-out is illustrated in Fig. 26. In this experiment a uniform (all pixels on) reference was used to record the hologram. During readout an opaque screen with an adjustable small round aperture was translated in front of the reference plane, allowing us to measure the weight vector connected to that portion of the reference. In this experiment the aperture area was 0.0091 of the reference, corresponding to a fanout of 110. (Measurements of larger fanouts were limited by the sensitivity of our CCD camera.) It is evident from the photograph that the entire object was reconstructed when read out with a small fraction of the reference, demonstrating global connectivity.

We also measured the time required to write holograms to saturation using fanned reference beams. We used a two slotted wheels on a common rotating shaft to periodically switch off the object beam in order to measure the buildup in diffraction efficiency as the hologram was being written. The results are shown in Fig. 27 for a crystal of $BaTiO_3$ which was cut at 45° to the c-axis. This crystallographic orientation maximizes the effective electrooptic coefficient. In this case the wavelength was in the green (514 nm) and the total optical power incident on the crystal was 17.5 mw. The time required to write to saturation was 25 msec. Weights are typically adjusted by small amounts during the learning phase, therefore all the weights connecting two neuron layers can be adjusted in less than 1 msec. This would correspond to learning rates of greater than $10^{11}$ connection updates per sec if each layer contained $10^4$ neurons. Since the photorefractive time constant is inversely proportional to the optical power, the update rate can be further increased by raising the laser power.

Improvements in packaging size and cost would be obtained if a laser diode light source could be used. Currently available $BaTiO_3$ is less sensitive at the infrared laser diode wavelengths and the photorefractive gain is less. Nevertheless, we were successful in demonstrating cascaded-grating holography in 45° $BaTiO_3$ using a laser diode light source operating at 830 nm. We used a 150 mw single mode laser diode from SpectraDiode Labs. The total optical power at the crystal was 31 mw. As shown in Fig. 28, we could resolve a holographic image with exposure times as short as 100 msec. Nevertheless, due to the much reduced sensitivity of $BaTiO_3$ at 830 nm, the time constant for hologram writing was much longer than at 514 nm. Referring to Fig. 29, we measured a saturation time of 10 sec for an incident power of 45 mw at 830 nm. Thus more work is needed to improve the infrared sensitivity of the recording medium for use with laser diodes.

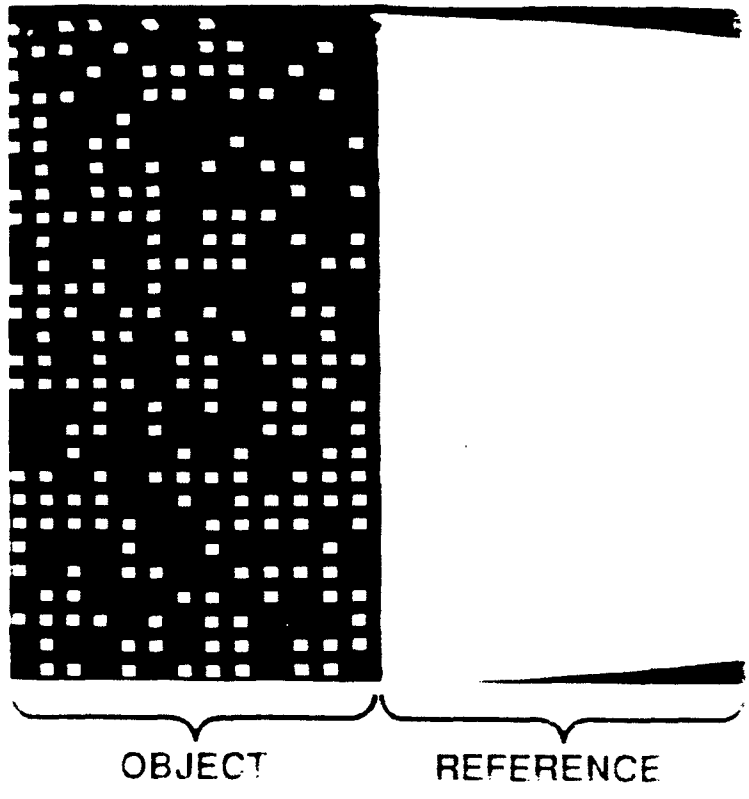



ORIGINAL OBJECT DETAIL




HOLOGRAM RECONSTRUCTED
USING 50% OF GRAY-SCALE REFERENCE

Fig. 25. 3-D plots showing detail of Pentagon images in Fig. 11. (a) Original object. (b) Hologram of object reconstructed using 50% of original gray-scale reference.



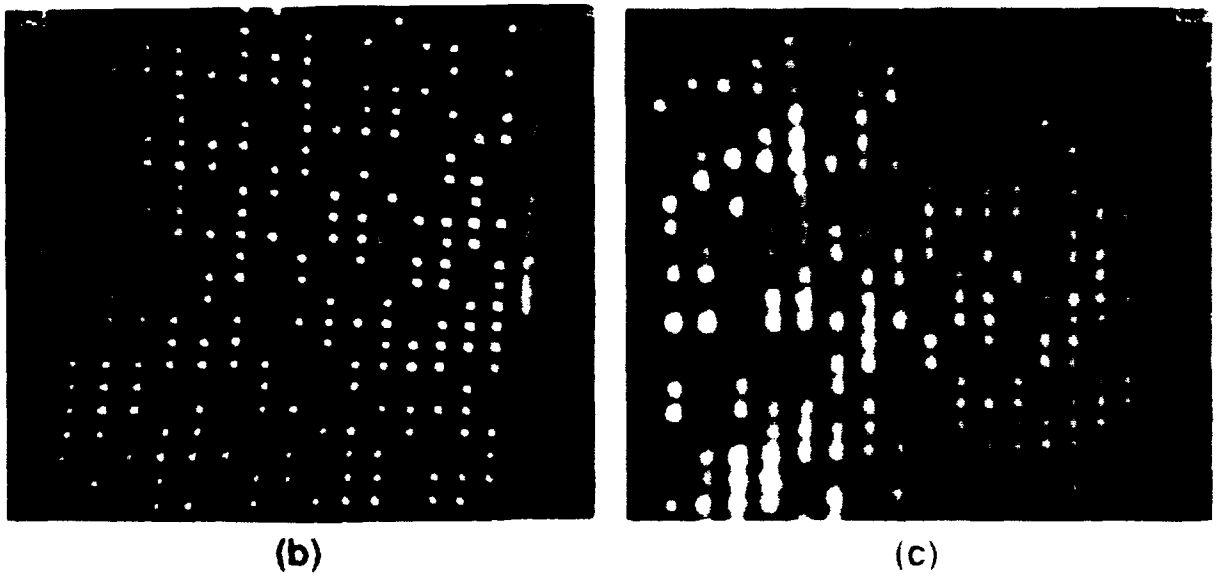




**Fig. 26. Experimental demonstration of fanout and global connectivity using fanned reference beam. (a) Object and reference used for recording. (b) Hologram of object using 100% of original reference. (c) Hologram reconstructed using 0.9% of original reference showing fanout of 1:110 and global connectivity. Demonstration of larger fanout was limited by CCD sensitivity.**

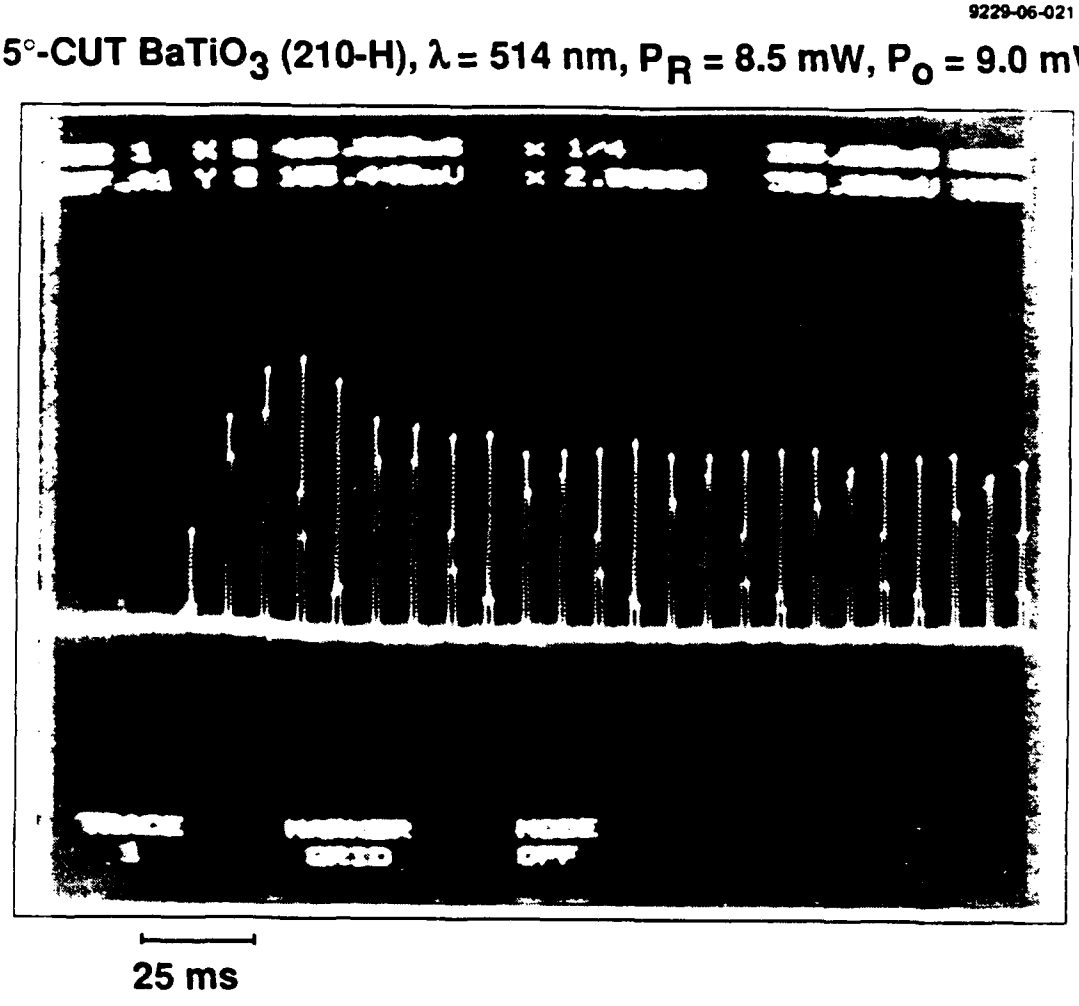



**Fig. 27. Hologram writing time (fanned reference).**

Alternatively, compact solid state lasers emitting in the green could be used instead. The impact of these options on package design are discussed in the packaging section of this report.

## 5.2 HOLOGRAM SUPERPOSITION

The learning phase of neural network models involves many parallel weight adjustments in response to an internally or externally generated error signal. In holographic optical neural networks this function is performed through the superposition of many exposures of the photorefractive crystal in which the strengths of individual gratings are strengthened or weakened. Ideally, the holographic process should not distort the linear superposition of weight update vectors. In the following we derive necessary conditions for linearity using a simple exponential decay model. We also point out that weight erasure can distort the neural network learning rule if precautions are not taken.

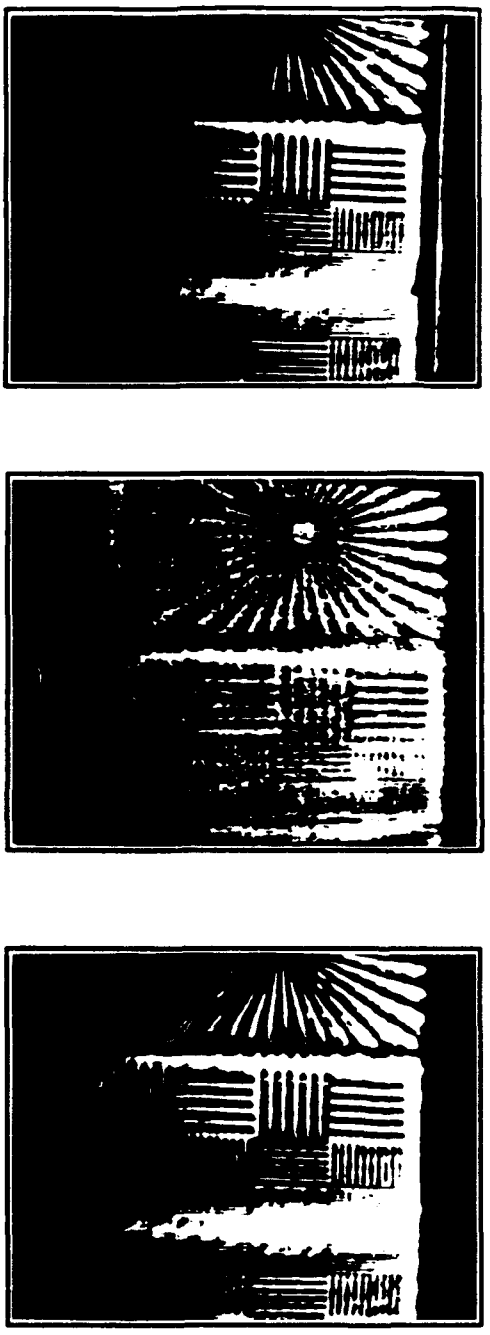


Fig 28. IR hologram in 45°-cut $BaTiO_3$ (210-H) (fanned uniform reference). $\lambda$ = 830 nm, $P_R$ = 21 mW, $P_O$ = mW.

9229-06-023

45°-CUT $BaTiO_3$ (210-H), $\lambda$ = 830 nm, $P_R$ = 21 mW, $P_O$ = 24 mW

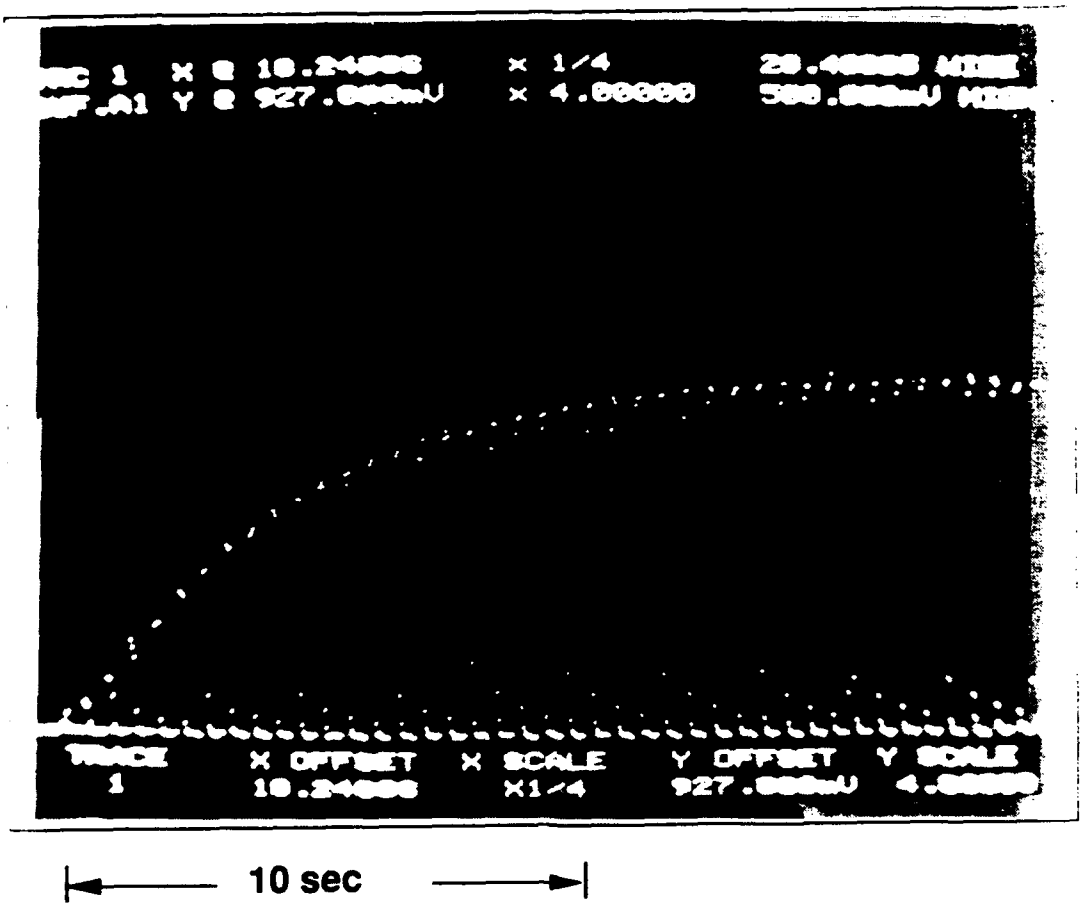



**Fig. 29. Hologram writing time (fanned reference).**

Following Valley[22], we assume a quasi-CW approximation for the photorefractive equations developed by Kukhtarev et al..[23] The coupled equations describing the time and space evolution of two plane waves with amplitudes $A_p$ and $A_s$ are then given by

$$\frac{\partial A_p}{\partial x_p} = \frac{-ik}{2n_p} r_{eff} A_s E$$

$$\frac{\partial A_s}{\partial x_s} = \frac{-ik}{2n_p} r_{eff} A_p E^*$$

$$\frac{\partial E}{\partial t} + \frac{E}{\tau} = \frac{iE_{sc} A_p A_s^*}{\tau I_0}$$

where

$$I_0 = |A_p|^2 + |A_s|^2$$

$$E_{sc} = \frac{E_D}{1 + \frac{E_D}{E_q}}$$

$$\tau = \tau_{di}\left(1 + \frac{E_D}{E_M}\right) \Big/ \left(1 + \frac{E_D}{E_q}\right)$$

and

$$\tau_{di} = \frac{\varepsilon\varepsilon_0}{e\mu n_c}$$

$$n_c = \frac{\alpha I_0 \tau_R}{h\upsilon}$$

$$E_D = k_B T k_g / e$$

$$E_q = \frac{eN_A(1 - N_A/N_D)}{\varepsilon\varepsilon_0 k_g}$$

$$E_M = \frac{1}{k_g \mu \tau_R}$$

The quantities $r_{eff}$, $\varepsilon$, $N_D$, $N_A$, $\alpha$, $\mu$, $n_s$, $n_p$, and $\tau_R$ are material constants, $k=2\pi/\lambda$, and $k_g$ is the wave number of the space charge field grating. The above equations are applicable to crystals in which one charge carrier dominates. The effects of absorption on the spatial derivatives of $A_p$ and $A_s$ are neglected.

In this simplified model the response time $\tau$ is inversely proportional to the total light intensity $I_0$. Let us define a normalized response time $\tau_1 = I_0\tau$ which is independent of the total light intensity. The equation for the time evolution of the grating space charge field then can be written as

$$\frac{\partial E}{\partial t} = -\frac{I_0}{\tau_1} E + \frac{iE_{sc} A_p A_s^*}{\tau_1}$$

The saturation value of the space charge field is

$$E_{sat} = iE_{sc} \frac{A_p A_s^*}{I_0}$$

The incremental change to the space charge field is given by

$$\delta E = -\frac{I_0}{\tau_1} E \delta t + \frac{iE_{sc}}{\tau_1} A_p A_s^* \delta t$$

The second term is the desired Hebbian contribution to E. It is modified by the first term which represents weight erasure. Weight erasure reduces the *magnitude* of a weight vector that is not currently being updated ($A_p$ and/or $A_s$ equal to 0) but not its *orientation* because $\delta E$ is then proportional to E. The orientation of a weight vector is affected by weight erasure if $A_p$ and $A_s$ are nonzero. The erasure term distorts gradient descent neural network models such as backpropagation which are sensitive to both orientation and magnitude of $\delta E$, but has little effect on other neural networks such as the Perceptron. If the output values of a Perceptron are (1,-1) and the threshold is 0, then weight decay will not affect the Perceptron output because the linear discriminant depends only on the orientation of the weight vector.

The above equation can be rewritten using neural network nomenclature. Each weight $w_{ij}$ connecting neurons i and j is then described by an equation of the form

$$\frac{dw_{ij}}{dt} = -\frac{w_{ij}}{\tau_E} + \eta_{ij}$$

where $\eta_{ij}=iE_{sc}A_iA_j^*/\tau_W$ is the updating term proportional to the product of the amplitudes of writing beams i and j. $\tau_W$ and $\tau_E$ are the writing and erasing time constants, respectively. The weight saturation value is $w_{sat}=\eta_{ij}\tau_W$. In the following we will assume $\tau_E$ and $\tau_W$ are constant (equivalent to assuming equal light intensities in all updates). We will also suppress the neuron indices i and j.

We consider *multi-epoch* recording (sometimes called incremental recording) in which each hologram is exposed for a short time and the set of holograms is cycled through many times.[24] This recording method is appropriate for neural networks in which the number of required exposures is not known beforehand. We will also assume *non-orthogonal* holograms in which each connection weight may be updated an arbitrary number of times during learning. Our objective is to derive conditions for the linear superposition of exposures in multi-epoch recording.

In the first epoch after the n-th exposure the solution to Eq. (1) is

$$w_n = e^{-t/\tau_E} w_{n-1} + \left(1 - e^{-t/\tau_W}\right)$$

in which subscripts now refer to exposures and t is the time elapsed since the start of the n-th exposure. Equation (2) expresses the current weight values in terms of the weight value at the end of the previous exposure. It can be solved recursively to yield

$$w_n = w_0 e^{-nT/\tau_E} + \tau_W\left(1-e^{-T/\tau_W}\right)\sum_{p=0}^{n-1}\eta_{n-p}e^{-pT/\tau_E}$$

where $w_0$ is the initial weight value and T is the exposure time for each update (assumed constant in multi-epoch recording). Thus at the end of the first epoch the weight value is an exponentially weighted sum of the updates $\eta_n$. Now assume $w_0=0$ and sum over Q epochs, each of which consists of N holographic exposures (one for each exemplar). After the Q-th epoch,

$$w_Q = \tau_W\left(1-e^{-T/\tau_W}\right)\sum_{q=1}^{Q}\sum_{p=0}^{N-1}\eta_{N-p,q}e^{-[(q-1)N+p]T/\tau_E}$$

where q is the epoch index. Now, if the order of exemplars is the same in each epoch, then by definition $\eta_{N\text{-}p,q}=\eta_{N\text{-}p}$ and the summation over q can be done analytically:

$$w_Q = \tau_W\left(1-e^{-T/\tau_W}\right)\left(\frac{1-e^{-QNT/\tau_E}}{1-e^{-NT/\tau_E}}\right)\sum_{p=0}^{N-1}\eta_{N-p}e^{-pT/\tau_E}$$

In order to minimize distortions due to photorefractive erasure, $w_Q$ should be forced to be proportional to $\eta$ averaged over the N exemplars. This is the case if $T<<\tau_E/N$ :

$$\bar{\eta} = \frac{1}{N}\sum_{n=1}^{N}\eta_n \qquad \text{if } NT<<\tau_E$$

The value of the weight (light amplitude diffraction efficiency) is maximized if in addition the number of epochs satisfies $Q>>\tau_E/NT$ :

$$w_Q \approx \frac{\tau_W\tau_E}{T}\left(1-e^{-T/\tau_W}\right)\bar{\eta} \quad \text{if } NT<<\tau_E<<QNT$$

where

$$\bar{\eta} = \frac{1}{N}\sum_{n=1}^{N}\eta_n$$

Thus under these conditions the weight is simply proportional to the linear average of the update values, which is desirable for neural network models. The diffraction efficiency, which is proportional to $|w_Q|^2$, decreases as $1/N^2$.

A demonstration of multi-epoch recording of superimposed holograms using fanned reference beams is illustrated in Fig. 30. In this case the objects were rotated versions of a gray-scale woman image and the references were orthogonal grid patterns. As shown in Figs. 30(b) and 30(c), each of the superimposed holograms could be read out with very little crosstalk between holograms and with approximately equal diffraction efficiencies. Fig. 31 is a 3-D plot of relative diffraction efficiency measured for 30 superimposed holograms recorded using multi-epoch exposures. The figure plots relative diffraction efficiency of each hologram when read out by each of the references. The objects and references were non-orthogonal random patterns. Due to the non-orthogonality of the patterns, one expects approximately 25% crosstalk, which agrees with experiment. The variations in diffraction efficiency along the diagonal can also be attributed to the random nature of the patterns.

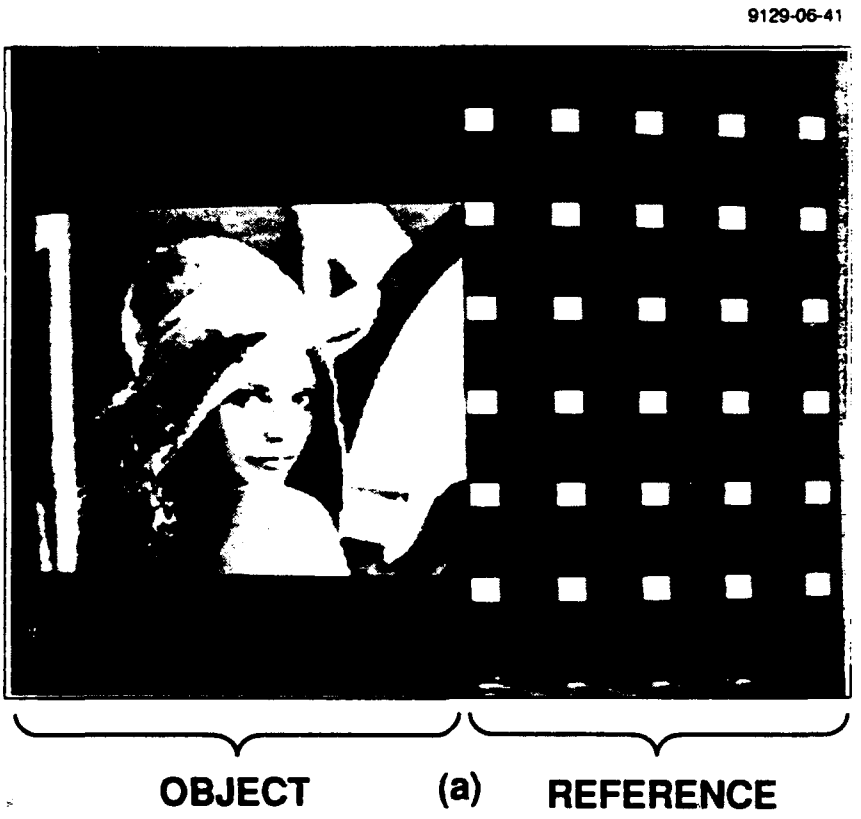



**Fig. 30. Experimental demonstration of hologram superposition using fanned reference beams. (a) One of the object-reference pairs used for recording. References were mutually orthogonal. Objects were rotated versions of the woman image.**

(b)

(c)

Fig. 30. Experimental demonstration of hologram superposition using fanned reference beams. (b) and (c). Readout of two holograms.

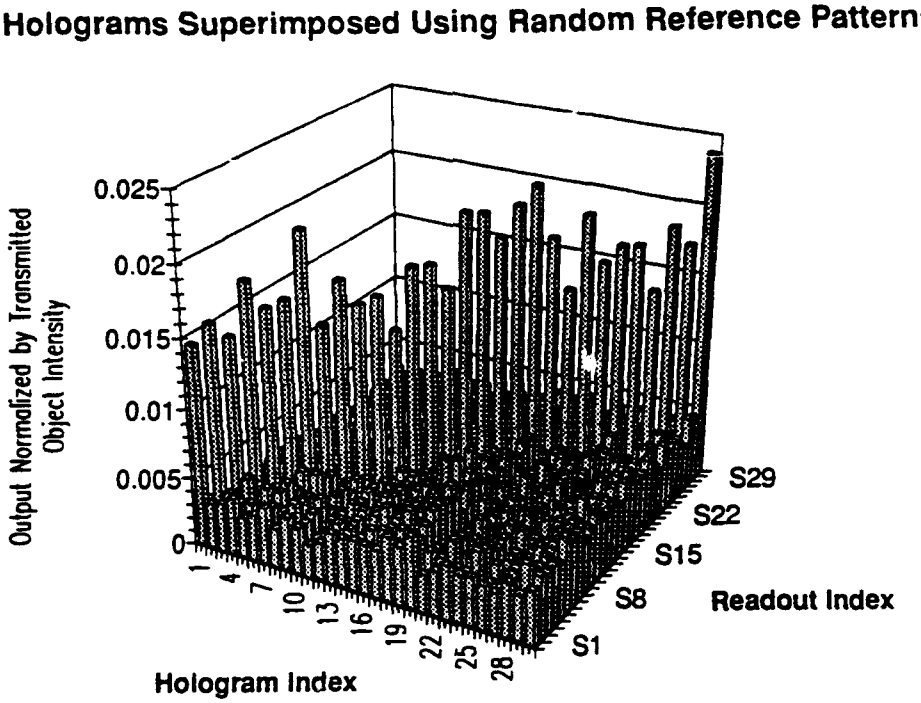



**Fig. 31. Readout of 30 holograms exposed using multi-epoch recording and random reference patterns.**

An interesting result is obtained if we plot the mean hologram diffraction efficiency against the number of recorded holograms on a log-log plot. If multiple holograms are recorded coherently in a single exposure then we would expect the diffraction efficiency per hologram to decrease as 1/N, where N is the number of holograms. If the N holograms are recorded in N time-sequenced exposures then we would expect the diffraction efficiency to decrease as $1/N^2$ due to grating erasure, as discussed above. Yet when we plot the experimental results for time-sequenced recording using fanned references, the data fall between the $1/N^2$ and 1/N lines, as shown in Fig. 32. We conjecture that this may be due to the spatially-multiplexed nature of the fanning gratings and the associated spreading and filamentation of the reference beam. Light from a particular reference pixel is directed by the fanning gratings to the signal gratings which connect it with object pixels. Reference and signal gratings which are not associated with that reference pixel tend to be bypassed, therefore each grating tends to be erased only by light beams which are connected by that grating. The gratings are therefore erased less than we would expect from the above analysis which assumes complete overlap between all of the beams. This improved efficiency performance may be an additional advantage of cascaded-grating holography.

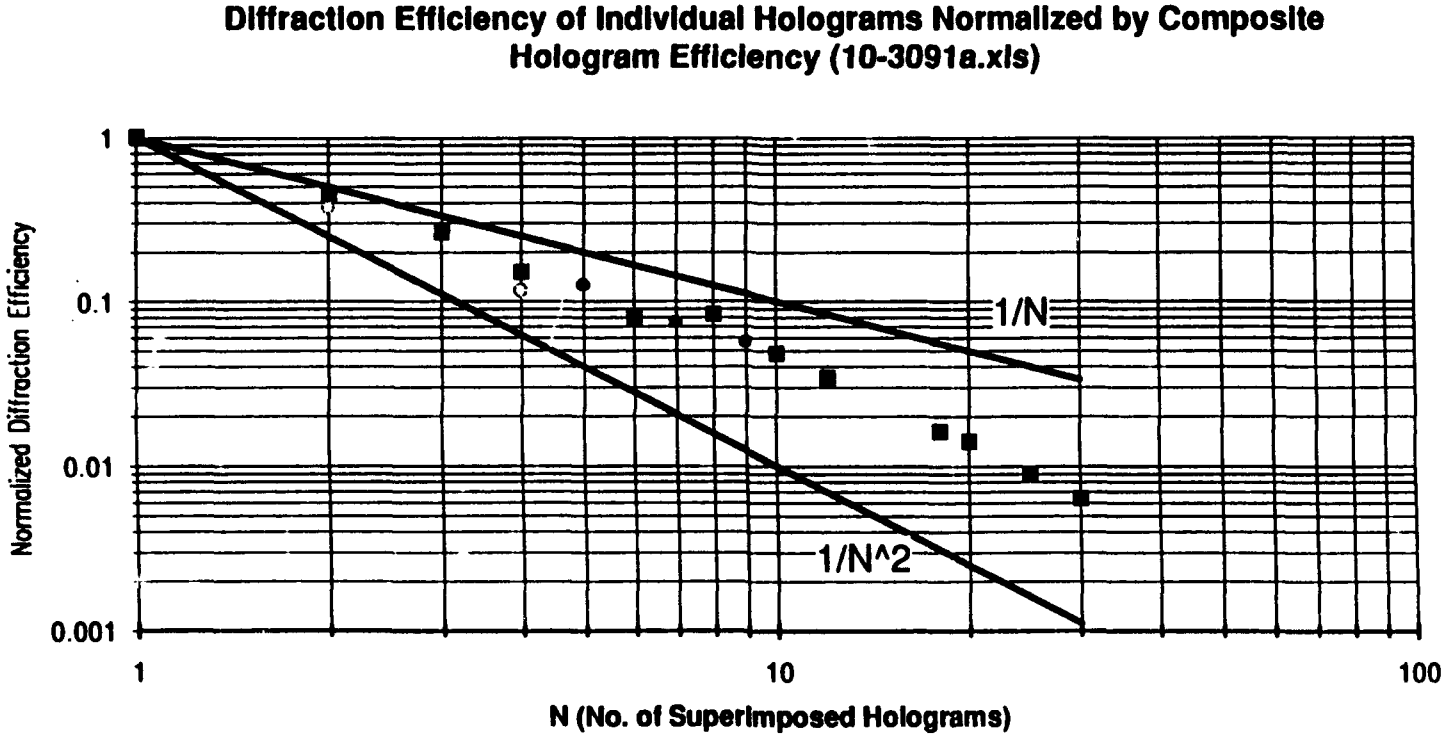



Fig. 32. Diffraction efficiency of individual holograms normalized by composite hologram efficiency and plotted versus number of superimposed holograms.

# SECTION 6

# OPTICAL NEUROCOMPUTER DESIGN AND CONSTRUCTION

## 6.1 OPTICAL REPRESENTATION OF NEURAL NETWORKS

Most neural network models require weights, error signals, and neuron outputs to assume both positive and negative values. Bipolar error signals and weights are necessary even if the neuron response function saturates at 1 and 0 in order to both rectify wrong responses and reinforce correct responses without saturating the outputs. Therefore, means must be provided in an ONN for bipolar inputs and outputs. Holographic ONNs can use coherent or incoherent methods for representing negative numbers. In coherent approaches direct phase modulation of light is used to shift the phase of written gratings. The phase of diffracted light beams is measured by mixing with a reference beam and using interferometric detection. Interferometric detection has potential benefits in terms of increased dynamic range, but it also has many practical difficulties. The phase shifting of the input must be done with great uniformity across the entire SLM input. (Although this can be accomplished using Stokes' principle.[25]) More problematic is the interferometic detection at the output detector array. It is difficult to maintain phase uniformity across the entire output detector array. The system also becomes very vibration sensitive. Finally, many presently available SLMs have an amplitude-dependent phase response which makes independent control of phase and amplitude impossible without using an additional compensating phase-only SLM, which would greatly increase alignment difficulties and system complexity.

In order to avoid these problems, we represent bipolar inputs, weights, and outputs in our SPONN system with spatial multiplexing and electronic subtraction. A bipolar input $y_j$ (which could be an input from neuron j or an error signal) is divided into two nonnegative quantities $y_j^+$ and $y_j^-$ where

$$y_j^+ = y_j \text{ and } y_j^- = 0 \text{ if } y_j >= 0$$
$$y_j^+ = 0 \text{ and } y_j^- = |y_j| \text{ if } y_j < 0$$

These operations are performed electronically in the host computer. $y_j^+$ and $y_j^-$ are then written to two spatially-separated SLM pixels in the reference half of the optical input plane. For the writing or weight adjustment phase of a neural network algorithm, two similar nonnegative quantities are written to two SLM pixels in the object section of the input plane which represent a bipolar error signal in neuron i. The crystal is then exposed with the object and reference beams, forming four

constant-phase weights $w_{ij}^{++}$, $w_{ij}^{+-}$, $w_{ij}^{-+}$, and $w_{ij}^{--}$ in which a bipolar effective weight connecting neurons i and j is encoded. The writing phase is further divided into two subphases in which the weights are adjusted twice: once with the outer-product $\varepsilon_i y_j$ and once with $(-\varepsilon_i)(-y_j)$. Although these two exposures are the same as far as the *effective* weight increment is concerned, they serve to ensure that the cross-diagonal weights are equal, e.g. $w_{ij}^{++}=w_{ij}^{--}=w_{ij}^{+}$ and $w_{ij}^{+-}=w_{ij}^{-+}=w_{ij}^{-}$. The significance of this will be evident below.

The bipolar algorithm in the readout phase is illustrated in Fig. 33. Input $y_j$ is again split into positive and negative parts which read out the weights in the hologram. Square-law detection of the diffracted light beams is performed at two CCD pixels, forming the two intermediate terms

$$d^{+} = \left| \sum_j w_{ij}^{++} y_j^{+} + \sum_j w_{ij}^{+-} y_j^{-} \right|^2$$

$$d^{-} = \left| \sum_j w_{ij}^{--} y_j^{-} + \sum_j w_{ij}^{-+} y_j^{+} \right|^2$$

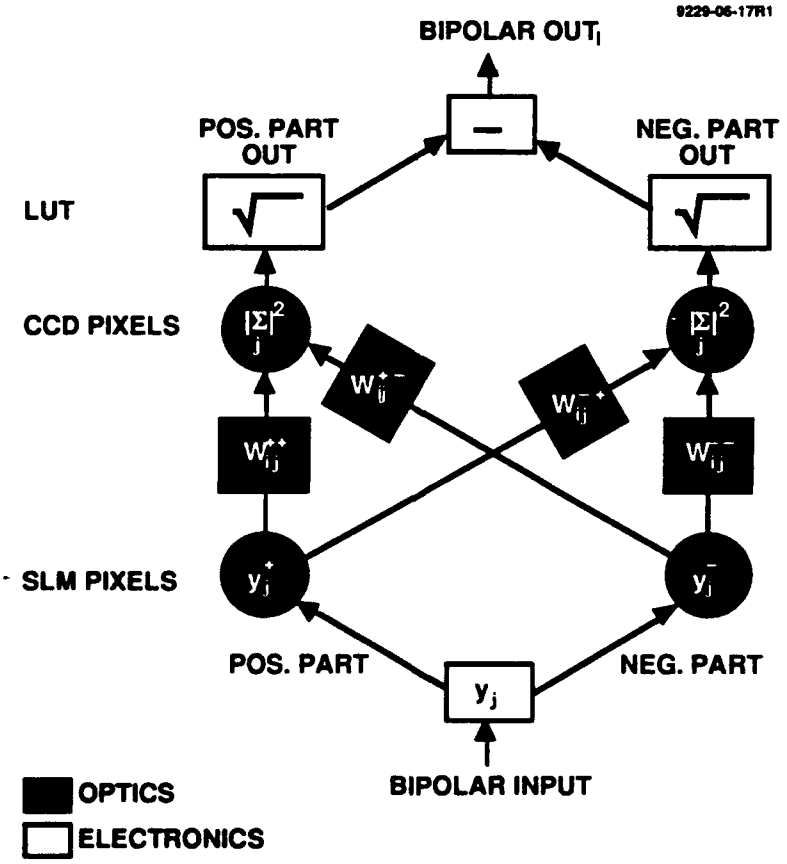



**Fig. 33. Algorithm for bipolar representation of weights and neuron values.**

(In our actual system many CCD pixels are used for each output so that pixel averaging as well as temporal averaging can be used to reduce noise.) The square-root of each of these CCD outputs is formed before subtracting them. Taking advantage of the forced equivalence of cross diagonal weight values, the final output is given by

$$\begin{aligned} out_i &= \sqrt{d^+} - \sqrt{d^-} \\ &= \sqrt{\left|\sum_j w_{ij}^+ y_j^+ + \sum_j w_{ij}^- y_j^-\right|^2} - \sqrt{\left|\sum_j w_{ij}^+ y_j^- + \sum_j w_{ij}^- y_j^+\right|^2} \\ &= \sum_j \left(w_{ij}^+ - w_{ij}^-\right)\left(y_j^+ - y_j^-\right) \\ &= \sum_j w_{ij} y_j \end{aligned}$$

which represents a true bipolar output with bipolar weights and inputs. The electronic portions of the algorithm are not bottlenecks. The square-root is performed at video rates on the entire video frame using a lookup table in the output image processor card while integer subtraction is also performed very quickly. Note that this method relies on the fact that all terms within each of the magnitude-squared brackets have the same phase, namely that of the object beam at the respective CCD pixel.

A diagram of the operation of a single neural network layer in the SPONN system during readout and write phases is shown in Fig. 34. Shaded components denote the optical portion of the system. The first step in the readout phase is the transfer of output from the previous neural layer (stored in host memory) to SLM pixels in the R input plane. Each bipolar value is first converted to two nonnegative components as described above. The R plane pixels then form the R beam which reads out the hologram, performing an optical matrix-vector multiplication. The output is detected by two CCD pixels resulting in the raw outputs $d^+$ and $d^-$. The square root of each output is taken using lookup tables on the output frame grabber board before they are subtracted and passed through the neuron response function, resulting in the output for that layer.

The host computer then calculates an error signal $\varepsilon$ according to the particular neural network model being implemented. The $\varepsilon$ is usually quantized to +1, 0, or -1 because the neural algorithm requires it (as in the Perceptron or quantized backpropagation) or due to the limitations of the input SLM, or both. In our experiments we have used both a Hughes liquid crystal light valve (LCLV) and a Sony active-matrix liquid crystal TV (LCTV). The LCLV has an amplitude-dependent phase response which results in phase errors during learning. A simple way to avoid

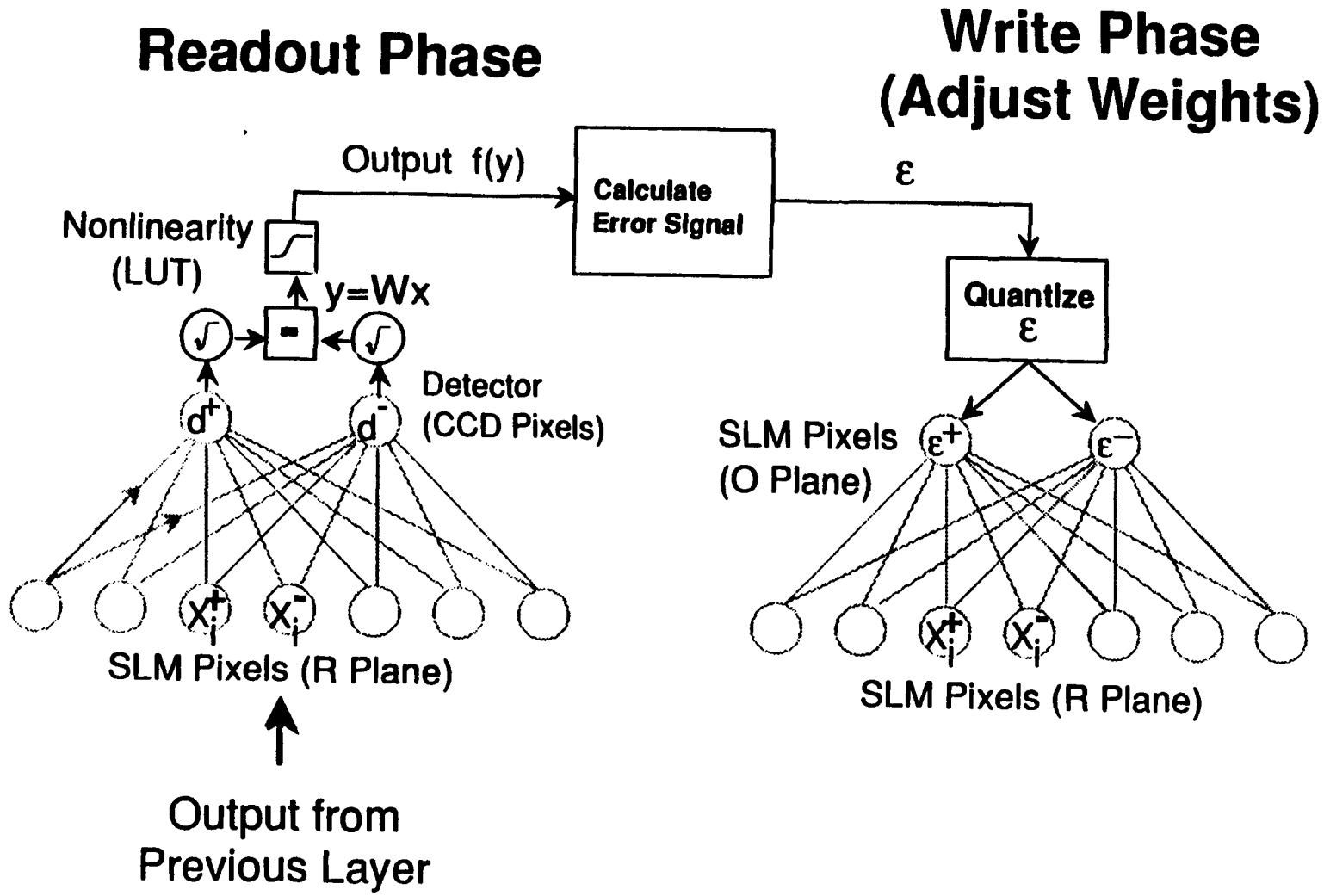



Fig. 34. Schematic of information flow in one layer of SPONN. Shaded lines denote optical pathways.

such errors is to restrict the SLM pixel values to 1 and 0. Our bipolar algorithm then allows us to represent effective values of 1, 0, and -1.

Each quantized error value is then written to two SLM pixels in the O plane according to the bipolar algorithm. The hologram is then exposed using the O and R beams (the original input pattern is still on the R beam), implementing a weight matrix update equal to the outer-product between the ε and input pattern vectors. This sequence is repeated for each layer in the network, each exemplar, and each training epoch.

## 6.2 EXPERIMENTAL SETUP

A diagram of our programmable optical neurocomputer (SPONN or Stimulated Photorefractive Optical Neural Network) is shown in Fig. 35. In assembling this system our goals were hardware simplicity and programmability. SPONN contains a single photorefractive crystal, SLM, and CCD detector. Multi-layer networks are implemented by superimposing different weight layers in the same crystal. A plane wave readout beam from an argon laser at 514 nm was spatially modulated by the SLM. In the experiments described here the SLM was a Sony active-matrix liquid crystal TV (LCTV) which we disassembled. It has 30,000 pixels and a contrast ratio of 200 at 514 nm. We also used a Hughes Liquid Crystal Light Valve (LCLV) in our early experiments. (Although the LCLV is superior to the LCTV in terms of resolution, we took it out of the system due to its more complicated and bulky addressing circuitry, its reflection mode of operation, and the slow response time of the CdS LCLVs that were available to us.)

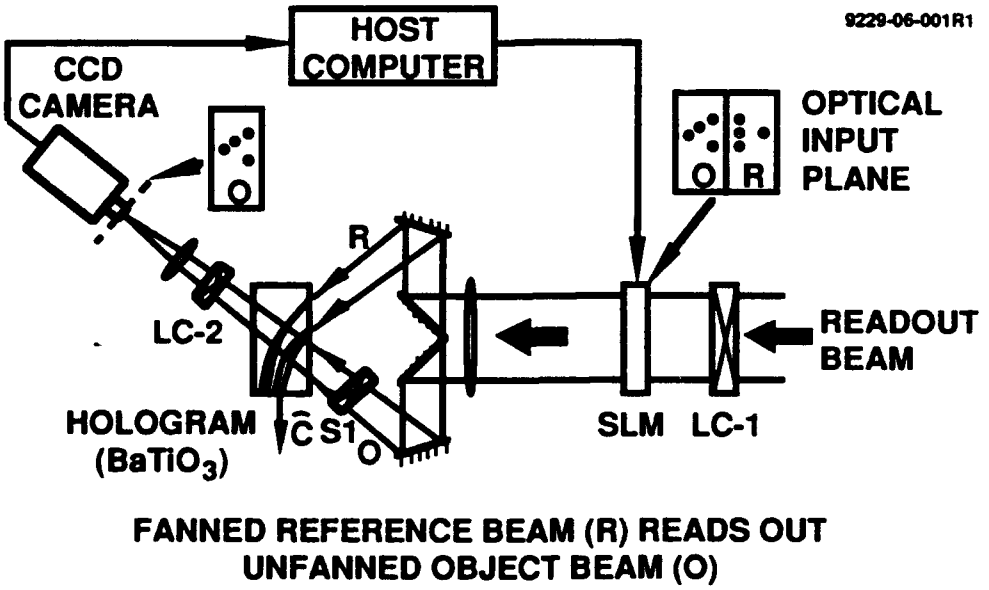



**Fig. 35. Stimulated Photorefractive Optical Neural Network (SPONN) experimental configuration.**

The input plane displayed on the SLM consisted of two portions: the reference (R) and object (O) planes. A system of mirrors separated the O and R planes. The R beam entered the $BaTiO_3$ crystal at an angle relative to the c-axis such that it experienced high two-wave mixing gain, resulting in fanning. The O beam was directed at a low-gain angle and did not fan. (Fanning of the object beam would have degraded the optical quality of reconstructed images since the original O beam would be distorted.) Two image processing cards (VS100 and FG100 from Imaging Technology, Inc.) were installed in the host computer (an 80386 PC). The input image processor was dedicated to providing video input to the SLM. The output image processor grabbed video from the CCD camera which detected the optical output of SPONN. Pixels in the O and R planes represent neurons in the various network layers.

After a hologram was written, liquid crystal shutter S1 was closed by the host computer and the R beam read out the hologram. (Closing S1 ensured that no light leakage due to the finite contrast ratio of the SLM was passed through from the O input plane to the detector during readout.) The CCD camera detected the holographic reconstruction of the O plane. Bragg degeneracy was greatly reduced due to the fanning of the R beam and the reconstructed image quality was good, as explained in the previous sections.

Several measures were taken to reduce noise and distortions. In order to reduce spatial aliasing, the "optical neurons" were oversampled by the CCD pixels. An automated alignment program was used to pick an operator-selected number of CCD pixels to use for each output neuron. Coarse alignment was done using an affine coordinate transformation between the SLM input and CCD output planes calculated from the positions of three points and their corresponding images in the two planes. Final CCD pixel selections were made by sorting and picking the brightest ones. The CCD pixel values were then summed in order to reduce detector noise. Multiple video frames could also be summed to further reduce noise. In addition, nonlinearities in the SLM transfer function were measured and compensated by modifying the output lookup tables of the input image processor. This provided us with greater accuracy and control of the neuron activation function. We found that system noise was dominated by fluctuations in the laser power. In our experiments we used an old argon ion laser with relatively poor stability. Without pixel or frame averaging we measured the output signal relative standard deviation to be 1.9%. By spatially averaging 9 pixels and temporally averaging 9 frames, we reduced the relative standard deviation of the noise to 0.74%. Future SPONN models will use more stable lasers and feedback systems to reduce the noise level.

Computer-controlled liquid crystal cells LC-1 and LC-2 were used to control the optical power during the hologram writing and reading phases. During writing, LC-1 and S1 were both turned on (full transmittance) for high power R and O beams and quick writing times. Simultaneously, LC-2 was turned off so that the CCD camera was not saturated by the O beam.

During reading LC-1 was switched to low so that the gratings were not quickly erased by the R beam. S1 was closed so that the input O beam did not interfere with the reconstructed O beam and LC-2 was turned fully on for maximum detector sensitivity.

The topology of the network being implemented is determined by the organization of the input plane. As shown in Fig. 36, a single layer network with only forward connections is implemented by devoting the entire R plane to the input layer of the network, $L_0$. Likewise, the output layer, $L_1$, occupies the entire O plane. Turning pixels on in the R and O planes forms connections between $L_0$ and $L_1$. $L_0$ then reads out the weights connecting the R and O planes. Multiple layer neural networks are implemented by dividing the input plane into sectors. For example, to implement a net with an input layer $L_0$, a hidden layer $L_1$, and an output layer $L_2$, the input plane is divided into four sectors, as shown in Fig. 18. Connections between $L_0$ and $L_1$ are formed by writing values to the $L_0$ and $L_1$ sectors in the R and O planes, respectively. Similarly, connections between $L_1$ and $L_2$ are formed by writing values to the $L_1$ and $L_2$ sectors in the R and O planes, respectively. The two weight layers are exposed separately to avoid unwanted links between $L_0$ and $L_2$. To read out the network, an input pattern is first written to $L_0$ in the R plane. The optical matrix-vector product is then detected by the CCD camera at $L_1$ in the O plane and electronically processed to form the neuron values in the hidden layer $L_1$. These values are then written to the $L_1$ sector in the R portion of the optical input plane. Another optical matrix-vector product is formed, detected at $L_2$ in the O plane, and processed to form the output neuron values in $L_2$.

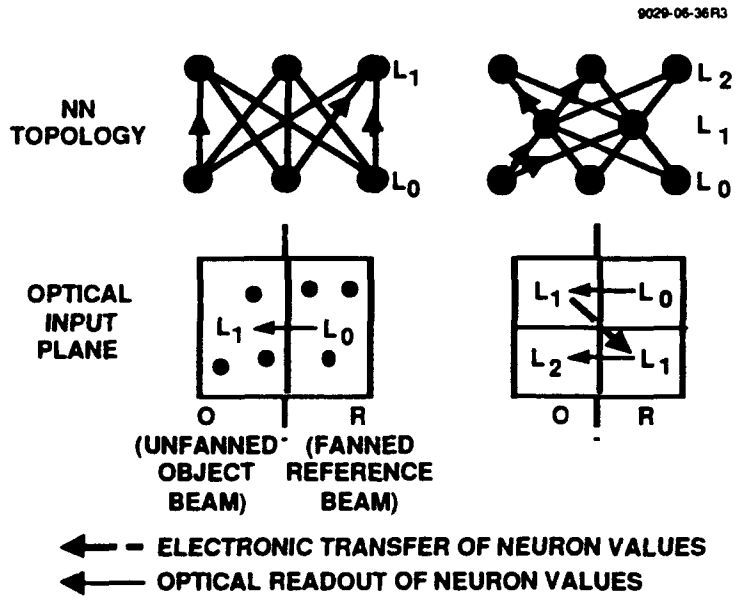



**Fig. 36. SPONN implementation of multi-layered neural networks by spatial organization of optical input plane.**

If more complicated networks with additional features such as backward connections or additional layers are required, they can be implemented by simply dividing the optical input plane into more sectors and changing the software. It is not necessary to adjust the hardware or add more crystals. This easy reconfigurability is due to the capability of arranging neurons in arbitrary patterns in the optical input plane, a feature made possible by the elimination of Bragg degeneracy.

## 6.3 PACKAGING

The prototype research model of SPONN described in the previous section was designed for maximum flexibility using off-the-shelf components, including a bulky water-cooled argon ion laser. No effort was made to package the system in a compact volume as the emphasis of this program was the demonstration of working optical neural networks. However, we did investigate packaging concepts for future versions of SPONN which would use custom components and compact solid state laser sources which should become available at reasonable cost in the next few years. In this section we describe two concepts, one using currently available components and the other using components which should become available within the next five years.

Our near term packaging concept is shown in Fig. 37. The overall size of this package is 15" by 20" plus 10" by 15" for the laser power supply, or about the size of a desktop PC. It is based on a diode-pumped YAG solid state laser available from Adlas, Inc. The laser is frequency doubled using a nonlinear crystal to emit 140 mw at 532 nm. This wavelength is close to optimum for typical undoped $BaTiO_3$ holograms. Although greater optical power would be desirable, it is enough for a demonstration system. We also assume a 3" LCTV as the SLM. The relatively large size of the LCTV sets the dimensions of the optical train. Note the use of a solid optics assembly constructed from prisms and employing total internal reflection to form the R and O beams. By attaching the photorefractive crystal to the optics assembly, the paths of the two beams are completely contained within a single glass or plastic unit. This greatly reduces vibration sensitivity and simplifies optical alignment. Solid optics techniques are used in laser gyroscopes for navigation and guidance of commercial and military aircraft and missiles. Such gyroscopes, which are also interferometric devices, are used successfully in high vibration, grueling environments. As part of our packaging investigation, we constructed a Michaelson interferometer mounted in an aluminum box with a laser diode light source. We found that the interference pattern was unaffected by casual handling of the package or by dropping heavy objects next to it. This increased our confidence that an inexpensive vibration-isolated package for SPONN can be constructed.

By replacing the YAG laser with a laser diode and using a 1" SLM, the package size can be dramatically reduced to less than 5.5" by 10", as shown in Fig. 38. Thanks to market demand from other fields, CCD cameras and laser diodes are now available which would fit into such a

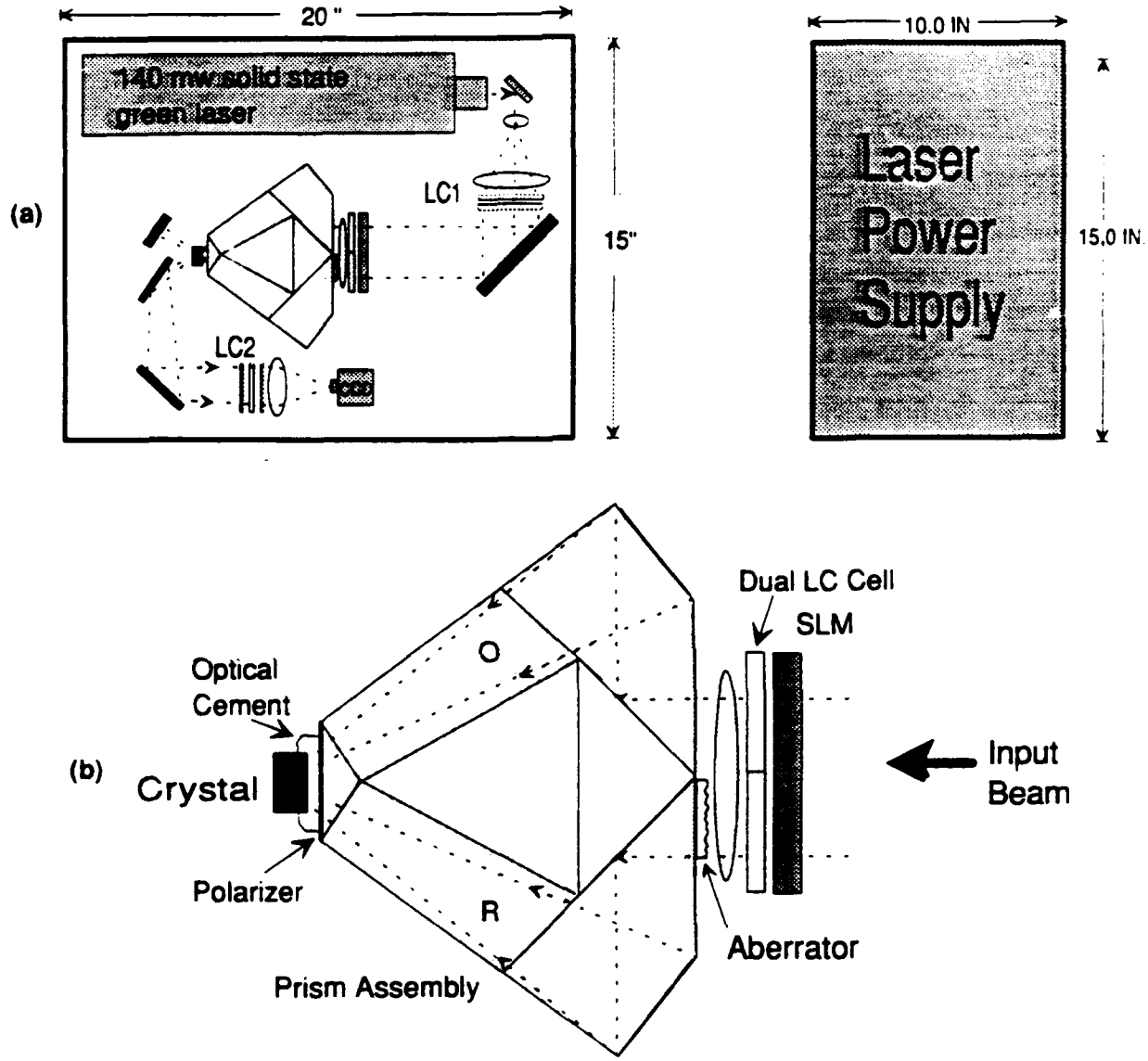



**Fig. 37. (a) Design layout for SPONN optical neural network using presently available green solid state laser. (b) SPONN solid optics assembly for beam direction and improved vibration resistance.**

volume. Components which should become available in the next few years are the 1" by 1" electrically-addressed SLM and doped $BaTiO_3$ which is sufficiently sensitive at the laser diode wavelength. One possibility for the SLM is the Hughes CCD-addressed LCLV.[26] A number of

other companies and universities are also working on electrically-addressed SLMs. An alternative to doped $BaTiO_3$ is to replace it with another photorefractive material which is sensitive in the infrared, such as CdTe, GaAs, or multiple quantum well structures.

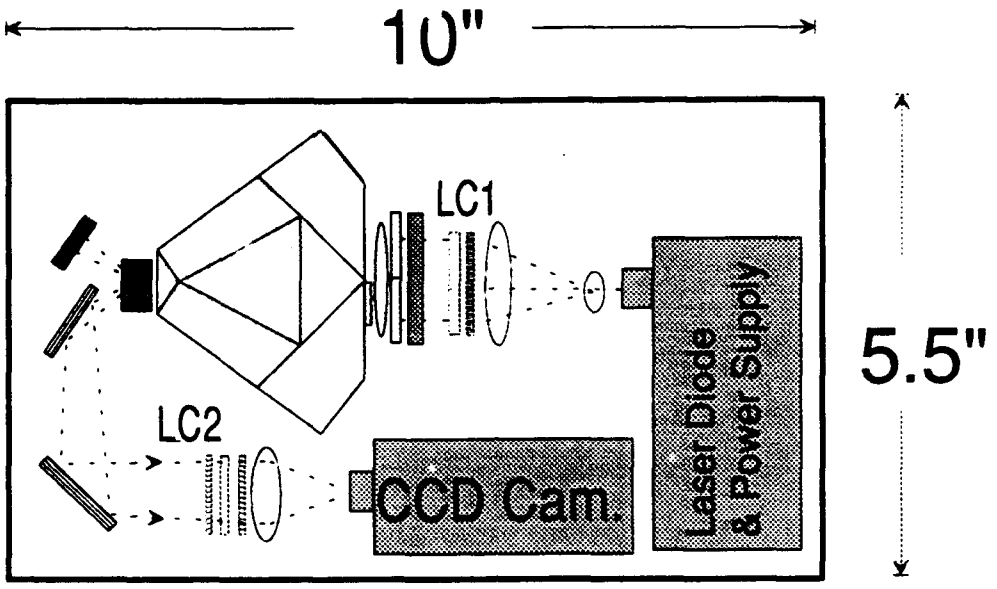



**Fig. 38. Design layout for SPONN optical neural network using laser diode and 1-inch SLM.**

# SECTION 7
# IMPLEMENTATION OF NEURAL NETWORK MODELS

## 7.1 PERCEPTRON

The Perceptron was one of the first neural networks to be invented. In its most commonly implemented form it consists of a single layer of weights connecting a field of input neurons with a single output neuron. (Some of the original Perceptron networks contained a "preprocessing" layer with fixed weights which were not adjusted during learning.) A single output neuron can dichotomize or separate the vector space of input patterns into two classes. The weight values together with the threshold value of the output neuron determine a separating hyperplane for pattern vectors. Therefore it can only distinguish between classes which are linearly separable. The Perceptron is appealing as a first test of neural hardware because of its simplicity and the fact that a learning rule with guaranteed convergence is known (provided a solution can be represented in the first place). Its main weakness is that as a single-layer network it is limited to linearly-separable solutions and therefore cannot solve many problems of practical interest.

The output of a simple Perceptron is given by

$$y = h\left(\sum_j w_j x_j - \theta\right)$$

where w is the weight vector connecting the input neurons x with the output neuron, $\theta$ is the output neuron threshold value, and h(z) is a hard-threshold response function with outputs 1 or -1 for z>0 and z<=0, respectively. Input patterns are binary and normally assume values of 1 or 0. $\theta$ can be learned by setting one of the input neurons to 1, although in our experiments we set $\theta$=0. The learning rule is simple and can be expressed in terms of the weight update vector as

$$\Delta w_j = -\eta(y - D_c)x_{j,c}$$

where $D_C$ is the desired output for exemplar c. If the output is correct, no change is made. If y=1 but $D_C$=-1, a change proportional to the negative of the exemplar is made. Finally, if y=-1 but $D_C$=1, a change proportional to the exemplar is made. A slight modification of this algorithm was

made in the optical implementation. A "forbidden zone" centered on zero was defined for neural outputs before thresholding: if the value of $\Sigma w_j x_j - \theta$ was within this zone then a correction to the weights was made even if the thresholded network output was correct. This made the system more robust by penalizing small weight values.

A diagram of the SPONN optical input plane for implementation of a Perceptron network is shown in Fig. 39. The R plane contains the pattern in the input neuron layer $L_0$ and the O plane contains the output neuron layer $L_1$. In our experiments we have used both a single output neuron and many output neurons. The latter case is essentially many Perceptrons operating in parallel on the same input patterns but with different classification goals. In all of the experimental examples described here the problem to be solved was the transformation of a set of random binary exemplars (2-D random patterns with pixel values of 1 and 0) into another set of random binary patterns.

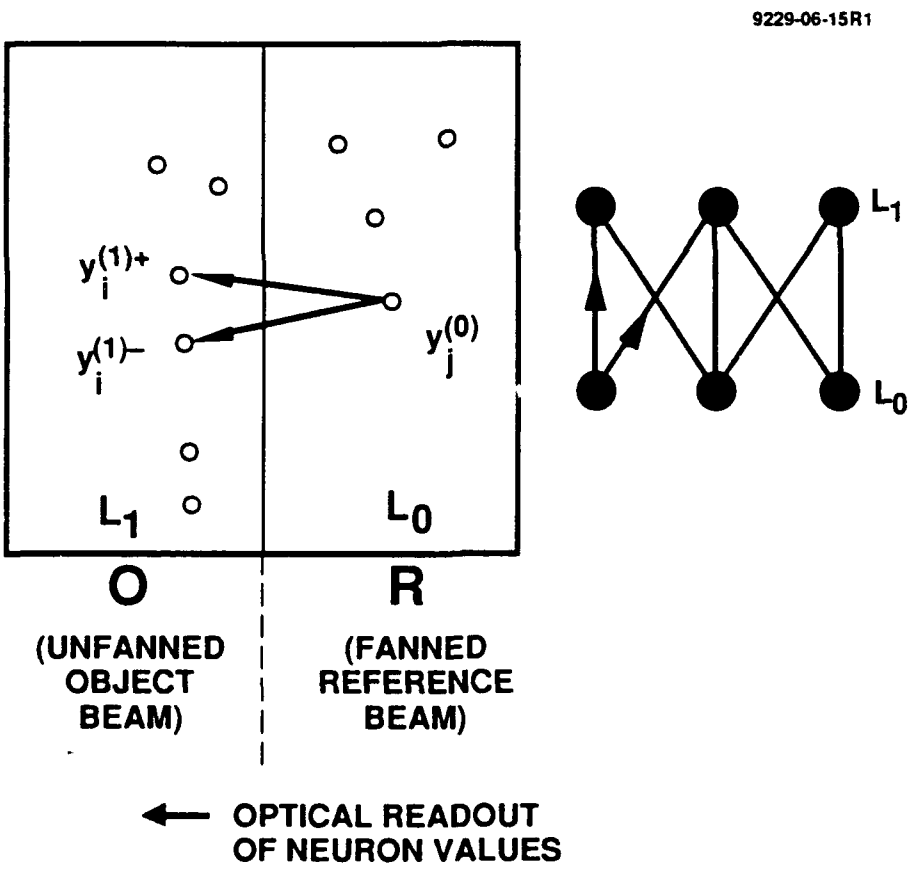



**Fig. 39. Optical input plane for single layer Perceptron.**

The first Perceptron experiment had a single output neuron, the goal was to separate or dichotomize a set of random patterns into two classes. The results of this experiment is shown in

Fig. 40 which is a plot of total error during learning versus epoch number. In this case the optical neural network learned to dichotomize 96 random patterns after 29 epochs. Each pattern consisted of 1920 pixels (60x32). Increasing the number of pixels tended to reduce the number of patterns that could be learned although with 7680 pixels (120x64) the system could still learn 42 patterns. In order to test the noise level in the system, we attempted to dichotomize two patterns. The patterns were identical except for a prescribed number of differing pixels. We reduced the number of differing pixels until the optical neural network could no longer distinguish them. As shown in Fig. 41, the system could separate patterns containing as few as 0.5% differing pixels.

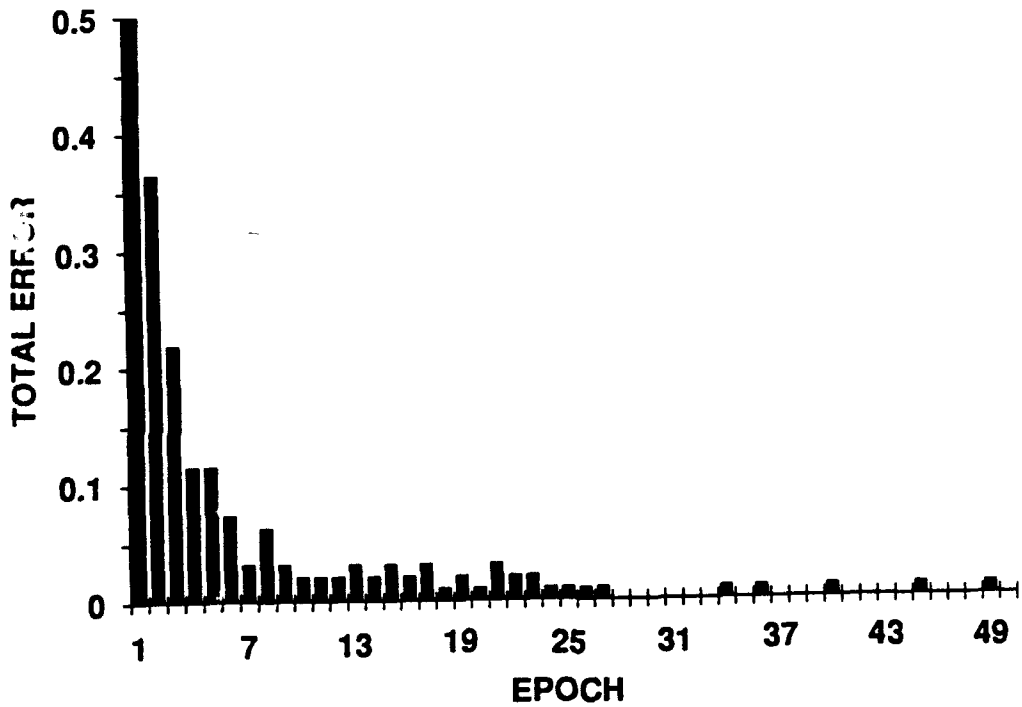



**Fig. 40. Optical Perceptron learning of 96 exemplar patterns with 1920 pixels.**

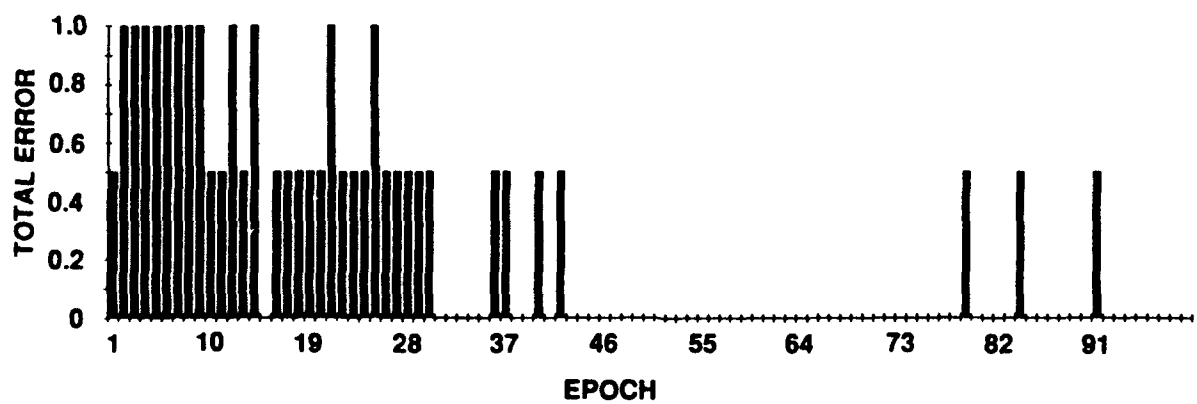



**Fig. 41. Optical Perceptron separation of two nearly identical patterns which have 1920 pixels and differ by 0.5%.**

We were also able to implement multiple-output-neuron Perceptron networks. The networks learned to perform a one-to-one transformation of a given set of random binary patterns (values 1 and 0) into another set of random patterns (values 1 and -1). The results of one experiment are shown in Fig. 42 in which the input and output layers contained 1740 and 870 neurons, respectively. This network had a total of 1.5 million weights. We then scaled this network up to 10,260 neurons and $2x10^7$ weights. The learning curve for this larger network is shown in Fig. 43. The processing rate was $2x10^7$ connections updated per second during learning. This rate was limited by the PC host bus since we had to transfer neuron values back and forth between the image processor cards and host memory. In the next phase of this project we will install an accelerator card with a local bus connection to the image processor cards which should greatly increase the processing rate.

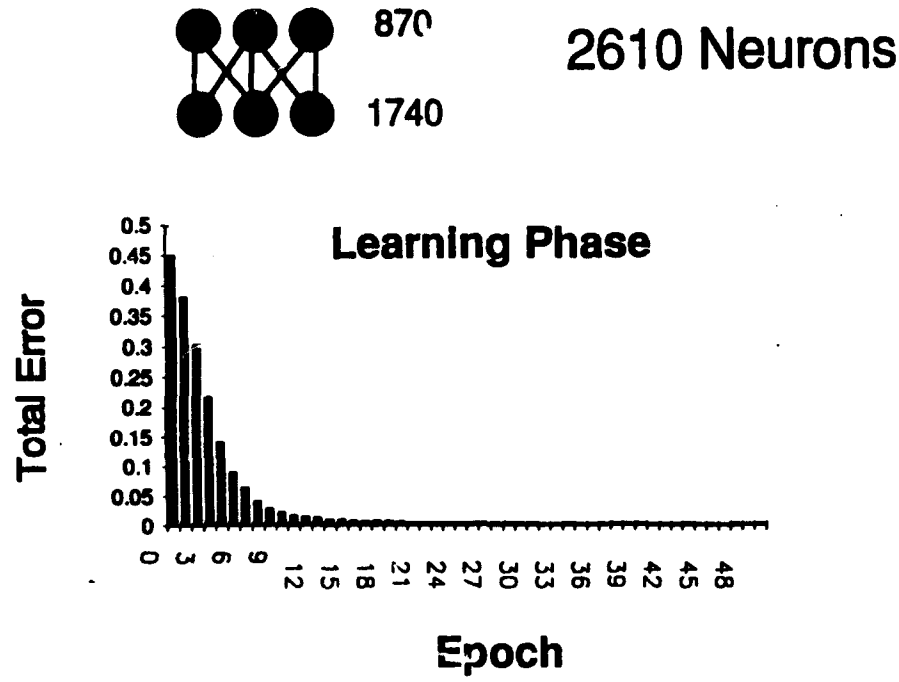



**Fig. 42. Multi-output-neuron single-layer optical Perceptron learning of 4 exemplars. The network consisted of 1740 input neurons and 870 output neurons.**

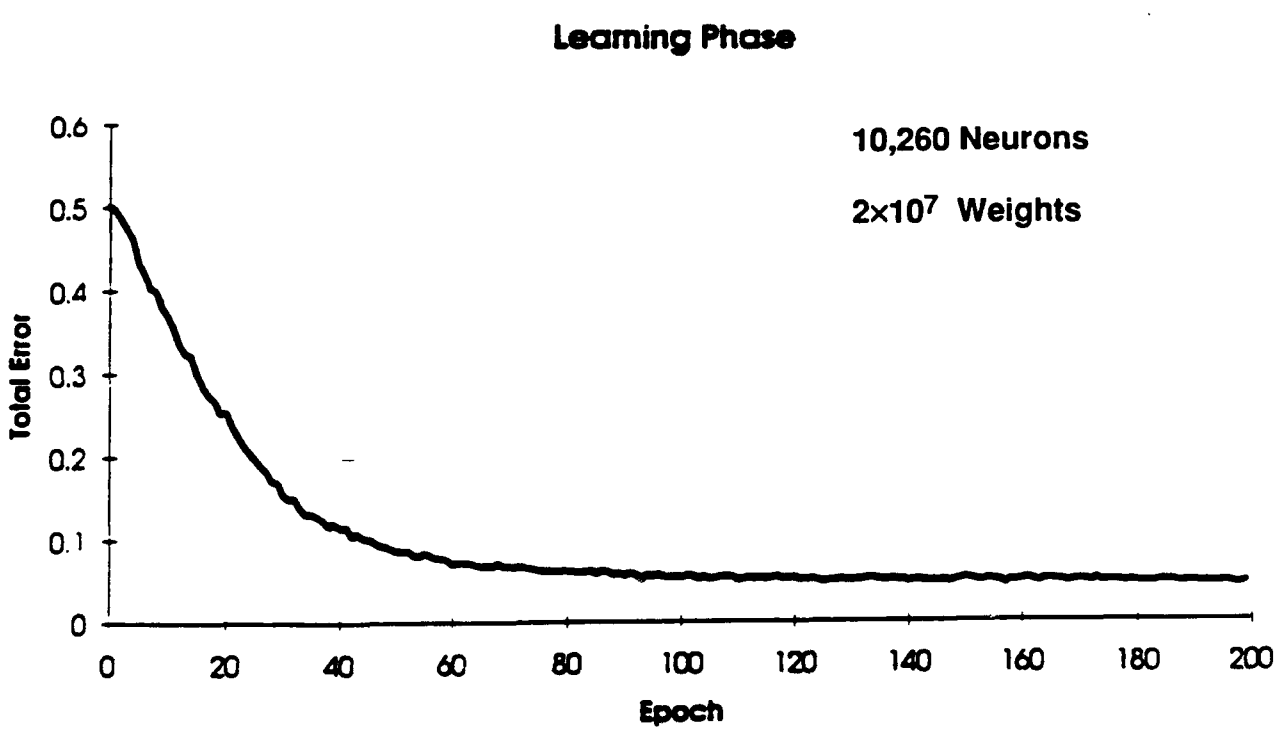
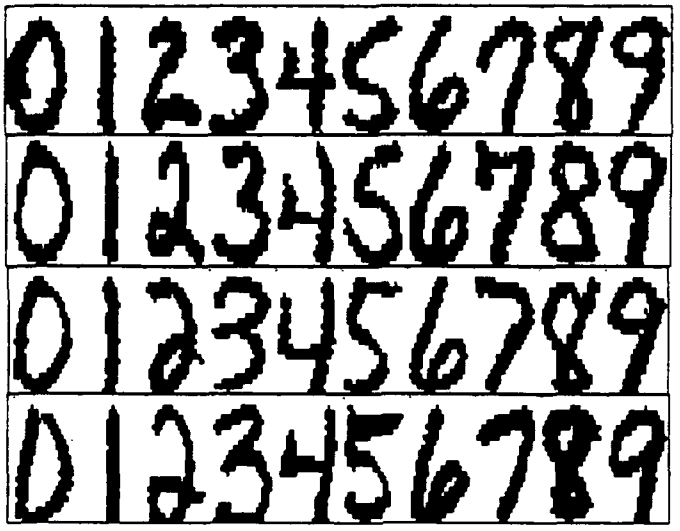



**Fig. 43. Multi-output-neuron single-layer optical Perceptron learning of 4 exemplars. The network consisted of 6840 input neurons, 3420 output neurons, and $2x10^7$ connection weights.**

We also used this type of network in a handwritten digit recognition application. Exemplars were extracted from a database of handwritten digits supplied by the U.S. Post Office. Examples of the digits are shown in Fig. 44. The network was trained to label inputs as one of the ten digits. The total error during learning of 80 exemplars is also shown in Fig. 44. We did not expect good performance because this network contains a single layer of weights and the character recognition problem is not linearly separable. The network achieved an 8% error rate on the training set and a 25% error rate on 40 test digits it had not seen before. Although this was not good enough for practical use, it did demonstrate generalization and learning since the expected untrained error rate is 90%.

# Handwritten Digit Recognition Using Optical Neural Network



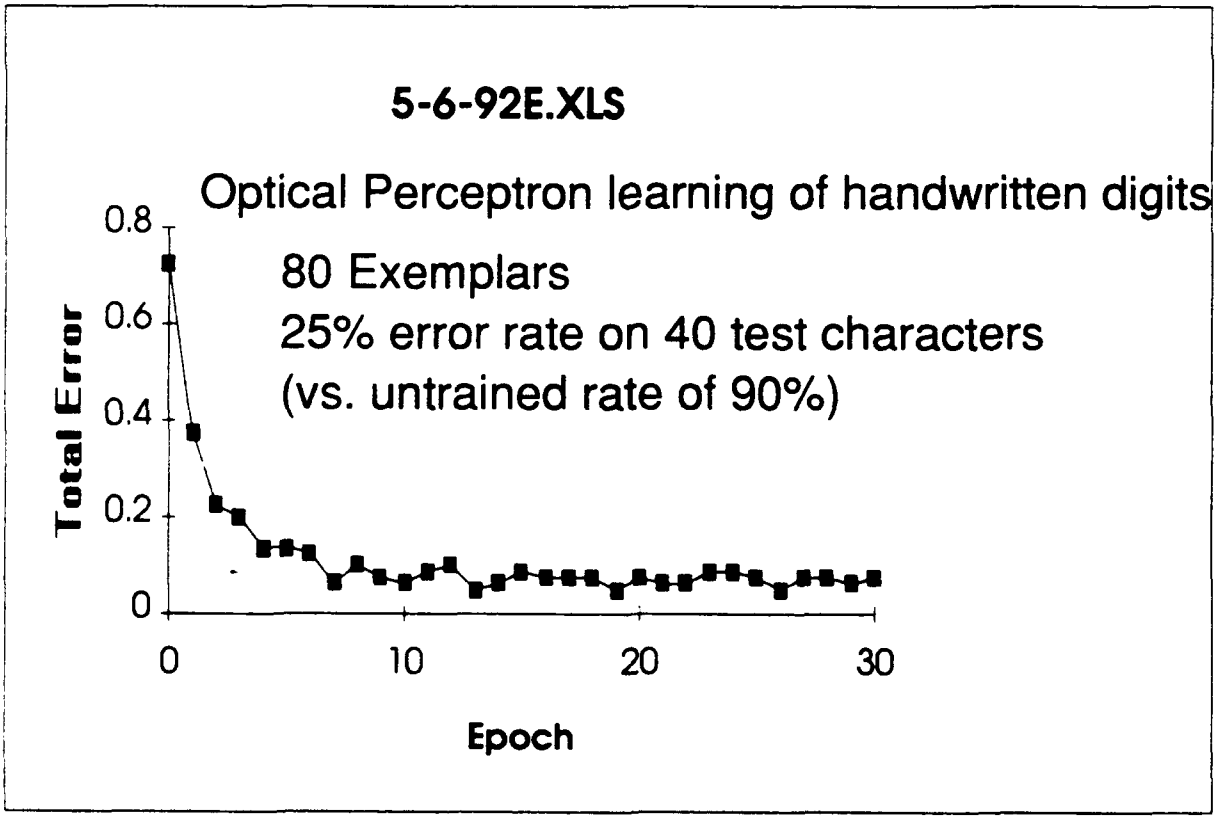




**Fig. 44. Handwritten digit recognition using optical Perceptron network.**

## 7.2 BIDIRECTIONAL ASSOCIATIVE MEMORY

The Bidirectional Associative Memory (BAM) neural network is a type of hetero-associative memory with some generalization or learning properties. Heteroassociative means that when a partial or distorted version of a stored pattern is input to the network, the stable output is a complete undistorted version of a stored output pattern that is "associated" or originally recorded with the input pattern. The BAM is illustrated in Fig. 45. It consists of two neuron layers $F_A$ and $F_B$. The neuron activation functions are hard thresholds. Patterns activating layer $F_A$ are thresholded, weighted, and transmitted to $F_B$ (bottom up). The resultant patterns are in turn thresholded by $F_B$ and transmitted back down to $F_A$ via the same set of weights (top down). The cycle then repeats. Kosko[27] has shown that the function

$$E(\alpha,\beta) = -\left(\beta^T H - \theta_A^T\right)\alpha - \left(\alpha^T H^T - \theta_B^T\right)\beta$$

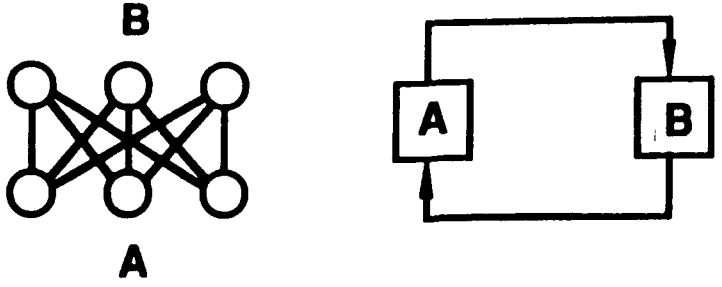
always decreases as the system evolves. In the above expression $(\alpha,\beta)$ are column vectors representing the patterns in $(F_A,F_B)$ and $(\theta_A,\theta_B)$ are the corresponding threshold levels. Since E is bounded from below and it always decreases, it is an "energy" or Lyapunov function that is minimized as the system evolves in time. The minima of E correspond to stable limit points. The only condition on the connection matrix for this to be true is that the weights in the bottom up direction are the same as in the top down direction. This corresponds to having the same weight connecting neurons i and j in both directions (e.g., $h_{ij}=h_{ji}$ or $H=H^T$). Kosko showed that the limit points correspond to stored associated pattern pairs $a^m$ and $b^m$ if the connection weights are given by a sum of outer products:

$$h_{ij} = \sum_{m=1}^{M} a_i^m b_j^m$$

Kosko also showed that BAM is capable of limited learning or generalization from examples.

We implemented the BAM neural network on SPONN by modifying the Perceptron software to divide each of the R and O planes into two sections. R was divided into subsections $R_1$ and $R_2$ and O into $O_1$ and $O_2$. During the readout phase layer $F_A$ for the bottom up direction was represented in $R_1$ and $F_B$ for the top down direction was in $R_2$. In order to always use the fanned R beam to read out the hologram, separate physical connections were used for the bottom up and top down weights, although they were forced to be equal in value as required by the BAM model. During the writing or weight adjustment phase, $F_B$ was represented in $O_1$ and $F_A$ in $O_2$. Thus the bottom up $F_A$->$F_B$ weights were adjusted by turning on $R_1$ and $O_1$ while the top down $F_B$->$F_A$ weights were exposed by $R_2$ and $O_2$. The two sets of weights were exposed simultaneously to ensure their equality.

C9129-08-022




**800 NEURONS, 2 x $10^5$ WEIGHTS, $10^6$ WEIGHTS UPDATED PER SEC**

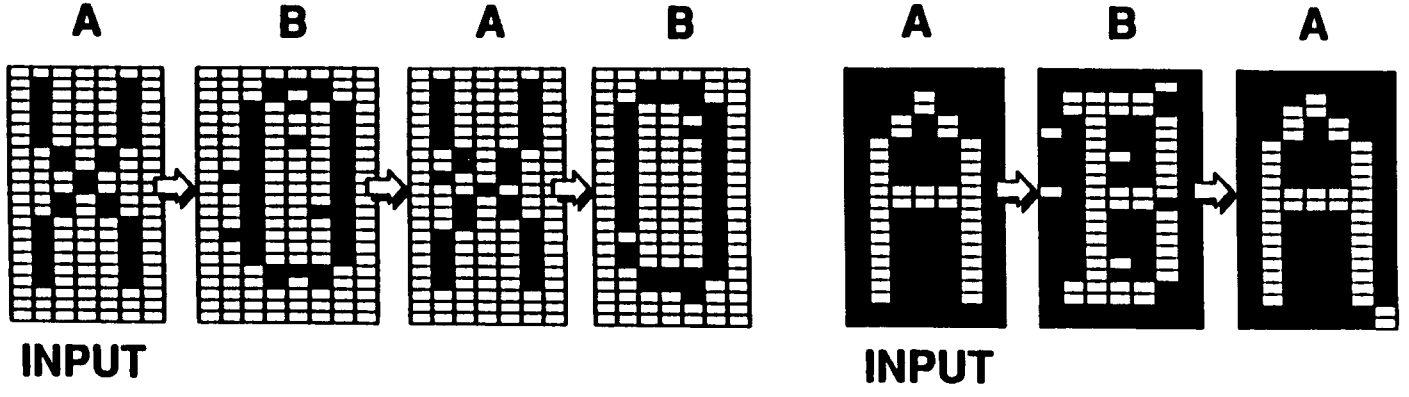



Fig. 45. Demonstration of optical BAMM neural network (hetero-associative memory).

Typical experimental results are shown in Fig. 45 which illustrates the recall of two associated pattern pairs. Addressing the system with a noisy version of stored pattern $\alpha$ resulted in a stable state which cycled between $\alpha$ and its associated pattern $\beta$.

## 7.3 BACKPROPAGATION

Although Perceptron and BAM networks are interesting and useful as tests of neural hardware, their single-layer nature limits their usefulness to separable problems. Adding a layer greatly increases the power of a single-layer network. In fact, it has been shown that a single-hidden-layer network (input, hidden, and output neuron layers with two layers of weights) can approximate any function to arbitrary accuracy provided a sufficient (but finite) number of neurons is available.[28] One of the most popular multi-layer learning algorithms in terms of applications is backpropagation.[29,30] In this section we describe our optical implementation of the backpropagation algorithm.

Backpropagation is based on steepest descent of an error surface defined by

$$E = \sum_c \sum_i \left( y_{i,c}^{(n)} - D_{i,c} \right)^2$$

where E is the total error, $D_{i,c}$ is the desired value for output neuron i given exemplar input c, and $y^{(n)}_{i,c}$ is the actual output for exemplar c. The weight adjustment rule during learning is based on the steepest descent rule:

$$\Delta w_{i,j}^{(n)} = -\eta \frac{\partial E}{\partial w_{i,j}^{(n)}}$$

where the superscript (n) refers to the weight layer being updated. Assuming a two-layer network, and without going into the details of the derivation here, the error gradient is given by the following set of equations for the output layer (see ref. 22 for details):

$$\frac{\partial E}{\partial w_{i,j}^{(2)}} = \delta_i^{(2)} y_j^{(1)},$$

$$\delta_i^{(2)} = (y_i^{(2)} - D_i) g\left(y_i^{(2)}\right),$$

$$g(y) = f'\left[f^{-1}(y)\right] = y(1-y)$$

and by these equations for the hidden layer:

$$\frac{\partial E}{\partial w_{j,k}^{(1)}} = \delta_j^{(1)} y_k^{(0)},$$

$$\delta_j^{(1)} = g(y_j^{(1)}) e_j^{(1)},$$

$$e_j^{(1)} = \sum_i \delta_i^{(2)} w_{ij}^{(2)}$$

where $f(x)=1/(1+e^{-x})$ is the neuron sigmoidal response function.

One possible issue in the optical implementation of backpropagation is its sensitivity to the accuracy of representation of the functions f() and g(). Due to inherent nonlinearities and nonuniformity in present SLMs, f() and g() may differ from the above form. We have performed computer simulations of backpropagation with slightly different f()s and g()s. We found that although the network is robust with regard to changes in f(), performance degrades if g() departs from its proper dependence on f(). Therefore, in our optical implementations we measured the SLM nonlinearity and electronically compensated for it so that g() could be implemented accurately.

In the optical system we actually implemented a variation of backpropagation in which the input error signals were trinary quantized to +1, 0, and -1 according to the algorithm reported by Shoemaker et al..[31] They found that trinary quantization improved the convergence speed of backpropagation for a wide variety of problems. Our own computer simulations confirmed this. However, our primary reason for trinary quantization was to avoid amplitude-dependent phase errors in the LCLV.

The optical input plane for backpropagation is shown in Fig. 46. The network consisted of three neuron layers ($L_0$, $L_1$, and $L_2$) and two weight layers. Note that the $L_1$-$L_2$ weights are actually implemented as two separate sets of weights, one for the forward pass $L_1$ - > $L_2$ and another one for the backward pass $L_2$ -> $L_1$. As explained previously, this is done because the fanned reference beam (R plane) is always used to read out the unfanned object beam (O plane). In order to implement forward and backward passes through the same set of effective weights, two sets of photorefractive weights must be exposed. They are exposed so as to make the forward and backward weights as nearly equal as possible (symmetric connections, $w_{ij}=w_{ji}$) although they can also be made different (asymmetric connections) if required for certain networks. Although the $L_1$ -> $L_2$ and $L_2$ -> $L_1$ sections of the input plane are shown spatially separated for clarity, in actuality they are spatially interleaved in order to make the connections as symmetric as possible. The solid arrows indicate optical connections between neurons via the hologram. Dashed arrows denote electronic transfer of detected outputs from one layer (O plane) to the inputs of the next layer in the R plane. This electronic operation is an order N operation while the optical connection is order $N^2$,

where N is the number of neurons. The speedup factor over a single electronic processor is therefore proportional to N.

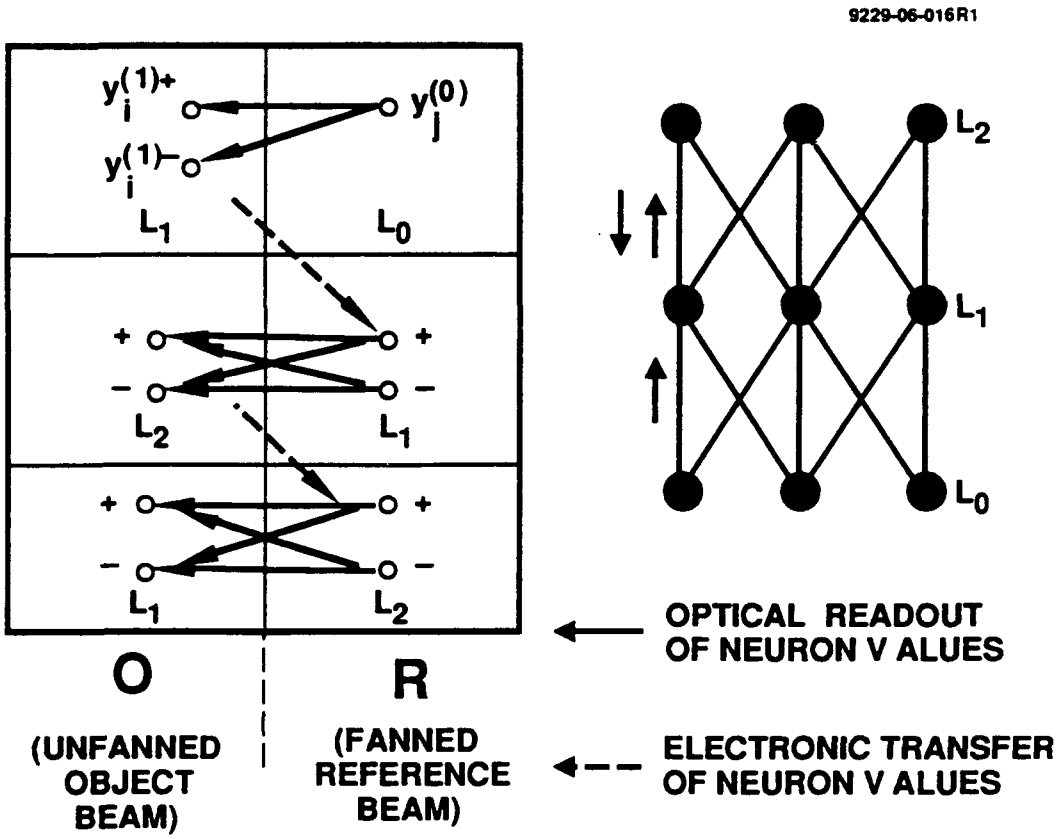



**Fig. 46. Optical input plane for optical backpropagation with a single hidden layer.**

A flow diagram for the optical backpropagation algorithm is shown in Fig. 47. The forward and backward weights between layers $L_1$ and $L_2$ were adjusted simultaneously in order to keep the weights as symmetric as possible. This also caused self-connections from each neuron to itself to be formed. The self-connections do not, however, affect the operation of the backpropagation network. As described in the previous section on the bipolar algorithm, the $L_1$ to $L_2$ weights are exposed twice [$(L_1)(L_2)$ and $(-L_1)(-L_2)$] in order to ensure that $w_{ij}^{++}=w_{ij}^{--}=w_{ij}^{+}$ and $w_{ij}^{+-}=w_{ij}^{-+}=w_{ij}^{-}$.

Experimental results for the problem of transforming one random binary pattern into another using optical backpropagation are shown in Fig. 48 where we plot the total output error versus epoch during learning of two pattern transformations. In this case the network contained 252 neurons arranged in three layers as 128-62-62. The output error decreased to 5% after 30 epochs and reached zero after 90 epochs. In order to track the evolution of the output pattern for each of the two exemplar inputs, we also plotted the projection of the output pattern vector on the

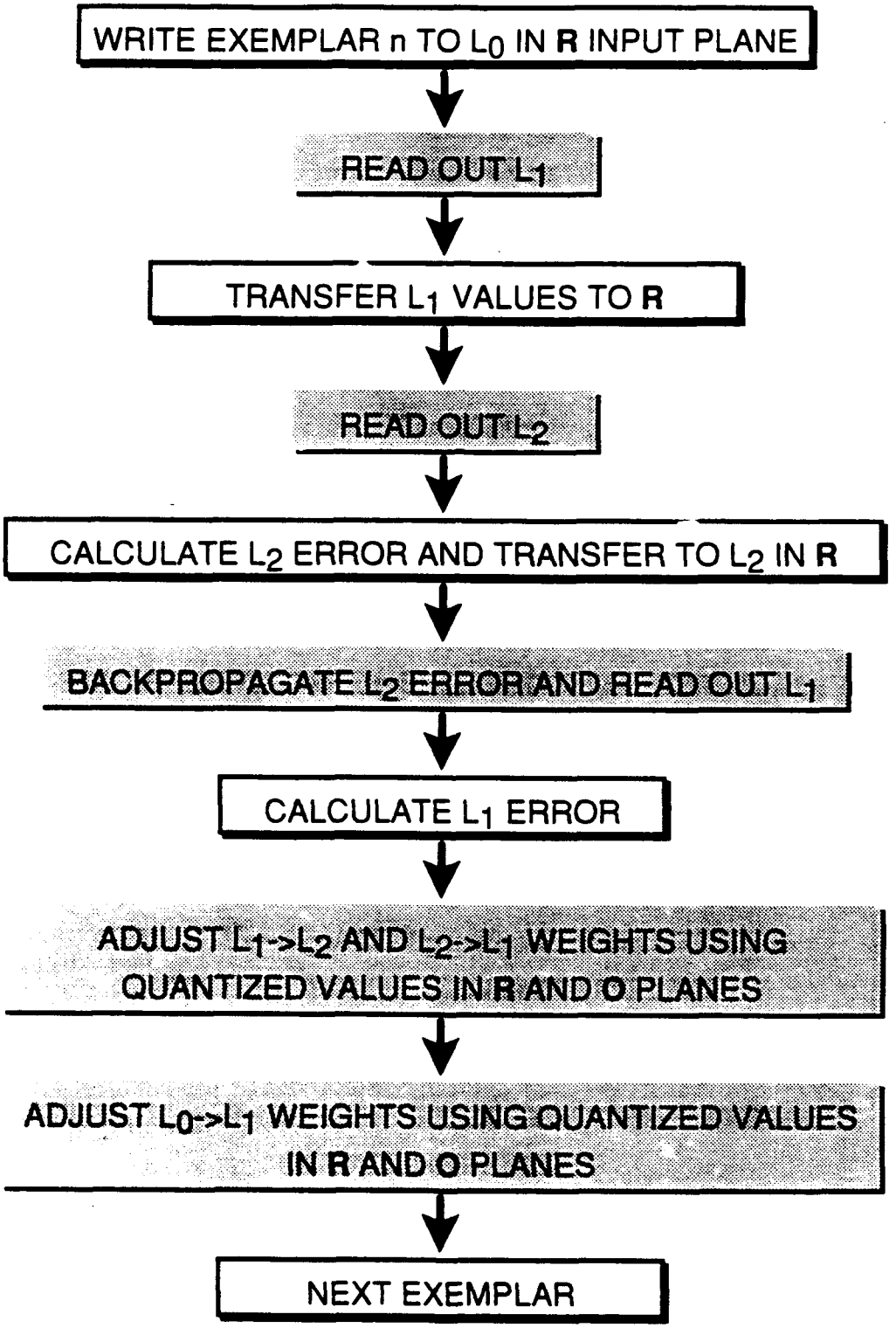



**Fig. 47. Flow chart for optical backpropagation program.**

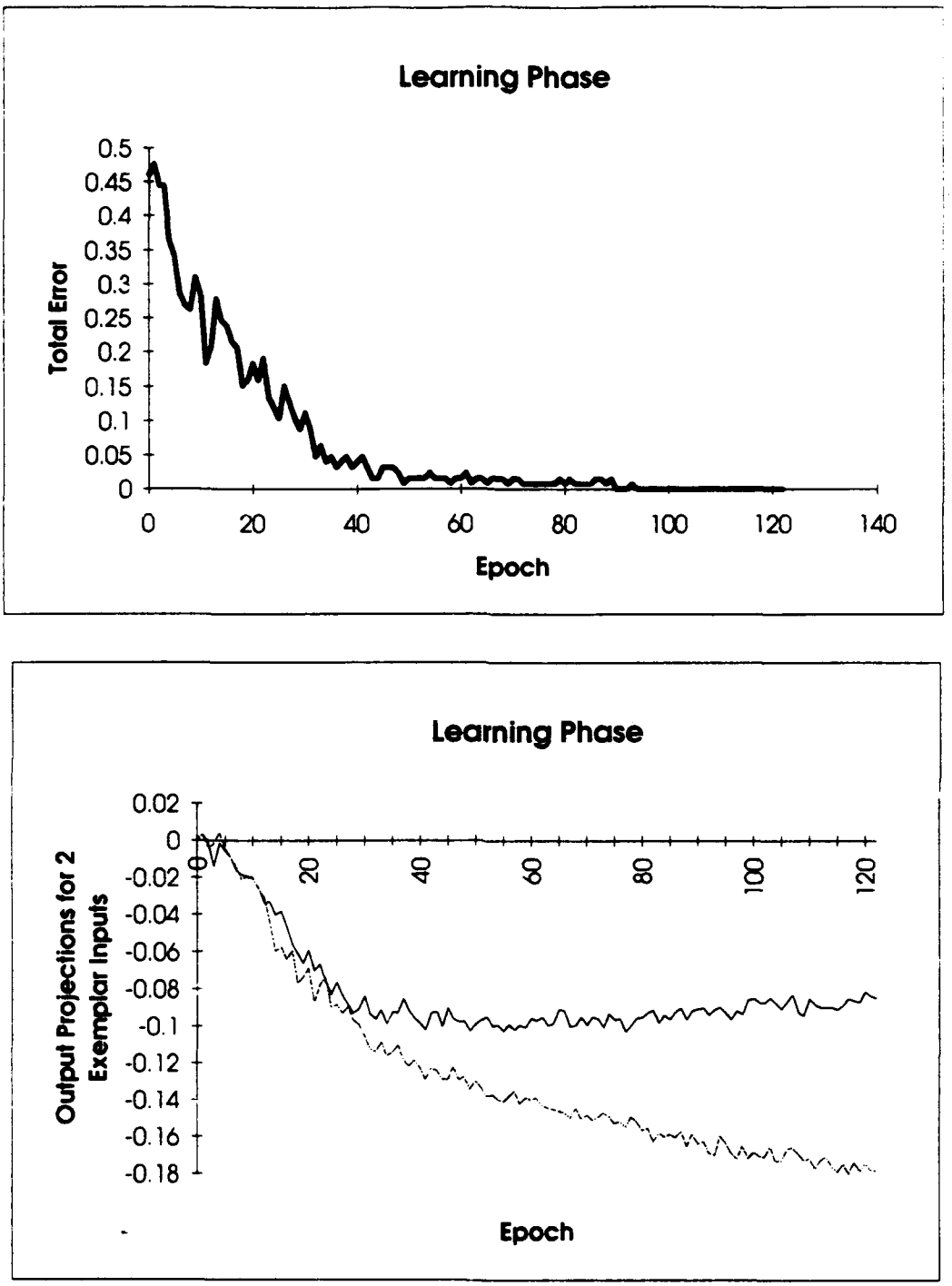
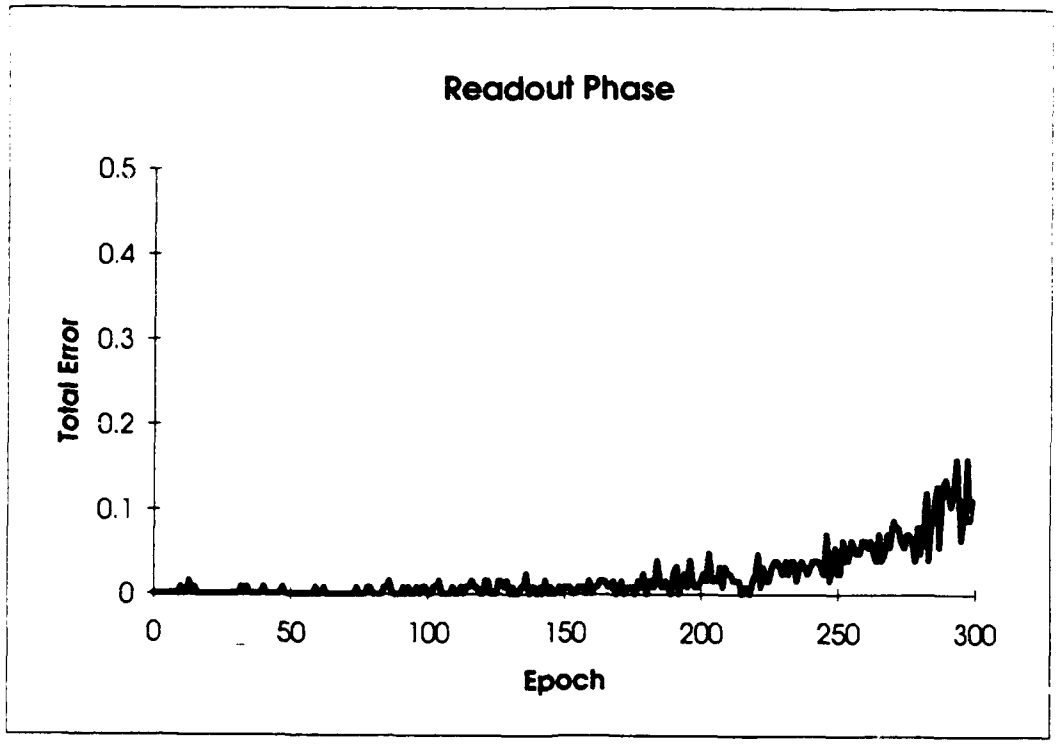



**Fig. 48. Optical backpropagation learning of two pattern transformations (see text).**

reference vector (1,1,1,...,1). We can clearly see the buildup of the weights from small initial values to saturation. In Fig. 49 we plot the total error versus epoch after learning is completed and the network is continuously read out. Due to the weight decay discussed previously, the errors began to increase dramatically after 200 readouts of the network. The weight decay is clearly seen in the simultaneous plots of the output pattern vector projections. Note that the output error does not increase until after the weights have decreased by a large amount, indicating the network has learned a "safety margin" which makes it less sensitive to weight decay. Analogous results are shown in Fig. 50 for the same network learning four pattern transformations.

Results for larger backpropagation networks are shown in Fig. 51, one with 320 neurons distributed among three layers as 160-80-80 (20,000 weights) and another with 1140 neurons distributed as 570-285-285 (244,000 weights). The first network converged to approximately 3% error after 50 epochs while the second one reached 6% error after 100 epochs. In both cases the exemplar set consisted of two patterns. In these initial backpropagation results the performance degraded for eight or more exemplars. As of this writing we are still investigating why this is so. Possible reasons include residual inaccuracy in representing g() and excessive laser noise.



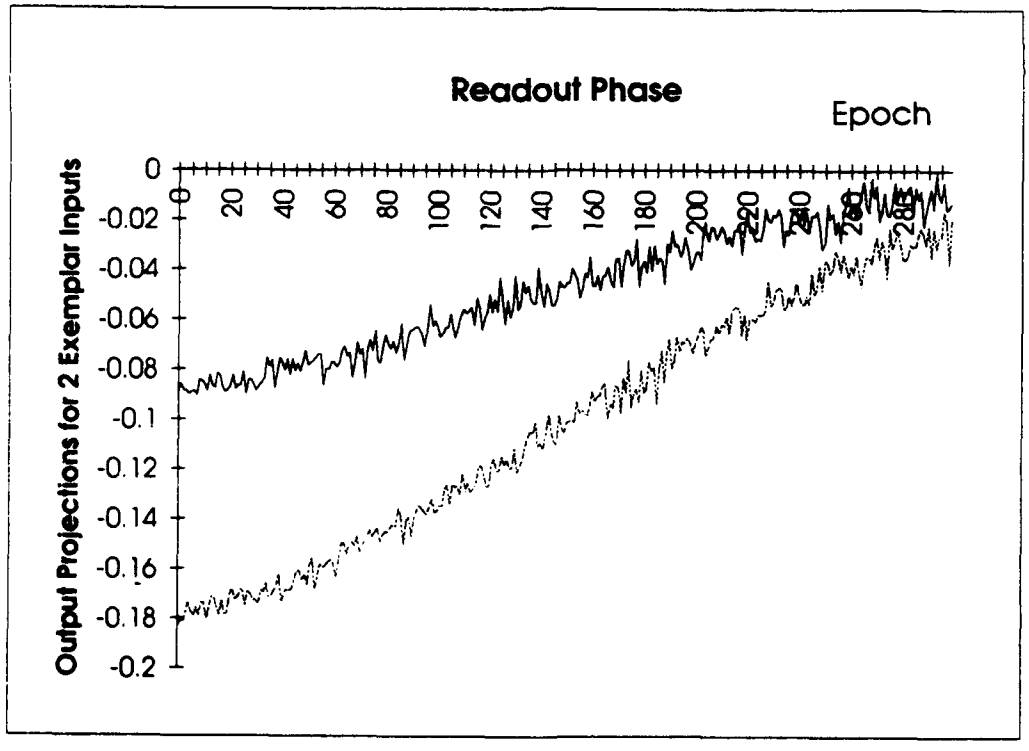
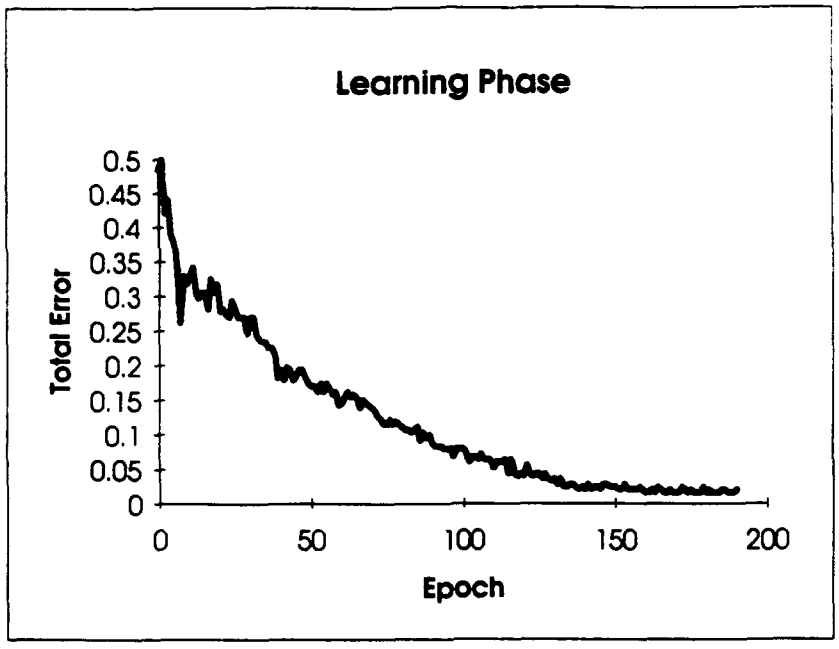




**Fig. 49. Continuous readout of optical backpropagation network after learning has been completed.**



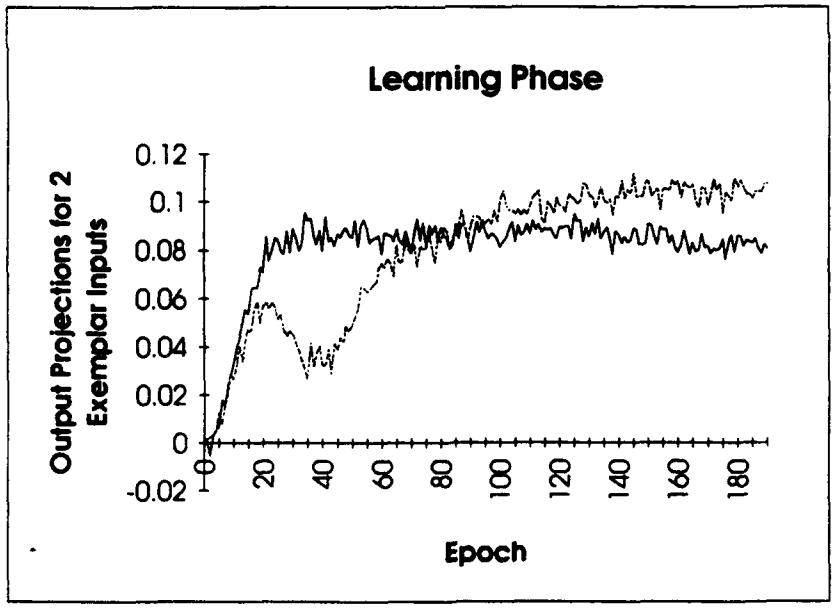




**Fig. 50. Optical backpropagation learning of four pattern transformations (see text).**

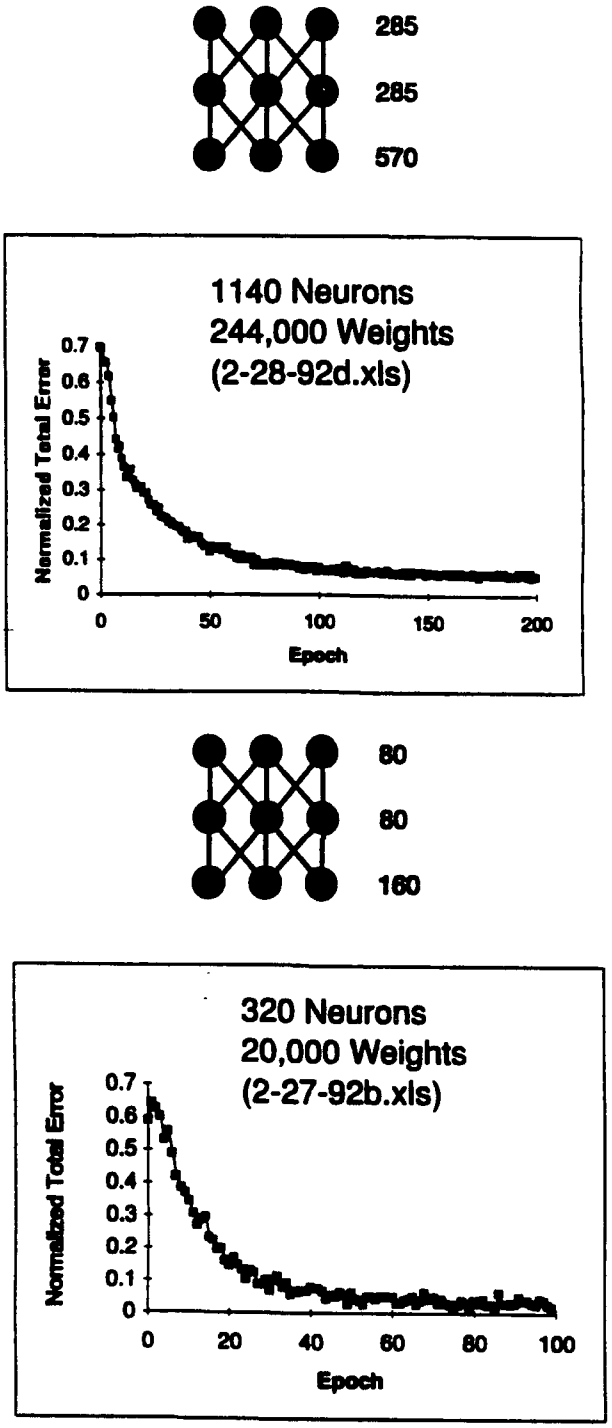



**Fig. 51. Optical backpropagation learning of two pattern transformations. Results for two networks are shown, one with 320 neurons and the other with 1140 neurons.**

# SECTION 8

# PERMANENT RECORDING OF WEIGHTS IN PHOTOREFRACTIVE CRYSTALS

## 8.1 HOLOGRAM FIXING TECHNIQUES

The refractive index variations in photorefractive materials that are normally used for real-time holography, phase conjugation and optical signal processing are produced by internal electric field variations associated with charges trapped at mid-gap levels. These holograms are generally not permanent under illumination with light at the wavelength used to write them because this light continues the writing process. Thus illumination with an interference pattern of a new period both erases the old grating and writes a new one, similarly illumination with uniform light erases the old grating. In order to obtain a fixed hologram in a photorefractive material the internal space-charge pattern must be transferred to a charged defect or dopant level that is not optically active at the wavelength of interest. Two methods that can be used to do this *in situ* are thermal and electrical activation. In this section we review the method we pursued to obtain fixed holograms.

From a systems point of view, the most attractive hologram fixing technique involved the application of an external electric field in order to transfer a normal photorefractive hologram in the photosensitive deep levels to a space charge hologram in an insensitive or non-sensitive levels. The generic process involves the steps illustrated in Fig. 52. Initially, the crystal of $BaTiO_3$ is polled with a field $E_1$, shown in the upper plot. At time $t_1$-$t_2$, lower plot, a hologram is written with the usual photorefractive process. An electrical field $-E_2$ is applied at $t_2$. A mirror image (180° phase shifted) space-charge pattern that nearly cancels the photorefractive hologram is produced in charges that are immobile and optically insensitive at room temperature. At time $t_3$, the photorefractive hologram is erased leaving only the mirror image hologram that cannot be erased with optical radiation alone. At time $t_4$ a field $E_3$ is used to reset the crystal to its initial condition, erasing the fixed hologram. Note: in some cases it may be advantageous to perform two or three of these processes at the same time.

In materials with rather low coercive fields (on the order of 1 kV/cm) such as $BaTiO_3$ and $Sr_{1-x}Ba_xNb_2O_6$ (SBN), electric fixing with domain reversal has been discussed in the literature. The coercive field is the critical applied electrical field required to reverse the spontaneous polarization of a ferroelectric crystal. It has been suggested that by applying a field with an amplitude just below the coercive value and with an orientation opposed to the orientation of the existing polarization, a spatial pattern of polarized clusters may be produced. If this operation is

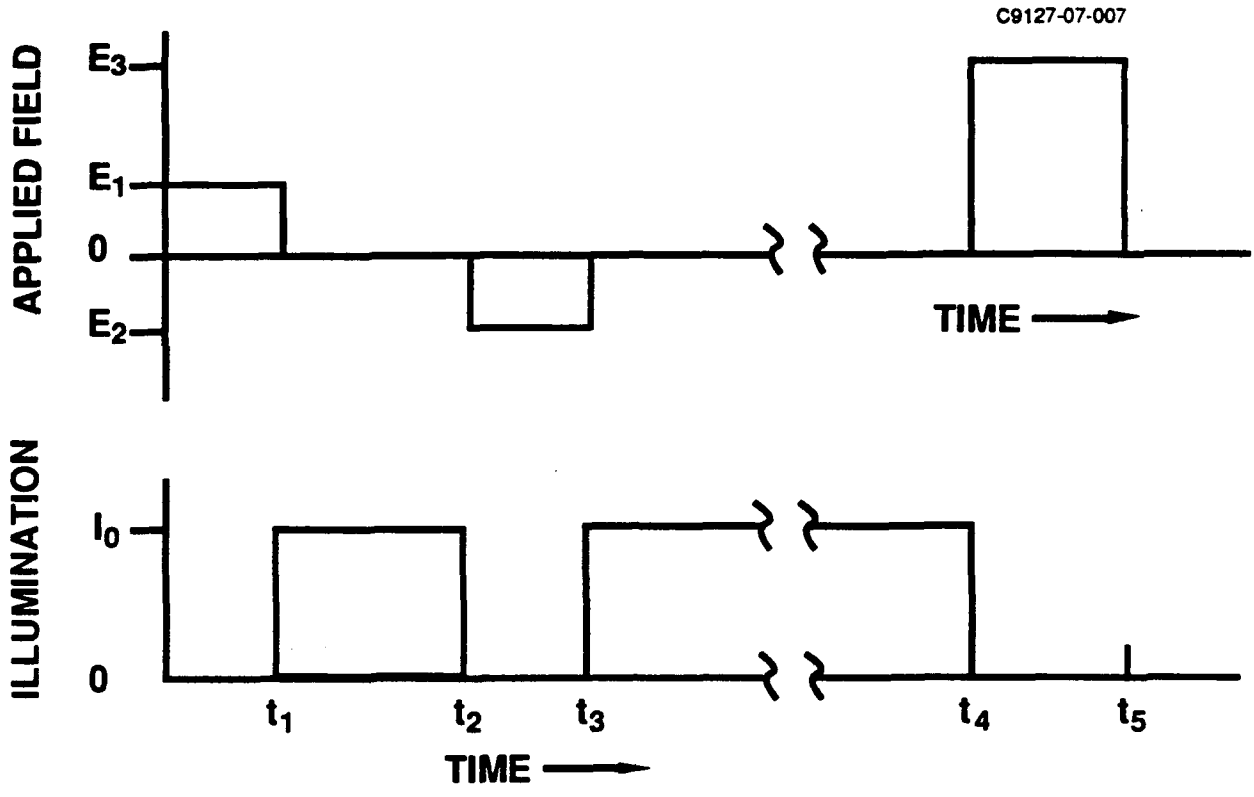



**Fig. 52. Waveforms used in hologram fixing experiments.**

performed after the recording of a hologram, the cluster pattern may mirror the recorded hologram. Holograms fixed using domain reversal should not be erased optically, but application of a strong poling field will restore the initial "blank" state in the crystal.

Demonstration of this electrical fixing technique at room temperature was discussed by Micheron and Bismuth in $BaTiO_3$. Several research groups have tried to reproduce the data, however, at the time of this report, have not been able to observe the published results.

In order to explain the electrical fixing and erasing of holograms in $BaTiO_3$ and SBN, Micheron and coworkers assumed that with an applied field $E_2$ of reversed polarity, ions or vacancies drift under the influence of the internal field $E_i$ caused by the photoelectrons trapped during the writing step and clusters or domains of reversed polarity are formed. The field $E_2$, however, is not large enough to excite photoelectrons. With $E_2$ applied, fast ionic displacements occur that cancel the spatial modulation of $E_i$. The diffraction efficiency drops to zero. Under uniform illumination, photoelectrons are again photoexcited and diffuse uniformly, thus leaving behind an uncompensated ionic pattern and revealing the original hologram. This pattern is stable during readout and constitutes the fixed hologram. Erasure is achieved when the applied field is high enough to restore uniform ionic distributions. This explanation has been based on the hypothesis that domain patterns replicating the written hologram could be induced in both $BaTiO_3$

and SBN by application of the field $E_2$. This hypothesis accounts for the erasure process. Their theory was partially supported by two experimental observations. First, photoinduced clusters could be written at temperatures above the Curie temperature in SBN and fixed by cooling the crystal (Micheron and Trotier 1974), thus establishing a link between the occurrence of frozen clusters and antiparallel domains. Moreover, holographic fixing in SBN is only observed when the voltage $E_2d$ (where d is the electrode spacing) is larger than the coercive voltage Vc = 775 V. This is again consistent with the picture of domain reversal. It remains, however, to be explained why the same authors did observe domain reversal in $BaTiO_3$ below the coercive voltage.

## 8.2 MEASUREMENTS OF $BATIO_3$ ELECTRICAL PROPERTIES

In this section we present an overview of some of the experiments we conducted with the goal of electrically fixing photorefractive holograms. In order to characterize the state of our crystals prior to attempting to fix the holograms, we made measurements of the electrical properties, including a determination of the coercive field for each sample.

$BaTiO_3$ and SBN are ferroelectric crystals because they have two orientational states of spontaneous polarization in the absence of an electric field and can be shifted from one to another of these states by applying an electric field. An as grown crystal can have different orientations of these domains throughout the crystal. By applying first a mechanical and then a sufficiently large electric field with the correct orientation, all of the domains can be made to align with one another. In our holographic measurements, we initially start with a single domain crystal and write a grating by interfering two optical beams. We then applied various electrical waveforms to the crystal in an attempt to create a pattern of domains which mimic the distribution of the light intensity.

The most important characteristic of a ferroelectric crystal is that the relationship between P and E is represented by a hysteresis loop. Ferroelectric hysteresis loops can be observed with the circuit shown in Fig. 53, which is patterned after one first described by Sawyer and Tower. An a.c. field is generated by a frequency synthesizer or D/A board in a computer, and amplified by a high voltage amplifier. The crystal is placed in series with a capacitor $C_1$ which has a capacitance significantly larger than that of the crystal. The applied field is monitored with a high voltage probe. The voltage drop across the capacitor $C_1$ is proportional to the charge which accumulates on the crystal.

An ideal plot of the charge or polarization of the crystal versus the applied field is shown in Fig. 54. This plot clearly displays a hysteresis loop. The polarization initially increase as a function of increasing applied filed and reaches a saturated value, essentially independent of the field. When the field is decreased, the value of the polarization does not re-traverses the initial curve. Some of the domains remain aligned in the positive direction and the value for zero field is

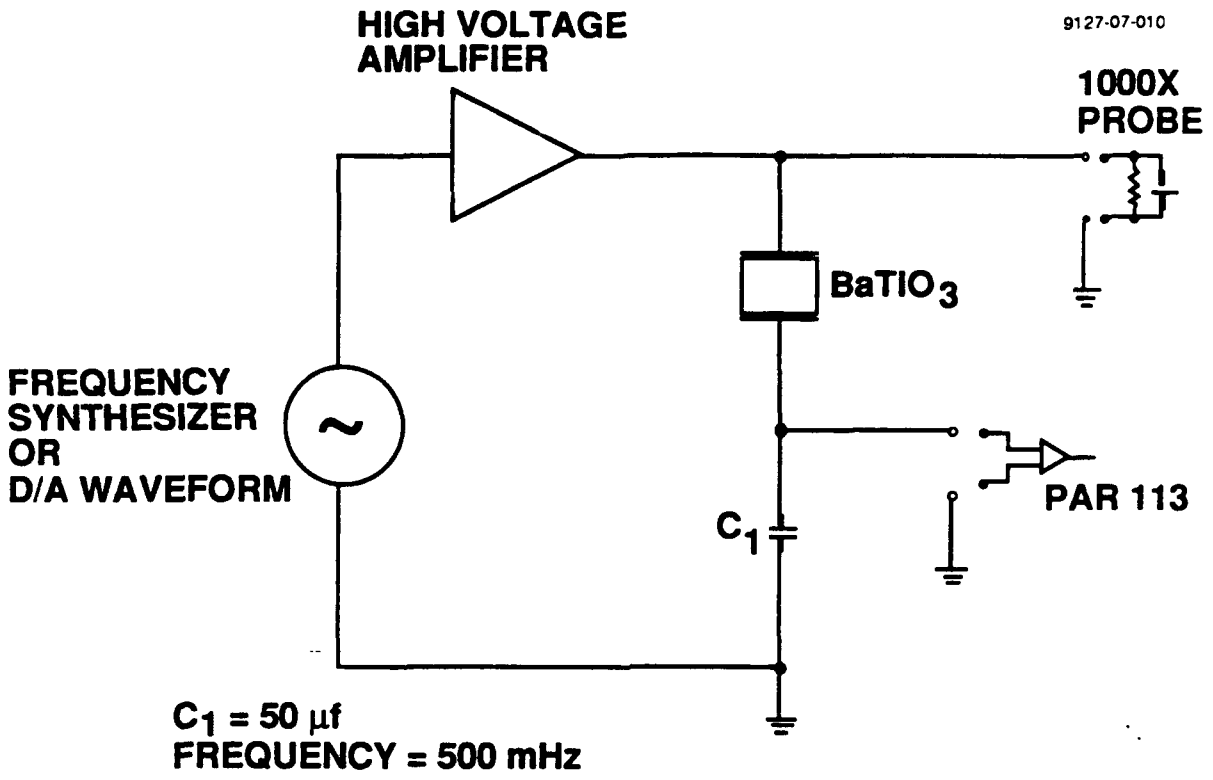



Fig. 53. Circuit for observing hysteresis loop.

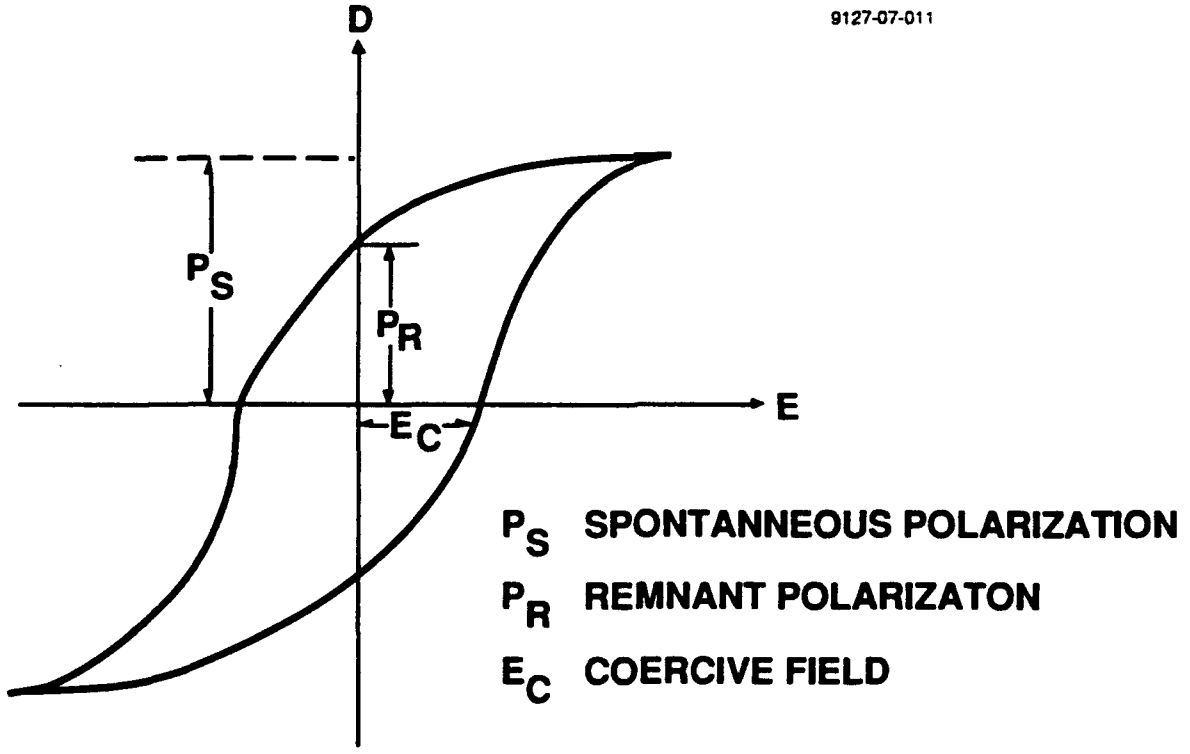



Fig. 54. Ideal hysteresis loop.

called the remnant polarization, $P_s$. The value of the spontaneous polarization, $P_s$, can be determined by extrapolating of the linear upper portion of the curve back to the polarization axis.

In order to remove the overall polarization of the crystal it is necessary to apply an electric field in the opposite or negative direction. From this graph, one can measure the coercive field, $E_c$, the field required to reduce P to zero. Further increase of the negative field eventually causes almost all of the dipoles to align in the direction opposite to that when the saturated positive field was applied.

Experimentally obtained hysteresis loops for one sample of $BaTiO_3$ are shown in Fig. 55. The graph is qualitatively representative of the different crystal samples, and shows two individual hysteresis loops which essentially overlap each other. These loops were qualitatively similar to graphs obtained for samples of SBN. We obtained close to two thousand hysteresis curves from one of our samples of $BaTiO_3$. The initial curves, yielded a value of 1.3 KVolts/cm for the coercive field and $2.5 \times 10^{-5}$ coul/cm$^2$ for the spontaneous polarization.

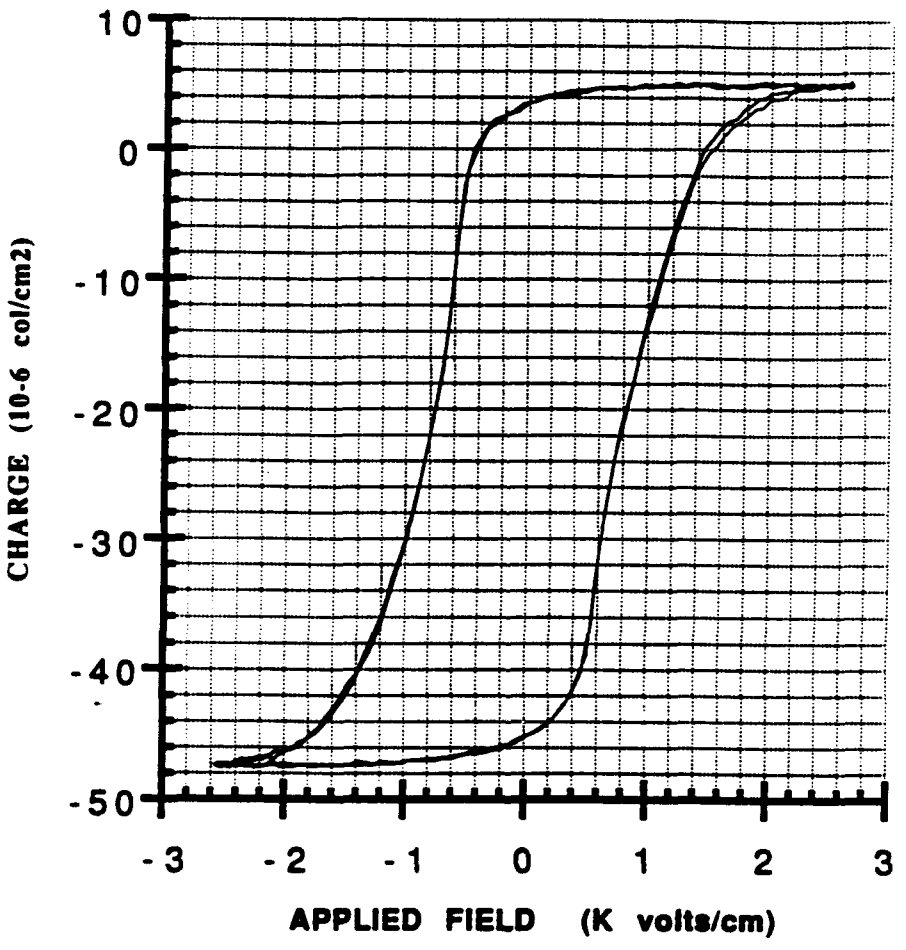



**Fig. 55. Measured hysteresis loop of $BaTiO_3$ crystal.**

However, we found that after cycling the same crystal through several thousand hysteresis loops, the crystal could not attain the same degree of polarizability as when it was first grown and thermally polled. This implied that some of the domains were being pinned in one orientation. Also, after taking the hysteresis measurements we found that the crystal faces to which the electrodes were attached had been damaged. The crystal positive c-face was crazed and this could have impacted the magnitude and spatial uniformity of the applied field.

## 8.3 EXPERIMENTS IN ELECTRICAL FIXING OF $BATIO_3$

In this section we present a review of the experiments we conducted to determine the effect of applying electrical fields on photorefractive crystals after a photorefractive grating was written in the crystal. These series of experiments were designed to fix the photorefractive holograms. A summary of the parameter space we investigated is shown in tabular form in Table 1. Our group and others, in particular at Stanford University and the University of Southern California, have not been able to reproduce the room temperature electrical fixing results published by Micheron and Bismuth.

The state of the photorefractive hologram can be determined by monitoring the two writing beams and the diffraction efficiency of a probe beam. Typically, the probe beam is incident on the crystal at the Bragg angle, and two beams emerge from the other side of the crystal: the diffracted and undiffracted beams. The absolute diffraction efficiency of a grating, $\eta_{abs}$, is the ratio of the diffracted intensity to the incident intensity. The relative diffraction efficiency, $\eta_{rel}$, is the ratio of the diffracted intensity to the sum of the diffracted intensity plus the undiffracted intensity. One can convert between the two efficiencies with the following formula: $\eta_{abs}=\eta_{rel}t^2e^{-\alpha L}$, where t is the interface transmission and $\alpha$ is the absorption coefficient.

The optical train that we used in our experiments is shown schematically in Figure 56. We constructed our optical train in a manner which provided a high degree of mechanical and thermal stability. The output of the argon ion laser passed through a half-wave plate/polarizer combination as a means for controlling the overall beam intensity. The beam was spatially filtered and collimated in order to produce a uniform intensity pattern at the crystal. A portion of the beam transmitted through the first beam splitter (BS1) was used to monitor the total power in the pump and probe beams. The reflected beam was used to generate the two writing beams using a second beam splitter (BS2). The two beams, designated as "A" the pump and "B", the probe, were aligned to interfere in the crystal. The transmitted pump and probe beams were monitored by the detectors labeled "A" and "B", respectively, in the figure. The detectors were used to monitor the transfer of energy, or two-beam coupling gain, while the gratings were written in order to determine the photorefractive grating strength and stability. These detectors had interference filters (514 nm) in front of them to improve the signal to noise ratio. The probe and pump beams were

TABLE 1. Status of Electrical Fixing of Photorefractive Holograms in $BaTiO_3$.

| $BaTiO_3$ CRYSTAL | MICHERON & BISMUTH | HUGHES I | HUGHES II | HUGHES III |
|---|---|---|---|---|
| Growth Technique | Remeika | Top Seed Sol. Growth | Top Seed Sol. Growth | Top Seed Sol. Growth |
| Crystal Dimensions | $2x3mm^2x100\mu m$ | $5x5x5\ mm^3$ | $5x5x5\ mm^3$ | $5x5x5\ mm^3$ |
| **WRITING OF HOLOGRAM** | | | | |
| Wavelength | 4880Å | 5145Å | 5145Å | 5145Å |
| Write Power | $100\ W/cm^2$ | $0.1\ W/cm^2$ | $0.1\ W/cm^2$ | $0.1\ W/cm^2$ |
| Write Time | 0.5 sec | 10 sec | 10 sec | 10 sec |
| Pump/Probe Ratio | ? | 1:1 | 1:1 | 1:1 |
| Pump-Probe Angle | 30° | 30° | 30° | 30° |
| Pump & Probe Polar. | p-polarised | p-polarised | p-polarised | p-polarised |
| Monitor wavelength | 6328Å | 6328Å | 6328Å | 6328Å |
| HeNe Power | 5 mW | 4.7 μW | 4.7 μW | 4.7μW |
| Diff. Efficiency Max. | 2.2% | 1.7% | 1.7% | 2.5% |
| Diff. Efficiency Fix | 1.0% | N.A. | N.A. | N.A. |
| **ELECTRICAL FIXING OF HOLOGRAM** | | | | |
| Coercive Field | 1.1 Kv/cm | 1.3 Kv/cm | 1.3 Kv/cm | 1.3 Kv/cm |
| Poling Field | >4 Kv/cm | 3.7 Kv/cm | 3.7 Kv/cm | 3.7 Kv/cm |
| Poling F. Waveform | Square | Sine | Sine | Sine |
| Poling F. Duration | ? | 2 Sec | 2 Sec | 2 Sec |
| Fixing Waveform | Square | Square | Half Sine | Square |
| Fix Field Duration | 0.1 Sec | 0.1 Sec | 1 Sec | 1 Sec |
| Fixing Field (Kv/cm) | -1.0 | -(0.38-3.0)* | -(0.46-1.9)** | -(.76-2.1)*** |
| When Field Applied | After Write | Prime After Write | Prime After Write | During Write |
| Reveal Illumination | Reference Beam | Ref. & Incoherent | Ref. & Incoherent | Incoherent |
| Erasing Field | >10 Kv/cm | N.A. | N.A. | N.A. |
| Erase Field Waveform | Square | N.A. | N.A. | N.A. |
| Erase Field Duration | 3 Sec | N.A. | N.A. | N.A. |

* 5 Hz Square wave, produced 100 ms unipolar pulse, using amplitude (A) and offset (A/2) Hewlett Packard 8116A function generator. Volts/cm at crystal: 380, 571, 761, 952, 1142, 1523, 1904, 2284, 2665, 3046.

** 500 mHz, positive half sine wave generated by Data Translation DT2831-G D/A board, file PHSXPY (x.y volts) into high voltage amplifier. Volts/cm at crystal: 459, 547, 952, 1285, 1428, 1523, 1713, 1808, 1904.

*** 500 mHz Square wave amplitude (A) with d.c. offset (A/2), produced 1 sec square unipolar pulse, Hewlett Packard 8116A function generator. Volts/cm at crystal: 761, 952, 1142, 1332, 1523, 1904, 2094.

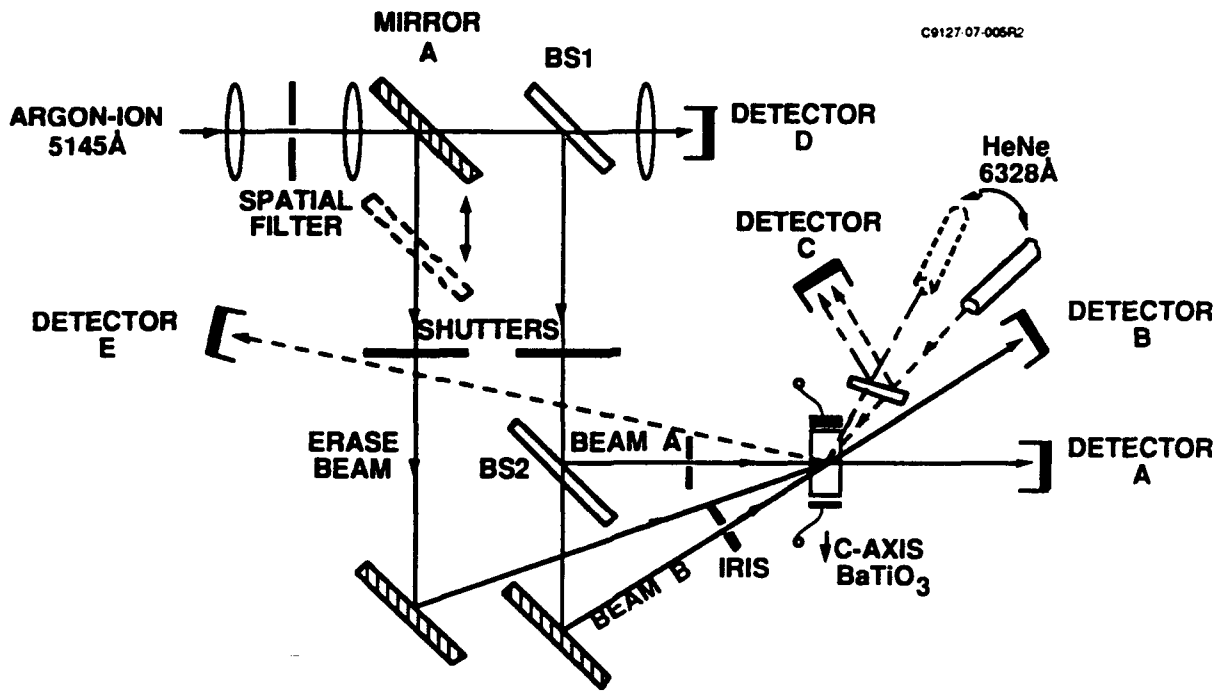



**Fig. 56. Apparatus used in hologram fixing experiments.**

comparable in both transverse dimension and flux. An electronic shutter in the beam path was used to control the crystal exposure time.

Mirror "A", located after the spatial filter, could be moved into the beam path to generate an erase beam. This beam could be used to uniformly illuminate the crystal at an angle other than the Bragg matched condition for erasing the photorefractive gratings. Note: if one of the writing beams were used to erase the grating, new gratings could be written because the diffracted signal would interfere with the original incident "erase" beam. A second electronic shutter was used to control the illumination time of the crystal to the erase beam. In our system the crystal could also be illuminated with an auxiliary incoherent source to erase the photorefractive gratings. The advantage of using the incoherent source is that it produced a spectrum of wavelengths and cannot produce any interference effects.

A HeNe laser operating at 0.6328 μm was used to generate a probe beam which allowed us to monitor the evolution of the gratings in the crystal during the entire writing process. Note: the longer 0.6328 μm wavelength in conjunction with a significantly reduced flux (~5 μW) insured that there was minimal erasure produced by this monitoring technique. In order to simplify the Bragg matching condition, the HeNe laser was mounted on a rotatable platform which pivoted about the center of the $BaTiO_3$ crystal. The crystal was held in place by a custom support on the end of a post holder which was bolted to the optical table. A bearing assembly around the post holder allowed the HeNe laser platform to freely rotate parallel to the optical table centered about

the crystal. The incident HeNe beam was monitored by detector "C" and the diffracted signal was detected by detector "E" in the figure. Both detectors had 514 nm interference filters to improve the signal to noise ratio.

The relative diffraction efficiency was determined by measuring the diffracted signal and dividing it by the sum of the transmitted zero order and the diffracted signal. In order to convert these numbers to absolute diffraction efficiency absorption would have to be taken into account. In addition, the diffraction efficiency at 0.5145 μm is higher than at 0.6328 μm by approximately a factor of 3/2 due to the $\lambda^{-2}$ dependency. During the experiment, we used an A/D board to acquire up to eight channels of data simultaneously. The board had 12 bit resolution and could operate at up to 250 KHz on a single channel.

The data we obtained from one typical experimental run is shown in Fig. 57. Figure 57(a) is the diffracted probe beam as a function of time. Figure 57(b) is the transmitted pump beam while Fig. 57c is the transmitted probe beam. Finally, Fig. 57(d) is the applied electrical waveform versus time. The run starts by allowing the two write beams to fall on an erased crystal at time t=1.1 seconds. A transfer of energy is seen to occur between the two writing beams. As the transmitted probe beam, Fig. 57(c), grows in amplitude, the pump beam is depleted. The signal proportional to the diffraction efficiency of the probe beam, Fig. 57(a), grows with time to a peak value. At a time 41.6 seconds into the run, a square electrical pulse of one second duration was applied with the waveform shown in Fig. 57(d).

At that point the photorefractive gratings are disrupted and the two wave mixing gain is impacted. Therefore, the transmitted probe and pump intensities approach the initial condition before the gratings were established. However, because the two writing beams continue to illuminate the crystal, they begin to rewrite gratings which again causes an energy transfer.

The diffracted HeNe signal monitoring the photorefractive hologram decreased in amplitude at the occurrence of the leading edge of the electrical pulse. Then, because the writing beams were still on, the diffracted signal grows slightly. At time 43.1 seconds into the run, the pump and probe beams are blocked and the diffracted signal remains essentially constant in time. The crystal was then illuminated with an incoherent light at a time equal to 49.6 seconds. The apparent slight rise in the amplitude of the diffracted signal is due to a small percentage of scattered erase light entering detector "E". This noise occurred even though we used a an aperture and interference filter in front of the detector to minimize the effect.

Using a generalized curve fitting routine, we determined the functional form of the diffracted signal in the presence of the erase beam. The diffracted signal decayed with a single exponential time constant to the offset determined by the noise floor produced by the erase light. The erase light was turned off at time 75.9 seconds. As can be seen, no residual fixed grating was present in the crystal which would have diffracted the HeNe monitor beam. After the grating was

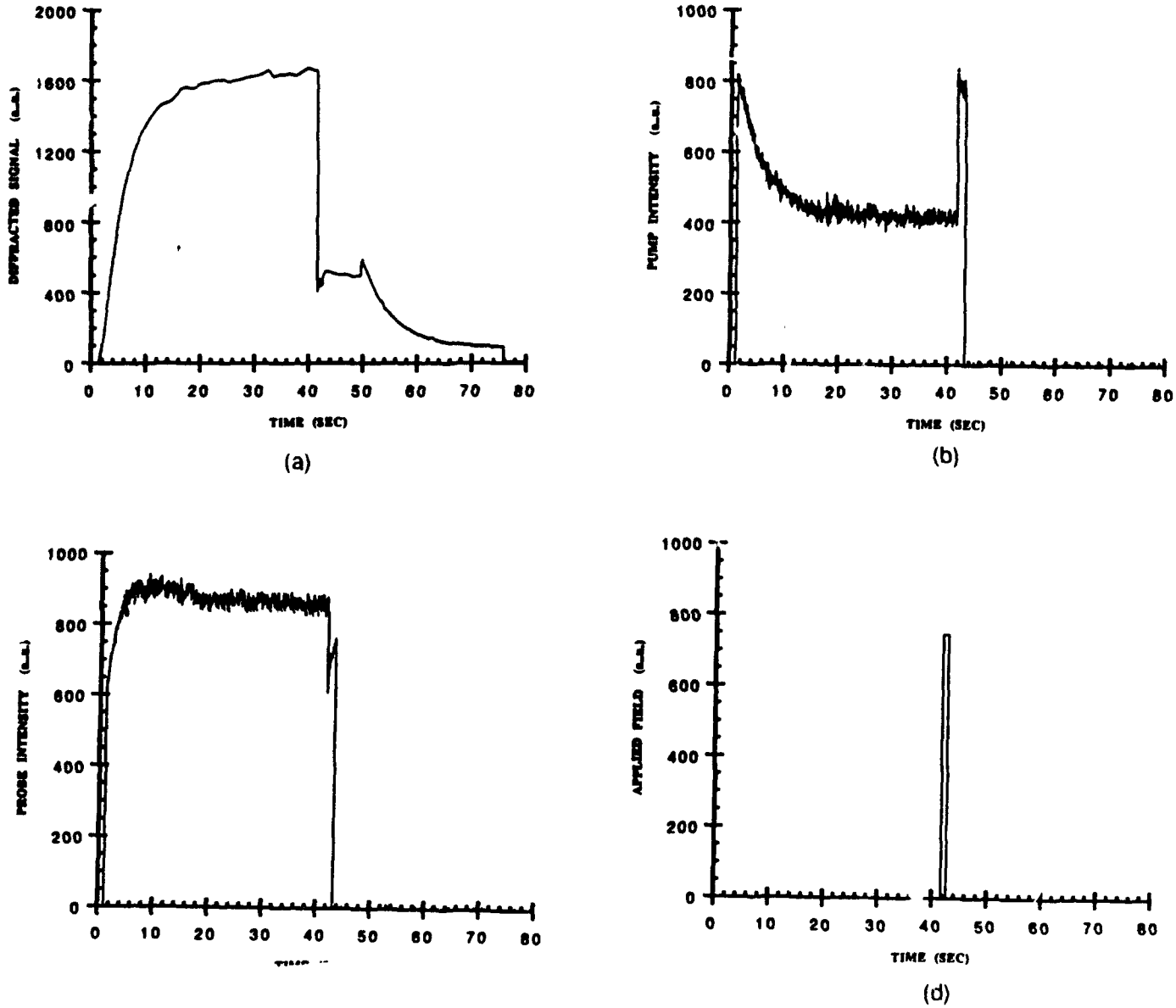



**Fig. 57. Typical experimental data from hologram fixing experiments. (a) Diffracted signal. (b) Pump beam intensity (c) Probe Beam intensity (d) Applied voltage.**

completely erased, the crystal was "reset" close to its initial state before trying another run. This was accomplished by re-polling, i.e., cycling through several hysteresis loops or applying half sine waves with sufficient amplitude to realign the domains.

To-date we have not been successful in electrically fixing gratings in $BaTiO_3$ crystals. However, on another contract we were successful in thermally fixing gratings in photorefractive $LiNbO_3$ crystals. We achieved fixed diffraction efficiencies of up to 52%. Thermal fixing may thus be a reasonable alternative to electrical fixing of $BaTiO_3$.

An alternative to fixing weights is to continuously refresh the weights after the neural network has learned the solution of a particular problem. This can be done by interleaving occasional learning or refresh cycles into the readout phase. Once the network has learned a solution, only occasional "reminders" are necessary to keep the network from "forgetting" the solution. This technique is analogous to refresh cycles in conventional DRAM.

# SECTION 9
# CONCLUSIONS

During the course of this three year DARPA contract we demonstrated the application of cascaded grating holography (CGH) in optical neural networks. By distributing each connection weight among a plethora of cascaded gratings which vary in position and orientation, CGH improves holographic optical neural networks in several ways. First, Bragg degeneracy is eliminated which permits the placement of pixels in arbitrary 2-D patterns in the input and output planes, leading to new flexibility in holographic recording. Second, CGH reduces distortions due to beam coupling, resulting in undistorted weight vectors and good quality reconstructed images. The flexibility of CGH allows the design of compact programmable holographic neurocomputers which are composed of a single photorefractive crystal, SLM, and detector. The utilization of the SLM is maximized because every pixel can be used without fear of crosstalk.

Under this contract we constructed an optical neurocomputer and showed its flexibility by demonstrating several neural networks without adjustment of the optical components. The networks were Perceptron, bidirectional associative memory, and backpropagation with a single hidden layer. We also demonstrated CGH in the infrared using compact laser diode sources and designed packaging concepts based on laser diodes.

In short, we have demonstrated the viability and potential of CGH-based optical neural networks. Future work should include refinement of current algorithms and implementation of new ones, improvement of storage capacity and processing speed, demonstration of compact packages, and application of this technology to a real-world problem.

# APPENDIX A

# OPTICAL BACKPROPAGATION PROGRAM SOURCE CODE

```
/* learn34n.c Optical Backpropagation, November, 1992.

  This C language program implements a three-layer, single hidden-layer,
  bipolar neuron backpropagation neural network in
  forward-scattering SPONN which maps random input patterns
  to random output patterns. In addition to the host IBM-compatible PC, it also
  controls the FG100 input image processor, the VS100 output image processor,
  and the LC cells which shutter the R and O beams. Rough output coordinates
  are read from align34n.dat, which is generated using a separate program, alignvs6.c.
  Final coordinates are stored in align34n.dat. Local search and sorting
  are used to find the brightest output pixels. Neuron outputs are represented
  using a balanced dual-rail method. L1 and L2 fields are spatially
  interleaved and up and down weights are exposed simultaneously
  to force equality between the two sets of weights. Neuron pixel size is adjustable.
  Trinary quantization of weight updates is an option. Thresholded output error
  measures correctness of output sign only. The FG100 input image processor must be
  initialized at 8 MHz to work properly.

  This program was written by Yuri Owechko, Hughes Aircraft Company, 3011 Malibu
  Canyon Rd., Malibu, CA 90265. E-mail: owechko@csfvax.dnet.hac.com
  310-317-5839.

  © Copyright 1992, Hughes Aircraft Co.
*/

#include <graph.h>
#include <ctype.h>
#include <stdio.h>
#include <conio.h>
#include <itex100.h>
#include <math.h>
#include <process.h>
#include <time.h>
#include <float.h>
#include <dos.h>
#include <search.h>
#include <stdlib.h>

/* First bit controls laser and detector LC cells,
  second bit controls object LC cell, third bit controls
  detector shutter */

#define WRITE 3
#define READ 4
```

```
#define STANDBY 0
#define SPONN_INPUT 0x340,0xd0000L,2,1
#define SPONN_OUTPUT 0x360,0xd0000L,2,1

int brand(),getbits();
void conf_samples(),readout_L1(),readout_L2();
void change_sign_L1(),change_sign_L2();
void exemplar_random_L0(),disp_cell(),transfer_L1();
void train_W1(),train_W2(),backward();
int round(),compare(),hsgn(),satur();
float error_learn(),error_read(),fsigmo(),ghump();
float compare_sign(),avg_rpixel();

FILE *fopen(),*stream;

int xout[32][64][16],yout[32][64][16],xt[100],yt[100];
int xin_obj[32],yin_obj[64],mxmax,mymax,mmax,seed,seed2;
int xin_ref[32],yin_ref[64],npixels,size;
int arraydim,xmax,xmax1,ymax,ymax1,ymax2,arraydim2,base1,base2;
int pxout[32][64],pyout[32][64];
long int on_time,off_time;
float sum1[32][20],sum2[32][20];
float yy1[16][20],yy2[16][20],delta1[16][20],delta2[16][20];
float toterr[500],proj_w[500][2],proj_r[500][2],toterr_read[500],fgamma;

struct dosdate_t date;
struct dostime_t time1;
struct dostime_t time2;

    main()
    {
    int n,m,mx,my,i,j,k,nepoch,nepoch2,ntotal;
    int err_value,nframes,nxavg2,nyavg2,pixval;
    int gain,offsetval,xbase,ybase,zoomval,v1[256];
    int i1,i2,nx,ny,nxavg,nyavg,value,shift,decquant,declut,declut2,decAlign;
    long int delay3,int_time,lval,klong;
    long int ltime,prevtime;
    float ftime,looptime,gain2,ontime,offtime,error,f1,f2;
    float inttime,inttime2,fval,eps1,eps2;
    char comment[80],comment2[80],comment3[80],stemp[200],blk,ch;

    base1=592; /* 592=250h PPI 1 base address */
    base2=596; /* 596=254h PPI 2 base address */
    blk=219;   /* Solid block ASCII character */

/*  Set mode of DIO48 I/O card for Mode 0, all outputs  */

    outp(base1+3,128);
    outp(base2+3,128);

    if ((stream=fopen("learn34n.inp","r"))==NULL)
    {printf("cannot open learn34n.inp\n");
    exit(0);}
    else
```

```
printf("learn34n.inp parameter file opened for reading\n");

fgets(comment2,80,stream);
fgets(comment,80,stream);
fgets(comment3,80,stream);
printf("Enter number of exemplars in x and y\n(mxmax<=16 and mymax<=15 for 64x64 display fields): ");
fgets(stemp,200,stream);
fscanf(stream,"%d %d\n",&mxmax,&mymax);
printf("%d %d\n",mxmax,mymax);
mmax=mxmax*mymax;
printf("No. of exemplars= %d\n",mmax);

printf("Enter zoom value for display field size (3->64x60, 2->128x120, 1->256x240, 0->512x480): ");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&zoomval);
printf("%d\n",zoomval);
arraydim=pow(2,9-zoomval);
arraydim2=arraydim/2;
printf("Display field dimension is %d\n",arraydim);

printf("Enter xmax and ymax (for 64x64 arrays, xmax=30 & ymax=57): ");
fgets(stemp,200,stream);
fscanf(stream,"%d %d\n",&xmax,&ymax);
printf("%d %d\n",xmax,ymax);
xmax1=xmax/2;
ymax1=ymax/3;
ymax2=2*ymax/3;

printf("Enter subneuron edge dimension: ");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&size);
printf("%d\n",size);

printf("Enter 1 to realign, 0 otherwise: ");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&decAlign);
printf("%d\n",decAlign);

printf("Enter no. of epochs up to 500: ");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&nepoch);
printf("%d\n",nepoch);
nepoch2=nepoch;

printf("Enter exemplar integration time in sec: ");
fgets(stemp,200,stream);
fscanf(stream,"%f\n",&inttime);
printf("%f\n",inttime);

printf("Enter Output Frame Grabber gain from 1 to 4: ");
fgets(stemp,200,stream);
fscanf(stream,"%f\n",&gain2);
printf("%f\n",gain2);
gain=255*gain2/4;
```

```
printf("Enter Output Frame Grabber offset from 0 to 255: ");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&offsetval);
printf("%d\n",offsetval);

printf("Output Frame Grabber input LUT: 0 for unmodified, 1 for sqrt \n");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&declut);
printf("%d\n",declut);

printf("Input Frame Grabber output LUT: 0 for unmodified, 1 for linearized output\n");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&declut2);
printf("%d\n",declut2);

printf("Enter fsigmo gamma value: ");
fgets(stemp,200,stream);
fscanf(stream,"%f\n",&fgamma);
printf("%f\n",fgamma);

printf("Enter 1 for trinary quantization of weight updates, 0 otherwise:\n");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&decquant);
printf("%d\n",decquant);

printf("Enter eps1 for y1 quantization and eps2 for delta1 and delta2 quantization:\n");
fgets(stemp,200,stream);
fscanf(stream,"%f %f\n",&eps1,&eps2);
printf("%f %f\n",eps1,eps2);

printf("Enter SLM turn-on and turn-off times in sec: ");
fgets(stemp,200,stream);
fscanf(stream,"%f %f\n",&ontime,&offtime);
printf("%f %f\n",ontime,offtime);

printf("Enter x and y dimensions of averaging regions (odd numbers<=9): ");
fgets(stemp,200,stream);
fscanf(stream,"%d %d\n",&nxavg,&nyavg);
printf("%d %d\n",nxavg,nyavg);
nxavg2=(nxavg-1)/2;
nyavg2=(nyavg-1)/2;

printf("Enter no. of samples per subneuron (<=16): ");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&npixels);
printf("%d\n",npixels);

printf("Enter no. of video frames to average (<=100): ");
fgets(stemp,200,stream);
fscanf(stream,"%d\n",&nframes);
printf("%d\n",nframes);

printf("Enter seed integer for random number generator: ");
```

```
    fgets(stemp,200,stream);
    fscanf(stream,"%d\n",&seed);
    printf("%d\n",seed);
    seed2=seed;

    fclose(stream);

/* Turn off Input Frame Grabber */

    sethdw(SPONN_INPUT);
    setdim(1024,1024,12);
    fgoff();

/* Set up Output Frame Grabber */

    sethdw(SPONN_OUTPUT);
    initialize();
    extsync();
    setcamera(0);
    static_luts();
    setgain(gain);
    setoffset(offsetval);

/*   Modify INPUT LUT of Output Frame Grabber */

    if(declut){
     for(j=0;j<=255;j++) {
      v1[j]=round(sqrt(255.)*sqrt(j));  }
     walut(INPUT,0,0,256,v1);
     setlut(INPUT,0); }

/* Set output LUTs of Output Frame Grabber for display. */

    if(declut){
     for(j=0;j<=255;j++){
      v1[j]=round(pow(j,2)/255.); }
     walut(RED,0,0,256,v1);
     v1[0]=255;
     v1[255]=0;
     walut(BLUE,0,0,256,v1);
     v1[0]=0;
     walut(GREEN,0,0,256,v1);
     setlut(GREEN,0); }
    else {
     for(j=0;j<=255;j++){
      v1[j]=j; }
     walut(RED,0,0,256,v1);
     v1[0]=255;
     v1[255]=0;
     walut(BLUE,0,0,256,v1);
     v1[0]=0;
     walut(GREEN,0,0,256,v1);
     setlut(GREEN,0); }
```

```
    grab(0);

/* Set up Input Frame Grabber */

    switch(declut2)
     {
      case 0:
        for(j=0;j<=255;j++){
         v1[j]=j;}
       break;
      case 1:
        if((stream=fopen("linear.lut","r"))==NULL)
           {printf("cannot open linear.lut\n");
           exit(0);}
       else
           printf("\nlinear.lut file opened for reading\n");
        for(j=0;j<=255;j++){
         fscanf(stream,"%d\n",&v1[j]);}
        fclose(stream);
       break;
     }

     fgoff();
     sethdw(SPONN_INPUT);
     fgon();

     walut(GREEN,0,0,256,v1);    /* Adjust Input Frame Grabber output LUT */
     walut(RED,0,0,256,v1);
     v1[254]=0;
     v1[255]=0;
     walut(BLUE,0,0,256,v1);

     aclear(0,0,1023,1023,0);

     zoom(zoomval);
     disp_cell(0,0);

/* Calibrate time base */

     time(&ltime);
     prevtime=ltime;
     for(klong=0;klong<2000000;klong++);
     time(&ltime);
     ltime=ltime-prevtime;
     looptime=ltime/2000000.0;
     int_time=intime/looptime;
     on_time=ontime/looptime;
     off_time=offtime/looptime;

/* Generate SPONN input coordinate vectors */

     for(i=0;i<xmax;i++){
     xin_ref[i]=arraydim2+i*size;}
     for(i=0;i<ymax;i++){
```

```
    yin_ref[i]=i*size;}

    for(i=0;i<xmax;i++){
    xin_obj[i]=i*size;}
    for(i=0;i<ymax;i++){
    yin_obj[i]=i*size;}

/* Store zero pattern in (0,0); ref-on in (1,0); object subsets A and B
  in (3,0) and (4,0), respectively. */

    xbase=arraydim;
    ybase=0;

    for(j=0;j<ymax;j++){
     for(i=0;i<xmax;i++){
      block(xbase+xin_ref[i],ybase+yin_ref[j],size,size,255);
      }}

    xbase=3*arraydim;
    ybase=0;

    for(j=0;j<ymax;j++){
     shift=ceil(fmod(j,2.0));
     for(i=0;i<xmax;i+=2){
      block(xbase+xin_obj[i+shift],ybase+yin_obj[j],size,size,255);
      }}

    xbase=4*arraydim;
    ybase=0;

    for(j=0;j<ymax;j++){
     shift=1-ceil(fmod(j,2.0));
     for(i=0;i<xmax;i+=2){
      block(xbase+xin_obj[i+shift],ybase+yin_obj[j],size,size,255);
      }}


/* Calculate output coordinates and fine-tune by searching
  local neighborhoods. */

if(decAlign)
 {
   if ((stream=fopen("align34n.dat","r"))==NULL)
   {printf("\ncannot open align34n.dat\n");
   exit(0);}
   else
   printf("\nalign34n.dat coordinate file opened for reading\n");

   for(j=0;j<ymax;j++){
    for(i=0;i<xmax;i++){
     fscanf(stream,"%d %d ",&xout[i][j][0],&yout[i][j][0]);}
    fscanf(stream,"\n");}

   fclose(stream);
```

```
    disp_cell(3,0); /* Turn on all object "A" pixels */

    fgoff();
    sethdw(SPONN_OUTPUT);
    fgon();

printf("\nAdjust brightness then hit any key to save output coordinates\n");

    outp(base1,READ);
    nxavg2=(nxavg-1)/2;
    nyavg2=(nyavg-1)/2;

    for(my=0;my<ymax;my++){
     for(mx=0;mx<xmax;mx++){
      pxout[mx][my]=xout[mx][my][0];
      pyout[mx][my]=yout[mx][my][0];}}

    for(;;){
      for(my=0;my<ymax;my++){
       shift=ceil(fmod(my,2.0));
        for(mx=0;mx<xmax;mx+=2){
         rectangle(pxout[shift+mx][my]-nxavg2,pyout[shift+mx][my]-nyavg2,nxavg,nyavg,255);
         rectangle(pxout[shift+mx][my]-nxavg2,pyout[shift+mx][my]-nyavg2,nxavg,nyavg,255);}}
      snap(WAIT);
      if(kbhit())break;
        }

    ch=getch(); /* Empties key buffer */

    printf("\nConfiguring output pixel coordinates, please wait...\n");

    fgoff();
    sethdw(SPONN_INPUT);
    fgon();
    disp_cell(0,0);
    fgoff();
    sethdw(SPONN_OUTPUT);
    fgon();

    for(my=0;my<ymax;my++){
     shift=ceil(fmod(my,2.0));
     for(mx=0;mx<xmax;mx+=2){
      conf_samples(shift+mx,my,nxavg2,nyavg2);}}

    fgoff();
    sethdw(SPONN_INPUT); /* Turn on all object "B" pixels */
    fgon();
    disp_cell(4,0);
    lval=3*on_time;
    for(klong=0;klong<lval;klong++);

    fgoff();
    sethdw(SPONN_OUTPUT);
```

```
    fgon();

    snap(WAIT);

    fgoff();
    sethdw(SPONN_INPUT);
    fgon();
    disp_cell(0,0);
    fgoff();
    sethdw(SPONN_OUTPUT);
    fgon();

     for(my=0;my<ymax;my++){
      shift=1-ceil(fmod(my,2.0));
      for(mx=0;mx<xmax;mx+=2){
       conf_samples(shift+mx,my,nxavg2,nyavg2);}}

     for(my=0;my<ymax;my++){
      for(mx=0;mx<xmax;mx++){
       for(i=0;i<npixels;i++){
        block(xout[mx][my][i],yout[mx][my][i],size,size,255);}}}

    printf("\nOutput pixels marked in red on monitor\n");

    lval=3.0/looptime;
    for(klong=0;klong<lval;klong++);

 /* Write output coordinates to file align34n.dat */

    if ((stream=fopen("align34n.dat","w"))==NULL)
    {printf("\ncannot open align34n.dat\n");
    exit(0);}
    else
    printf("\nalign34n.dat coordinate file opened for writing\n");

     for(my=0;my<ymax;my++){
      for(mx=0;mx<xmax;mx++){
       for(i=0;i<npixels;i++){
        fprintf(stream,"%d %d ",xout[mx][my][i],yout[mx][my][i]);}
       fprintf(stream,"\n");}}

    fclose(stream);
    grab(0);
  }
else
 {
    if ((stream=fopen("align34n.dat","r"))==NULL)
    {printf("\ncannot open align34n.dat\n");
    exit(0);}
    else
    printf("\nalign34n.dat coordinate file opened for reading\n");

     for(my=0;my<ymax;my++){
      for(mx=0;mx<xmax;mx++){
```

```
        for(i=0;i<npixels;i++){
          fscanf(stream,"%d %d ",&xout[mx][my][i],&yout[mx][my][i]);}
        fscanf(stream,"\n");}}

    fclose(stream);
    fgoff();
    sethdw(SPONN_OUTPUT);
    fgon();
    grab(0);
}

/* Store L0 exemplars in (0,1) to (mxmax-1,mymax) */

    fgoff();
    sethdw(SPONN_INPUT);
    fgon();

    for(mx=0;mx<mxmax;mx++){
      for(my=0;my<mymax;my++){
        exemplar_random_L0(mx,my);}}

/* Initialize weights using sum of random outer-products */

    printf("\nInitializing weights with sum of random outer-products...\n\nHit any key to begin learning\n");
    lval=4*int_time;

    outp(base1,WRITE);
    for(;;){

/* (L0)(L1) Weight initialization vectors in (5,0). */

    ybase=0;
    n=5;
    xbase=n*arraydim;
    srand(seed+16000+n);

      for(my=0;my<ymax1;my++){
        for(mx=0;mx<xmax;mx+=2){
          if(brand()==-1){
            block(xin_ref[mx]+xbase,yin_ref[my]+ybase,size,size,255);
            block(xin_ref[mx+1]+xbase,yin_ref[my]+ybase,size,size,0);}
          else {
            block(xin_ref[mx]+xbase,yin_ref[my]+ybase,size,size,0);
            block(xin_ref[mx+1]+xbase,yin_ref[my]+ybase,size,size,255);}
                    }}

      for(my=0;my<ymax1;my++){
        for(mx=0;mx<xmax;mx+=2){
          if(brand()==-1){
            block(xin_obj[mx]+xbase,yin_obj[my]+ybase,size,size,255);
            block(xin_obj[mx+1]+xbase,yin_obj[my]+ybase,size,size,0);}
          else {
            block(xin_obj[mx]+xbase,yin_obj[my]+ybase,size,size,0);
            block(xin_obj[mx+1]+xbase,yin_obj[my]+ybase,size,size,255);}
```

```
                }}

/* (L1)(L2) vectors for weight initialization in (7,0). */

    ybase=0;
    n=7;
    xbase=n*arraydim;
    srand(seed+20000+n);

      for(my=0;my<ymax1;my++){
       for(mx=0;mx<xmax;mx+=2){
        if(brand()==-1){
         block(xin_ref[mx]+xbase,yin_ref[2*my+ymax1]+ybase,size,size,255);
         block(xin_ref[mx+1]+xbase,yin_ref[2*my+ymax1]+ybase,size,size,0);
         block(xin_obj[mx]+xbase,yin_obj[2*my+ymax1+1]+ybase,size,size,255);
         block(xin_obj[mx+1]+xbase,yin_obj[2*my+ymax1+1]+ybase,size,size,0);
         }
        else {
         block(xin_ref[mx]+xbase,yin_ref[2*my+ymax1]+ybase,size,size,0);
         block(xin_ref[mx+1]+xbase,yin_ref[2*my+ymax1]+ybase,size,size,255);
         block(xin_obj[mx]+xbase,yin_obj[2*my+ymax1+1]+ybase,size,size,0);
         block(xin_obj[mx+1]+xbase,yin_obj[2*my+ymax1+1]+ybase,size,size,255);
         }
                }}

      for(my=0;my<ymax1;my++){
       for(mx=0;mx<xmax;mx+=2){
        if(brand()==-1){
         block(xin_ref[mx]+xbase,yin_ref[2*my+ymax1+1]+ybase,size,size,255);
         block(xin_ref[mx+1]+xbase,yin_ref[2*my+ymax1+1]+ybase,size,size,0);
         block(xin_obj[mx]+xbase,yin_obj[2*my+ymax1]+ybase,size,size,255);
         block(xin_obj[mx+1]+xbase,yin_obj[2*my+ymax1]+ybase,size,size,0);
         }
        else {
         block(xin_ref[mx]+xbase,yin_ref[2*my+ymax1+1]+ybase,size,size,0);
         block(xin_ref[mx+1]+xbase,yin_ref[2*my+ymax1+1]+ybase,size,size,255);
         block(xin_obj[mx]+xbase,yin_obj[2*my+ymax1]+ybase,size,size,0);
         block(xin_obj[mx+1]+xbase,yin_obj[2*my+ymax1]+ybase,size,size,255);
         }
                }}


    disp_cell(5,0);
    for(klong=0;klong<=lval;klong++);
    disp_cell(7,0);
    for(klong=0;klong<=int_time;klong++);

    seed++;
    if(kbhit())break;}

  disp_cell(0,0);
  outp(base1,READ);
```

```
    ch=getch();

/*****************************************************************/
/*          Learning Loop                                        */
/*****************************************************************/

      _dos_gettime(&time1);
      f1=2.0/(mmax*xmax*ymax1);

      for(n=0;n<nepoch;n++){
       error=0;
         for(my=0;my<mymax;my++){
          for(mx=0;mx<mxmax;mx++){
           disp_cell(mx,my+1); /* Input L0 exemplar (mx,my) */
           readout_L1(nframes);
           transfer_L1();
           readout_L2(nframes);
           err_value=error_learn(mx,my,n);
           error=error+err_value;
           backward(nframes);
           train_W2(int_time,decquant,eps1,eps2);
           train_W1(int_time,decquant,eps2,mx,my);}}

         toterr[n]=f1*error;

        printf("\nLearning Epoch= %3d Error= %5.3f ",n,toterr[n]);
        value=toterr[n]*40;
         for(i=0;i<value;i++){
          printf("%c",blk);}

       if(kbhit()){
       ch=getch();
       nepoch=n;
       break; }
      }

     _dos_gettime(&time2);

/*****************************************************************/
/*          Readout Loop                                         */
/*****************************************************************/

     printf("\n\nReadout of network:\n\n");

     for(n=0;n<nepoch2;n++){
      error=0;
        for(my=0;my<mymax;my++){
         for(mx=0;mx<mxmax;mx++){
          disp_cell(mx,my+1);
          readout_L1(nframes);
          transfer_L1();
          readout_L2(nframes);
          err_value=error_read(mx,my,n);
          error=error+err_value;
```

```
    }}

 toterr_read[n]=f1*error;

 printf("\nReadout Epoch=  %3d Error= %5.3f ",n,toterr_read[n]);
 value=toterr_read[n]*40;
 for(i=0;i<value;i++){
  printf("%c",blk);}

 if(kbhit()){
  nepoch2=n;
  break; }
 }

outp(base1,STANDBY);
disp_cell(1,0);
fgoff();
sethdw(SPONN_OUTPUT);
fgon();
grab(0);

if((stream=fopen("learn34n.csv","w"))==NULL)
{printf("cannot open learn34n.csv\n");
exit(0);}
else
printf("\nlearn34n.csv file opened for writing\n");

fprintf(stream,"learn34n.c Two-Layer Optical Backprop. with Symmetric Dual-Rail Levels (Random Mapping Problem)\n");
_dos_getdate(&date);
fprintf(stream,"Date: %d-%d-%d ",date.month,date.day,date.year);
fprintf(stream,"Crystal: %s",comment2);
fprintf(stream,"%s",comment);
fprintf(stream,"%s",comment3);
fprintf(stream,"Enter number of exemplars in x and y:\n");
fprintf(stream,"%d, %d\n",mxmax,mymax);
fprintf(stream,"zoomval= %d\n",zoomval);
fprintf(stream,"Display field dimension is \n%d\n",arraydim);
fprintf(stream,"xmax and ymax= \n%d, %d\n",xmax,ymax);
fval=xmax*ymax1+2*xmax1*ymax1;
fprintf(stream,"No. of neurons= \n%e\n",fval);
fval=xmax1*ymax1;
fval=(xmax*ymax1)*(fval)+(fval)*(fval);
fprintf(stream,"No. of weights= \n%e\n",fval);
fprintf(stream,"Enter subneuron edge dimension:\n%d\n",size);
fprintf(stream,"No. of epochs: ");
fprintf(stream,"%d\n",nepoch);
fprintf(stream,"Exemplar integration time in sec: ");
fprintf(stream,"%f\n",inttime);
fprintf(stream,"Output Frame Grabber gain: ");
fprintf(stream,"%f\n",gain2);
fprintf(stream,"Output Frame Grabber offset: ");
fprintf(stream,"%d\n",offsetval);
fprintf(stream,"Output Frame Grabber input LUT: 1 for square root; 0 for linear\n");
fprintf(stream,"%d\n",declut);
```

```
fprintf(stream,"Input Frame Grabber output LUT: 0 for unmodified; 1 for linearized SLM intensity\n");
fprintf(stream,"%d\n",declut2);
fprintf(stream,"Enter fsigmo gamma value\n");
fprintf(stream,"%f\n",fgamma);
fprintf(stream,"Enter 1 for trinary quantization of weight updates; 0 otherwise\n");
fprintf(stream,"%d\n",decquant);
fprintf(stream,"Enter eps1 for y1 quantization and eps2 for delta1 and delta2 quantization\n");
fprintf(stream,"%f, %f\n",eps1,eps2);
fprintf(stream,"LCLV turn-on and turn-off delays in sec= \n%f, %f\n",ontime,offtime);
fprintf(stream,"Enter x and y dimensions of averaging regions (odd numbers)\n");
fprintf(stream,"%d, %d\n",nxavg,nyavg);
fprintf(stream,"Enter no. of samples per subneuron\n%d\n",npixels);
fprintf(stream,"Enter no. of video frames to average (<=100)\n");
fprintf(stream,"%d\n",nframes);
fprintf(stream,"Enter starting seed for random number generator\n");
fprintf(stream,"%d\n",seed2);
fprintf(stream,"Learning start time: %d:%d:%d  Stop time: %d:%d:%d\n",time1.hour,
    time1.minute,time1.second,time2.hour,time2.minute,time2.second);

fprintf(stream,"\n\nLearning Error and 2 Output Projections vs. Epoch \n\n");
for(n=0;n<nepoch;n++){
 fprintf(stream,"%d, %f, ,%d, %f, %f\n",n,toterr[n],n,proj_w[n][0],proj_w[n][1]);}

fprintf(stream,"\n\nReading Error and 2 Output Projections vs. Epoch\n\n");
for(n=0;n<nepoch2;n++){
 fprintf(stream,"%d, %f, ,%d, %f, %f\n",n,toterr_read[n],n,proj_r[n][0],proj_r[n][1]);}

fclose(stream);


}

/* This function rounds a floating point number to the nearest integer */

int round(fval)
 float fval;
 {
 int val;
 float remain;

 remain=fval-floor(fval);
 if(remain>=0.5)
 val=ceil(fval);
 else
 val=floor(fval);

 return(val);
 }

/* This function finds the sample points for neuron (mx,my) */

 void conf_samples(mx,my,nxavg2,nyavg2)
 int mx,my,nxavg2,nyavg2;
 {
```

```
        int count,xts,yts,i,j,num[100];

        xts=pxout[mx][my];
        yts=pyout[mx][my];

/* Find brightest pixels by sorting */

        for(i=0;i<100;i++){num[i]=0;}

        count=0;

        for(j=-nyavg2+1;j<=nyavg2;j++){
          for(i=-nxavg2+1;i<=nxavg2;i++){
            xt[count]=xts+i;
            yt[count]=yts+j;
            num[count]=count;
            count++;}}

        qsort(num,100,sizeof(int),compare);

        for(i=0;i<npixels;i++){
          xout[mx][my][i]=xt[num[i]];
          yout[mx][my][i]=yt[num[i]];}

      }

/* Descending order comparison function for qsort(). It compares
   sampled pixel values. */

        int compare(i,j)
        int *i,*j;
        {
        int val,a,b;

        a=rpixel(xt[*j],yt[*j]);
        b=rpixel(xt[*i],yt[*i]);
        val=a-b;

        return(val);

        }

/* This function calculates L1 values */

      void readout_L1(nframes)
      int nframes;
      {
      int i1,i2,mx2,my2,val;
      long int klong;
      float fval;

      fgoff();
      sethdw(SPONN_OUTPUT);
      fgon();
```

```
 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax;mx2++){
   sum1[mx2][my2]=0;}}

 outp(base1,READ);
 for(klong=0;klong<on_time;klong++);
 for(i2=0;i2<nframes;i2++){
  snap(WAIT);
  for(my2=0;my2<ymax1;my2++){
   for(mx2=0;mx2<xmax;mx2++){
    sum1[mx2][my2]=sum1[mx2][my2]+avg_rpixel(mx2,my2);}}
            }
 outp(base1,READ);
 fval=1.0/nframes;
 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax;mx2++){
   sum1[mx2][my2]=fval*sum1[mx2][my2];}}

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax1;mx2++){
   yy1[mx2][my2]=fsigmo(sum1[2*mx2][my2]-sum1[2*mx2+1][my2]);}}
}

/* Transfers L1 output to L1 input */

 void transfer_L1()
 {
 int mx2,my2,pixval;

 fgoff();
 sethdw(SPONN_INPUT);
 fgon();

 disp_cell(0,0);

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax1;mx2++){
   pixval=127*yy1[mx2][my2];
   block(xin_ref[2*mx2],yin_ref[2*my2+ymax1],size,size,127+pixval);
   block(xin_ref[2*mx2+1],yin_ref[2*my2+ymax1],size,size,127-pixval);
            }}
 }

/* This function calculates L2 values */

 void readout_L2(nframes)
 int nframes;
 {
 int i1,i2,mx2,my2,val;
 long int klong;
 float fval;

 fgoff();
```

```
 sethdw(SPONN_OUTPUT);
 fgon();

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax;mx2++){
   sum1[mx2][my2]=0;
   sum2[mx2][my2]=0;}}

 outp(base1,READ);
 for(klong=0;klong<on_time;klong++);

 for(i2=0;i2<nframes;i2++){
  snap(WAIT);
  for(my2=0;my2<ymax1;my2++){
   for(mx2=0;mx2<xmax;mx2++){
    sum1[mx2][my2]=sum1[mx2][my2]+avg_rpixel(mx2,ymax1+2*my2);}}
            }
 change_sign_L1();
 for(klong=0;klong<on_time;klong++);

 fgoff();
 sethdw(SPONN_OUTPUT);
 fgon();

 for(i2=0;i2<nframes;i2++){
  snap(WAIT);
  for(my2=0;my2<ymax1;my2++){
   for(mx2=0;mx2<xmax;mx2++){
    sum2[mx2][my2]=sum2[mx2][my2]+avg_rpixel(mx2,ymax1+2*my2);}}
  }

 outp(base1,READ);

 fval=0.5/nframes;
 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax;mx2++){
   sum1[mx2][my2]=fval*sum1[mx2][my2];
   sum2[mx2][my2]=fval*sum2[mx2][my2];}}

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax1;mx2++){
   yy2[mx2][my2]=fsigmo(sum1[2*mx2][my2]-sum1[2*mx2+1][my2]-sum2[2*mx2][my2]+sum2[2*mx2+1][my2]);
   }}
}

/* This routine spatially averages bright pixels */

 float avg_rpixel(mx,my)
  int mx,my;
  {
  int j,i,val;
  float fval;

  val=0;
```

```
    for(j=0;j<npixels;j++){
     val=val+rpixel(xout[mx][my][j],yout[mx][my][j]);}
    fval=(float)val/npixels;
    return(fval);
    }


/* Calculates L2 output error for exemplar (mx,my)
   in learning phase */

    float error_learn(mx,my,n)
    int mx,my,n;
    {
    int mx2,my2,iex,c;
    float e2,error,fval;

    error=0;
    c=my*mxmax+mx;
    srand(seed+c);
    fval=0;
    for(my2=0;my2<ymax1;my2++){
     for(mx2=0;mx2<xmax1;mx2++){
      iex=brand();
      e2=0.5*(iex-yy2[mx2][my2]);
      error=error+compare_sign(iex,yy2[mx2][my2]);
      delta2[mx2][my2]=e2*ghump(yy2[mx2][my2]);
      fval=fval+yy2[mx2][my2];}}

    fval=fval/(xmax1*ymax1);
    printf("\nExemplar (%d,%d) error= %f projection= %f",mx,my,error,fval);

    if(c<2)
     proj_w[n][c]=fval;

    return(error);

    }

/* Calculates L2 output error for exemplar (mx,my) in reading phase */

    float error_read(mx,my,n)
    int mx,my,n;
    {
    int mx2,my2,iex,c;
    float e2,error,fval;

    error=0;
    c=my*mxmax+mx;
    srand(seed+c);

    fval=0;
    for(my2=0;my2<ymax1;my2++){
     for(mx2=0;mx2<xmax1;mx2++){
      iex=brand();
```

```
        e2=iex-yy2[mx2][my2];
        error=error+compare_sign(iex,yy2[mx2][my2]);
        fval=fval+yy2[mx2][my2];}}

      fval=fval/(xmax1*ymax1);
      printf("\nExemplar (%d,%d) error= %f projection= %f",mx,my,error,fval);

      if(c<2)
       proj_r[n][c]=fval;

      return(error);

      }

/* Backpropagate the error signal. */

      void backward(nframes)
      int nframes;
      {
      long int klong;
      int mx2,my2,i2,pixval;
      float e1,fval;

      fgoff();
      sethdw(SPONN_INPUT);
      fgon();

      /* Erase L1 input */

      block(arraydim2,ymax1*size,xmax*size,ymax2*size,0);

      /* Write delta2 to L2 reference */

      for(my2=0;my2<ymax1;my2++){
       for(mx2=0;mx2<xmax1;mx2++){
        pixval=127*delta2[mx2][my2];
        block(xin_ref[2*mx2],yin_ref[2*my2+ymax1+1],size,size,127+pixval);
        block(xin_ref[2*mx2+1],yin_ref[2*my2+ymax1+1],size,size,127-pixval);
                    }}

      /* Read L1 output and calculate delta1 */

      fgoff();
      sethdw(SPONN_OUTPUT);
      fgon();

      for(my2=0;my2<ymax1;my2++){
       for(mx2=0;mx2<xmax;mx2++){
        sum1[mx2][my2]=0;
        sum2[mx2][my2]=0;}}

      outp(base1,READ);
      for(klong=0;klong<on_time;klong++);
```

```
 for(i2=0;i2<nframes;i2++){
  snap(WAIT);
  for(my2=0;my2<ymax1;my2++){
   for(mx2=0;mx2<xmax;mx2++){
    sum1[mx2][my2]=sum1[mx2][my2]+avg_rpixel(mx2,ymax1+1+2*my2);}}
                }

 change_sign_L2();
 for(klong=0;klong<on_time;klong++);

 fgoff();
 sethdw(SPONN_OUTPUT);
 fgon();

 for(i2=0;i2<nframes;i2++){
  snap(WAIT);
  for(my2=0;my2<ymax1;my2++){
   for(mx2=0;mx2<xmax;mx2++){
    sum2[mx2][my2]=sum2[mx2][my2]+avg_rpixel(mx2,ymax1+1+2*my2);}}
  }

 outp(base1,READ);
 fval=0.5/nframes;

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax;mx2++){
   sum1[mx2][my2]=fval*sum1[mx2][my2];
   sum2[mx2][my2]=fval*sum2[mx2][my2];}}

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax1;mx2++){
   e1=0.00392*(sum1[2*mx2][my2]-sum1[2*mx2+1][my2]-sum2[2*mx2][my2]+sum2[2*mx2+1][my2]);
   delta1[mx2][my2]=e1*ghump(yy1[mx2][my2]);}}

 }

/* This function updates the weights between L2 and L1 */

 void train_W2(int_time,decquant,eps1,eps2)
 int decquant;
 float eps1,eps2;
 long int int_time;
 {
 int mx2,my2,pixval;
 long int klong;

 fgoff();
 sethdw(SPONN_INPUT);
 fgon();

 /* Write y1 to L1 object */

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax1;mx2++){
```

```
    if(decquant)
      pixval=127*hsgn(yy1[mx2][my2],eps1);
    else
      pixval=127*yy1[mx2][my2];
    block(xin_obj[2*mx2],yin_obj[2*my2+ymax1+1],size,size,127+pixval);
    block(xin_obj[2*mx2+1],yin_obj[2*my2+ymax1+1],size,size,127-pixval);
                }}

/* Write delta2 to L2 object */

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax1;mx2++){
    if(decquant)
      pixval=127*hsgn(delta2[mx2][my2],eps2);
    else
      pixval=127*delta2[mx2][my2];
    block(xin_obj[2*mx2],yin_obj[2*my2+ymax1],size,size,127+pixval);
    block(xin_obj[2*mx2+1],yin_obj[2*my2+ymax1],size,size,127-pixval);
                }}

/* Write y1 to L1 reference */

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax1;mx2++){
    if(decquant)
      pixval=127*hsgn(yy1[mx2][my2],eps1);
    else
      pixval=127*yy1[mx2][my2];
    block(xin_ref[2*mx2],yin_ref[2*my2+ymax1],size,size,127+pixval);
    block(xin_ref[2*mx2+1],yin_ref[2*my2+ymax1],size,size,127-pixval);
                }}

/* Write delta2 to L2 reference */

 for(my2=0;my2<ymax1;my2++){
  for(mx2=0;mx2<xmax1;mx2++){
    if(decquant)
      pixval=127*hsgn(delta2[mx2][my2],eps2);
    else
      pixval=127*delta2[mx2][my2];
    block(xin_ref[2*mx2],yin_ref[2*my2+ymax1+1],size,size,127+pixval);
    block(xin_ref[2*mx2+1],yin_ref[2*my2+ymax1+1],size,size,127-pixval);
                }}

/* Adjust L1-L2 and L2-L1 weights */

outp(base1,WRITE);
for(klong=0;klong<=int_time;klong++);
outp(base1,READ);

block(0,ymax1*size,arraydim,ymax2*size,0);

  }
```

```
/* This function updates the weights between L0 and L1 */

    void train_W1(int_time,decquant,eps2,mx,my)
    long int int_time;
    int mx,my,decquant;
    float eps2;
    {
    int mx2,my2,xbase,ybase,pixval;
    long int klong,lval;

    fgoff();
    sethdw(SPONN_INPUT);
    fgon();

    /* Display L0 exemplar */

    disp_cell(mx,my+1);

    /* Write delta1 to L1 object */

    xbase=mx*arraydim;
    ybase=(my+1)*arraydim;

    for(my2=0;my2<ymax1;my2++){
     for(mx2=0;mx2<xmax1;mx2++){
       if(decquant)
         pixval=127*hsgn(delta1[mx2][my2],eps2);
       else
         pixval=127*delta1[mx2][my2];
       block(xbase+xin_obj[2*mx2],ybase+yin_obj[my2],size,size,127+pixval);
       block(xbase+xin_obj[2*mx2+1],ybase+yin_obj[my2],size,size,127-pixval);
                    }}

    /* Adjust L0-L1 weights */

    lval=2*int_time;

    outp(base1,WRITE);
    for(klong=0;klong<=lval;klong++),
    outp(base1,READ);

    block(xbase,ybase,arraydim2,ymax1*size,0),

      }

/* Function to change sign of L1 ref. during readout */

    void change_sign_L1()
    {
    int mx2,my2,pixval;

    fgoff();
    sethdw(SPONN_INPUT);
    fgon();
```

```
    for(my2=0;my2<ymax1;my2++){
     for(mx2=0;mx2<xmax1;mx2++){
      pixval=-127*yy1[mx2][my2];
      block(xin_ref[2*mx2],yin_ref[2*my2+ymax1],size,size,127+pixval);
      block(xin_ref[2*mx2+1],yin_ref[2*my2+ymax1],size,size,127-pixval);
                    }}
    }

/* Function to change sign of L2 ref. during readout */

    void change_sign_L2()
    {
    int mx2,my2,pixval;

    fgoff();
    sethdw(SPONN_INPUT);
    fgon();

    for(my2=0;my2<ymax1;my2++){
     for(mx2=0;mx2<xmax1;mx2++){
      pixval=-127*delta2[mx2][my2];
      block(xin_ref[2*mx2],yin_ref[2*my2+ymax1+1],size,size,127+pixval);
      block(xin_ref[2*mx2+1],yin_ref[2*my2+ymax1+1],size,size,127-pixval);
                    }}
    }

/*  Neuron sigmoidal activation function */

    float fsigmo(fsum)
    float fsum;
    {
    float fval;
    fval=2.0/(1+exp(-fgamma*fsum))-1.0;
    return(fval);
    }


/* Backpropagation function */

    float ghump(fsum)
    float fsum;
    {
    float fval;
    fval=(1.0-fsum)*(1.0+fsum);
    return(fval);
    }

/* Saturation function */

    int satur(fsum)
    float fsum;
    {
    int ival;
```

```
    if(fsum>=255)
      ival=255;
    else
      ival=fsum;

    return(ival);
    }

/* Trinary quantization function */

    int hsgn(net,eps)
    float net,eps;
    {
    int ival;

    if(net<-eps)
      ival=-1;
    else if(net>eps)
      ival=1;
    else
      ival=0;

    return(ival);

    }

/*  This subroutine generates a (1,0) L0 random exemplar and stores
   it in a subregion of frame memory.  */

    void exemplar_random_L0(m,n)
    int m,n;
    {
    int mx2,my2,xbase,ybase,pixval;

    fgoff();
    sethdw(SPONN_INPUT);
    fgon();

    xbase=m*arraydim;
    ybase=(n+1)*arraydim;

    srand(seed+n*mxmax+m+mmax);

    for(my2=0;my2<ymax1;my2++){
      for(mx2=0;mx2<xmax;mx2++){
        if(brand()==-1){
          block(xin_ref[mx2]+xbase,yin_ref[my2]+ybase,size,size,255);}
        else {
          block(xin_ref[mx2]+xbase,yin_ref[my2]+ybase,size,size,0);}
                    }}

    }
```

```
/* Returns 1 or -1 randomly */

    int brand()
    {
    return(2*getbits(rand(),0,1)-1);
    }

/* This function gets n bits to the right of and including position p
   in integer x. */

    getbits(x,p,n)
    unsigned x,p,n;
    {
     return((x>>(p+1-n))&~(~0<<n));
    }

/* This subroutine displays cell (m,n) */

    void disp_cell(m,n)
    int m,n;
    {
    int xbase,ybase;

    fgoff();
    sethdw(SPONN_INPUT);
    fgon();

    xbase=m*arraydim;
    ybase=n*arraydim;

    pan(xbase);
    scroll(ybase);

    }

/* Returns 0 if arguments have same sign, 1 otherwise */

    float compare_sign(m,fn)
    int m;
    float fn;
    {
    float fval;

    if(m*fn>0)
      fval=0;
    else
      fval=1;

    return(fval);

    }
```

# Appendix B: Publications and Presentations

## Publications

Y. Owechko, "Cascaded-Grating Holography for Artificial Neural Networks," accepted for publication in Applied Optics, 1992.

Y. Owechko, "Optical Implementation of Backpropagation Neural Network Using Cascaded-Grating Holography," to be submitted to International Journal of Optical Computing in 1992 (Invited Paper).

Y. Owechko and B. H. Soffer, "A Programmable Optical Neuro-Computer Based on Photorefractive Holograms," Proceedings of Government Microcircuit Applications Conference, Las Vegas, 1992.

Y. Owechko and B. H. Soffer, "Multi-Layer Optical Neural Networks," SPIE Proceedings Vol 1773, 1992.

Y. Owechko and B. H. Soffer, "An Optical Interconnection Method for Neural Networks Using Self-Pumped Phase Conjugate Mirrors," Optics Letters 16, 675-677 (1991).

G. J. Dunning, Y. Owechko, and B.H. Soffer, "Hybrid Opto-Electronic Neural Networks Using a Mutually-Pumped Phase Conjugate Mirror," Optics Letters 16, 928-930(1991).

Y. Owechko and B. H. Soffer, "Optical Neural Networks Based on Liquid Crystal Light Valves and Photorefractive Crystals," Proceedings of SPIE/SPSE Symposium on Electronic Imaging Science and Technology, San Jose, Feb. 1991, Vol. 1455, 136-144. (Invited Paper)

B.H. Soffer, Y. Owechko, and G.J. Dunning, "A Photorefractive Optical Neural Network," SPIE Proceedings Vol. 1347, 1, (1990). (Invited Paper)

B.H. Soffer, Y. Owechko, and G.J. Dunning, "A Photorefractive Optical Neural Network," Proceedings of 15th Congress of International Commission for Optics, Aug. 1990, Garmish, Germany.

## Presentations

Y. Owechko, "Programmable Optical Neural Networks," OSA 1992 Annual Meeting, Albuquerque, 1992. (Invited Talk)

Y. Owechko and B. H. Soffer, "A Programmable Optical Neuro-Computer Based on Photorefractive Holograms," Government Microcircuit Applications Conference, Las Vegas, 1992.

Y. Owechko, Lake Louis conference on electronic and optical implementations of neural networks, 1992.

Y. Owechko and B. H. Soffer, "Multi-Layer Optical Neural Networks," SPIE Conference 1773 on Photonics for Computers, Neural Networks, and Memories, San Diego, 1992.

Y. Owechko and B. H. Soffer, "Recent Advances in Optical Neural Networks Using Beam Fanning in Photorefractive Crystals," LEOS '91, San Jose, Nov. 1991. (Invited Talk)

Y. Owechko and B. H. Soffer, "Optical Neural Networks Based on Liquid Crystal Light Valves and Photorefractive Crystals," SPIE/SPSE Symposium on Electronic Imaging Science and Technology, San Jose, Feb. 1991. (Invited Talk)

Y. Owechko, G. Dunning, and B. Soffer, "Optical Neural Networks Based on Stimulated Photorefractive Effects," OSA 1990 Annual Meeting, Boston. (Invited Talk)

Y. Owechko, "Holographic Neural Networks," 1990 International Topical Meeting on Optical Computing, Kobe, Japan. (Invited Talk)

Y. Owechko, B.H. Soffer, and G.J. Dunning, "Photorefractive Optical Neural Networks," International Joint Conference on Neural Networks, Washington, D.C., 1990. (Invited Talk)

G.J. Dunning and Y. Owechko, "Multi-Port Optical Neural Network Using a Mutually Pumped Phase Conjugate Mirror," IEEE Meeting on Nonlinear Optics: Materials, Phenomena, and Devices, Kuau, 1990.

G.J. Dunning, Y. Owechko, and B.H. Soffer, "Optical Neural Network Using a Mutually Pumped Phase Conjugate Mirror," OSA Annual Meeting, Orlando, 1989.

Y. Owechko, "Self-Pumped Optical Neural Networks," OSA Topical Meeting on Optical Computing, Salt Lake City, 1989.

Y. Owechko, "Stimulated Photorefractive Optical Neural Networks," Third Annual Parallel Processing Symposium, Cal. State Fullerton, 1989. (Invited Talk)

# References


[1]V. Y. Kreinovich, "Arbitrary Nonlinearity is Sufficient to Represent All Functions by Neural Networks: A Theorem," Neural Networks Vol. 4, pp. 381-383 (1991).

[2]S. Judd, "Learning in Networks is Hard," Proceedings of IEEE International Conference on Neural Networks, San Diego, 1987, p. II-685.

[3]Y. Abu-Mostafa, "Appendix D: Complexity in Neural Systems," in Analog VLSI and Neural Systems by C. Mead, Addison-Wesley, 1989.

[4]D. Psaltis, D. Brady, and K. Wagner, "Adaptive optical networks using photorefractive crystals," Appl. Opt. **27**, 1752-1759 (1988).

[5]F. H. Mok, M. C. Tackitt, and H. M. Stoll (1991). "Storage of 500 high resolution holograms in a $LiNbO_3$ crystal," Opt. Lett. **16**, 605-607.

[6]H. Kogelnik (1969). "Coupled wave theory for thick holograms," Bell Sys. Tech. J. **48**, 2909-2946.

[7]D. Psaltis, D. Brady, X.-G. Gu, and S. Lin, "Holography in artificial neural networks," Nature **343**, 325-330 (1990).

[8]K. Rastani and W. M. Hubbard, "Large interconnects in photorefractives: grating erasure problem and a proposed solution," Appl. Opt. **31**, 598-605 (1992).

[9]G. P. Nordin, "Analysis of Volume Diffraction Phenomena For Photonic Neural Network Implementations and Stratified Volume Holographic Optical Elements", PhD. Thesis, University of Southern California, (1992).

[10]J. A. Fleck, J. R. Morris, and M. D. Feit, "Time-Dependent Propagation of High Energy Laser Beams Through the Atmosphere", Appl. Phys., **10**, 129-160, (1976).

[11]R. J. Collier, C. B. Burckhardt, and L. H. Lin, *Optical Holography*, Academic Press, Inc., New York (1971).

[12]D. Psaltis, X. G. Gu, and D. Brady, "Fractal Sampling Grids For Holographic Interconnections", SPIE Vol. 963, p. 468-474, (1988).

[13]C. X. G. Gu, "Optical Neural Networks Using Volume Holograms", PhD. Thesis, California Institute of Technology, (1990).

[14]C. Slinger, "Analysis of the N-to-N volume holographic neural interconnect," J. Opt. Soc. Am. A **8**, 1074-1081 (1991).

[15]G. P. Nordin, "Analysis of Volume Diffraction Phenomena For Photonic Neural Network Implementations and Stratified Volume Holographic Optical Elements", PhD. Thesis, University of Southern California, (1992).

[16] J. Feinberg, Opt. Lett. **7**, 486 (1982).

[17]J. Feinberg and K. R. MacDonald, in Photorefractive Materials and Their Applications II, Springer-Verlag, 1989.

[18]A. V. Nowak, T. R. Moore, and R. A. Fisher, J. Opt. Soc. Am. B, Vol. 5, No. 9, 1864-1878, (1988).

[19]Y. Owechko, Proceedings of Topical Meeting on Optical Computing, Salt Lake City, Feb. 1989, 44-47.

[20]Y. Owechko, Conference Record of 1990 International Topical Meeting on Optical Computing, Kobe, Japan, 142-146, (1990).

[21]M. D. Ewbank, Opt. Lett. **13**, 47 (1988).

[22]G. C. Valley, "Competition between forward- and backward-stimulated photorefractive scattering in $BaTiO_3$," J. Opt. Soc. Am. B **4**, 14-19 (1987).

[23]N. V. Kukhtarev, V. Markov, and S. Odulov, "Transient energy transfer during hologram formation in $LiNbO_3$ in external electric field," Opt. Comm. **23**, 338-343 (1977).

[24]Y. Taketomi, J. E. Ford, H. Sasaki, J. Ma, Y. Fainman, and S. H. Lee, "Incremental recording for photorefractive hologram multiplexing," Opt. Lett. **16**, 1774-1776 (1991).

[25]J. H. Hong, S. Campbell, and P. Yeh, "Optical pattern classifier with Perceptron learning," Appl. Opt. **29**, 3019-3025 (1990).

[26]U. Efron et al., "The CCD-addressed Liquid Crystal Light Valve - an update," Proc. SPIE 1455, 237-247 (1991).

[27]B. Kosko, "Adaptive Bidirectional Associative Memories," Appl. Opt. **26**, 4947-4960 (1987).

[28]K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks," Neural Networks **2**, 183-192 (1989).

[29]D. B. Parker, "Learning Logic," Invention Report S81-64, File 1, Office of Technology Licensing, Stanford Univ. (Oct. 1982).

[30]D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing, Vol. 1*, D. E. Rumelhart and J. L. McClelland, Eds., MIT Press, (1986).

[31]P. A. Shoemaker, M. J. Carlin, and R. L. Shimabukuro, "Backpropagation Learning With Trinary Quantization of Weight Updates," Neural Networks **4**, 231-241 (1991).