AFOSR-TR- 88-0999

(2)

ELECTRICAL ENGINEERING

ANNUAL PROGRESS REPORT

RESEARCH IN OPTICAL SYMBOLIC COMPUTING TASKS

Grant AFOSR-86-0196

Research Period 1 June 1987 - 31 May 1988

UNIVERSITY OF SOUTHERN CALIFORNIA

Signal and Image Processing Institute
Powell Hall of Engineering
University Park/MC-0272
Los Angeles, CA 90089

(2)

ANNUAL PROGRESS REPORT

RESEARCH IN OPTICAL SYMBOLIC COMPUTING TASKS

Grant AFOSR-86-0196

Research Period 1 June 1987 - 31 May 1988

SECURITY CLASSIFICATION OF THIS PAGE

UNCLASSIFIED

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	Approved for public release distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) AFOSR-TR- 88 0999
6a. NAME OF PERFORMING ORGANIZATION: Univ of So. CA; 6b. OFFICE SYMBOL (If applicable):	7a. NAME OF MONITORING ORGANIZATION: AFOSR/NE
6c. ADDRESS (City, State and ZIP Code): Signal And Image Processing Institute, University Park/MC 0272, Los Angeles, CA 9089	7b. ADDRESS (City, State and ZIP Code): Bldg 410, Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: AFOSR/NE; 8b. OFFICE SYMBOL (If applicable):	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: AFOSR-86-0196
8c. ADDRESS (City, State and ZIP Code): Bldg 410, Bolling AFB, DC 20332-6448	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
61102F	2305	B1	

11. TITLE (Include Security Classification): Optical Symbolic ~~Tasks~~ Research in Computing Tasks

12. PERSONAL AUTHOR(S): Jenkins

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Annual	FROM 01/06/87 TO 31/05/88		

16. SUPPLEMENTARY NOTATION

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

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

This report ~~We have~~ concentrated on the following topics: complexity studies for optical neural and digital systems and learning algorithms for neural networks. Several conference and journal papers reporting the research findings have been published. A list of publications and presentations is given at the end of the report along with a set of repreints and preprints.

Keywords: → See p 2

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION
UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS ☐	UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Giles	(202) 767-4931	NE

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

Contents

A-1

Abstract

The research findings of the AFOSR Grant AFOSR-86-0196, "Optical Symbolic Computing Tasks" are summarized for the period 1 June 1987 – 31 May 1988. Specifically, we have concentrated on the following topics: complexity studies for optical neural and digital systems and learning algorithms for neural networks. Several conference and journal papers reporting the research findings have been published. A list of publications and presentations is given at the end of the report along with a set of reprints and preprints.

1 Complexity of optical neural and digital systems

1.1 Connectivity and hierarchical neural networks

In neural networks the connectivity can be very high and in many cases the nets are even fully connected. As has been shown by Psaltis, even optical systems may not be able to provide this much connectivity for nets with large numbers of neuron units (i.e., 2-D arrays of neuron units). One technique for optically reducing the physical interconnection requirements is to take advantage of any symmetry or regularity in an interconnection. Since neural nets are particularly useful for random problems, and this may imply random interconnections, at first thought utilizing symmetry may not seem plausible. However, in many cases nets may have a hierarchical structure, and this may often imply some repetition in the interconnections. For example, a network that utilizes a number representation scheme with binary neurons may have the same interconnections repeated for each group of neurons. Most number representation techniques that have been described for neural networks do not have this repetition, but we have found that variants of them do. We have designed such network structures that have repeated blocks, one for each represented number, and have incorporated proper update rules for the neurons to ensure convergence of the net. This work has focused on single layer feedback networks used for combinatorial optimization. This yields a hierarchical network in the sense that each block represents the lower level, and the interconnections from block to block represent the higher level. It may also be extendable to hierarchies with more than two levels.

1.2 Digital optical parallel system complexity

Our study of digital optical system complexity has been continuing; in year 2 of this grant it has included a comparison of optical and electronic interconnection network complexity, and a study of design and complexity tradeoffs for the implementation of a shared memory parallel computer. The complexity of some common interconnection networks have been analyzed for optical and electronic VLSI implementations in detail. The optical system used for analysis was the hybrid 2-hologram interconnection system of Jenkins, et al. Area complexity was compared and found to be

	VLSI	OPTICS
Banyan	$0(n^2)$	$0(nlog^2n)$
Shuffle/Exchange	$0(n^2/logn)$	$0(n^2logn)$
Hypercube	$0(n^2)$	$0(nlog^2n)$
2-D Cellular Hypercube	$\geq 0(n^2)$	$0(n)$

It should be noted that the electronic results have received a great deal of work on using various clever tricks and algorithms to reduce the result to near optimum. The optics case was only investigated by us and can likely be reduced further by using different layouts. The Banyan and shuffle/exchange networks are isomorphic and for them, optics has lower complexity for large n. An example of how the optical complexity can be lowered can be seen in the hypercube network.

The 2-D cellular hypercube is identical to two overlapping hypercube networks, so it has twice as many interconnections, yet its optical area complexity is much lower because it is space-invariant.

We have more recently applied our expertise and results in complexity analysis to the use of optics in the implementation of a parallel digital shared memory computer. This machine encompasses electronic processing elements, an optical reconfigurable interconnection network, and optical, electronic, or hybrid memory modules. We have been studying optimal uses of optics in the interconnections the associated control techniques, and machine performance vs. all-electronic parallel machines (especially the IBM GF11 and RP3 shared memory machines). We have also been and continue to study the possible use of superposition as part of the interconnections and memory access in this machine, which may permit parallel, simultaneous read access to memory, as well as reduce the well-known contention problems in large interconnection networks with distributed control.

2 Learning Algorithms

2.1 Potential Difference Learning

We have developed a new learning algorithm, *potential difference learning.* It is based on a temporal difference of the neuron unit potential,

$$\Delta w_{ij} \propto \Delta p_i x_j$$

where Δw_{ij} is the weight increment from neuron j to neuron i, Δp_i is the temporal difference of potential, and x_j is the jth input to neuron i, for self-organization in neural networks. Depending on the time sequence of the input patterns during learning, it can learn based on the input patterns themselves or based on the time difference of input patterns. It has no weight overflow as with a strict Hebbian law. It can, with suitable presentation of the input patterns, also be used to unlearn or erase stored states in an associative memory without access to the individual weights and without reversing the sign of the learning gain constant.

We have simulated potential difference learning on two different networks: (1) an Amari network, i.e. a single layer fully connected network with feedback, used as an associative memory, and (2) a 3-layer network used as two associative memories with a hidden layer to relate pairs of stored vectors. These are described in a paper attached to this report.

2.2 Stochastic Learning Networks for Computer Vision

We have developed stochastic learning networks for an important problem in Computer Vision, viz, texture segmentation. Our approach is based on minimizing an energy function, derived

through the representation of textures as Markov Random Fields (MRF). We use the Gauss Markov Random Field (GMRF) to represent the texture intensities and an Ising model to characterize the label distribution. We first used an adaptive Cohen-Grossberg/Hopfield network to minimize the resulting energy function. The solution obtained is a local optimum in general and may not be satisfactory in many cases. Although stochastic algorithms like simulated annealing have a potential of finding a global optimum, they are computationally expensive. We have developed an alternate approach based on the theory of *learning automaton* which introduces *stochastic learning* into the iterations of the Hopfield network. This approach consists of a two stage process with learning and relaxation alternating with each other and because of its stochastic nature has the potential of escaping the local minima.

The learning part of the system consists of a team of automata A_s, one automaton for each pixel site. Each automaton A_s at site s maintains a time varying probability vector $\mathbf{P_s} = [p_{s1}..., p_{sL}]$ where p_{sk} is the probability of assigning the texture class k to the pixel site s. Initially all these probabilities are equal. At the beginning of each cycle the learning system will choose a label configuration based on this probability distribution and present it to the Cohen-Grossberg/Hopfield neural network described above as an initial state. The neural network will then converge to a stable state. The probabilities for the labels in the stable configuration are increased according to the following updating rule: Let k_s be the label selected for the site $s = (i, j)$ in the stable state in the n-th cycle. Let $\lambda(n)$ denote a reinforcement signal received by the learning system in that cycle. Then,

$$p_{s_{k_s}}(n+1) = p_{s_{k_s}}(n) + a\lambda(n)[1 - p_{s_{k_s}}]$$

$$p_{s_j}(n) = p_{s_j}(n)[1 - a\lambda(n)], \forall j \neq k_s$$

for all $s = (i, j), 1 \leq i, j \leq M$.

In the above equation 'a' determines the learning rate of the system. The reinforcement signal determines whether the new state is good compared to the previous one in terms of the energy function. Using the new probabilities, a new initial state is randomly generated for the relaxation network and the process repeats. The above learning rule is called **Linear Reward-Inaction** rule in the learning automata terminology. More details of this algorithm may be found in [1]. A preprint of this paper is attached.

We have tested this algorithm in classifying some real textured images. The results are summarized in [6]. The Hopfield network solution has a misclassification error of about 14% without learning. The error decreased to 6.8% when stochastic learning was introduced. When simulated annealing was tried the error rate is 6.3%, but the number of iterations were considerably more. In general stochastic algorithms seem to perform better than any deterministic scheme.

Currently we are working on extending these methods to do hierarchical segmentation and the preliminary results are quite promising. We are also investigating the possibility of extending this

approach to other vision problems such as computation of optical flow. Under partial support from this grant and the USC URI Center for the Integration of Optical Computing, we have developed an adaptive neural network based algorithm for a fundamental problem in image processing, viz, the restoration of a blurred and noise corrupted image. One of the important stages of the algorithm is learning the blur parameteres form prototypes of original and degraded images. Details of this algorithm along with restoration results were presented in a paper which appeared in a special section on neural networks [1].

3 List of Publications

1. Y.T. Zhou, R. Chellappa and B.K. Jenkins, "A Novel Approach to Image Restoration Based on a Neural Network," *Proc. IEEE First International Conference on Neural Networks,* San Diego, vol. IV, pp. 269-276 June 1987.

2. C.H. Wang and B.K. Jenkins "Potential Difference Learning and Its Optical Design," accepted for *Proc. Soc. Photo-Opt. Instr. Eng.,* vol. 882, January 1988.

3. Y.T. Zhou, R. Chellappa, A. Vaid, and B.K. Jenkins, "Image Restoration Using a Neural Network," IEEE Trans. Acoust. Speech and Signal Processing, vol. ASSP-36, pp. 1141-1151, July 1988.

4. C.H. Wang and B.K. Jenkins, "The Implementation Consideration of a Subtracting Incoherent Optical Neuron," *The IEEE International Conference on Neural Networks,* San Diego, July 1988.

5. B.K. Jenkins and C.L. Giles, "Superposition in Optical Computing," accepted for *Proc. ICO Topical Meeting on Optical Computing,* Toulon, France, to appear, August 1988.

6. B.S. Manjunath and R. Chellappa, "Stochastic Learning Networks for Texture Segmentation", (Accepted for Publication at the Twenty Second Annual Asilomar Conference on Signals, Systems and Computers), Pacific Grove, CA, Oct. 1988.

7. B.K. Jenkins and C.H. Wang, "Model for an Incoherent Optical Neuron that Subtracts," *Optics Letters,* submitted January 1988, to appear October 1988.

4 Oral Presentations

1. B.K. Jenkins, "Optical Computing: Status and Prospects," briefing given to Dr. Bernard Paiewonsky, Deputy for Advanced Technology, Air Force, The Pentagon, Washington, D.C., September 1987.

2. B.K. Jenkins and C.H. Wang, "Model for An Incoherent Optical Neuron that Subtracts," annual meeting of the Optical Society of America, paper PD4, Rochester, New York, October 1987.

3. B.K. Jenkins and C. Lee Giles, "Massively Parallel Optical Computing," IEEE Communications Theory Workshop, Sedonna, Arizona, April 18-21, 1988.

4. B.K. Jenkins, "Optical Computing: Status and Prospects," IEEE Orange County Chapter meeting, Santa Ana, California, May 1988.

Model for an Incoherent Optical Neuron that Subtracts

B. K. Jenkins and C. H. Wang

Signal and Image Processing Institute, Department of Electrical Engineering

University of Southern California, Los Angeles, CA 90089-0272

Abstract

An **Incoherent Optical Neuron (ION)** is proposed that subtracts inhibitory inputs from excitatory inputs optically by utilizing two separate device responses. Functionally it accommodates positive and negative weights, excitatory and inhibitory inputs, nonnegative neuron outputs, and can be used in a variety of neural network models. An extension is given to include bipolar neuron outputs in the case of fully connected networks.

* submitted to Optics Letters, Dec. 1987.

To Appear, Oct. 1988

In this letter we propose a general incoherent optical neuron (ION) model that can process excitatory and inhibitory signals optically without electronic subtraction. Conceptually, the inhibitory signal represents a negative signal with a positive synaptic weight or a positive signal with a negative synaptic weight. The ION can be used in a network in which the neuron outputs are nonnegative and the synaptic weights are bipolar, for example, by connecting the interconnections with negative weights to the inhibitory neuron inputs and those with positive weights to the excitatory inputs. Our intent is to show that it is in principle not necessary to go to opto-electronic devices solely because of the requirement for subtraction capability.

Techniques that have been described to date are impractical in all-optical implementations of most neural networks. They utilize an intensity and/or weight bias, in some cases coupled with complemented weights or inputs. As noted in [1], these techniques suffer from bias buildup and/or thresholds that must vary from neuron to neuron. A technique described in [2] eliminates most of these drawbacks in the special case of fully connected networks.

The ION model uses separate device responses for inhibitory and excitatory inputs. This is modeled after the biological neuron which processes the excitatory and inhibitory signals by different mechanisms (e.g. chemical-selected receptors and ion-selected gate channels) [3]. The ION comprises two elements: an inhibitory (I) element and a nonlinear output (N) element. The inhibitory element provides inversion of the sum of the inhibitory signals; the nonlinear element operates on the sum of the excitatory signals, the inhibitory element output, and an optical bias to produce the output of the neuron. The inhibitory element is linear; the nonlinear threshold of the neuron is provided entirely by the nonlinear output element. Fig. 1(a) and 1(c) show the characteristic curve of the I and N elements respectively. The structure of the ION model is illustrated in Fig. 1(d). The input/output relationship of the I and N elements are

$$I_{out}^{(I)} = \hat{I}_{inh} = 1 - I_{inh} \tag{1}$$

$$I_{out}^{(N)} = \psi(I_{in}^{(N)} - \alpha) = \psi(\hat{I}_{inh} + I_{exc} + I_{bias} - \alpha) \tag{2}$$

where I_{inh} and I_{exc} represent the total inhibitory and excitatory inputs, $I_{in}^{(N)}$ is the total input to the N elements, I_{bias} is the bias term for the N element, which can be varied to change the threshold, and α is the offset of the characteristic curve of the N element. $\psi(\cdot)$ denotes the nonlinear output function of the neuron. If we choose I_{bias} to be $\alpha - 1$, the output of the N element is

$$I_{out}^{(N)} = \psi(I_{exc} - I_{inh}) \tag{3}$$

which is the desired subtraction. In general, the I element will not be normalized (Fig 1(b)), in which case the offset, a_1, and slope of its response can be compensated by setting $I_{bias} = \alpha - a_1$ and attenuating the output of the I element by a factor b_1/a_1, respectively. The unnormalized I element must have gain greater than or equal to 1. A nonzero neuron threshold, θ, can be implemented by shifting the bias by the same amount, so $I_{bias} = \alpha - a_1 - \theta$ for the unnormalized I element.

The ION model can be implemented by using separate devices for the I and N elements as depicted in Fig. 1 (heterogeneous case), or by using a single device with a nonmonotonic response (Fig. 2) to implement both elements (homogeneous case). Possible devices for ION implementation include bistable optical arrays and spatial light modulators such as liquid crystal light valves. A single Hughes liquid crystal light valve could implement both elements. The offset of the device response must satisfy $\alpha \geq na_1 + \theta$, where $n = 1$ for a heterogeneous implementation and $n = 2$ for a homogeneous implementation.

A device to realize the inhibitory (I) element will of course not have a perfectly linear response. To assess the robustness of this model to *non*linear I elements, and compensation techniques for large deviations from ideal response, we have performed simulations of a network similar to Grossberg's competitive network [4] for edge detection. The simulated network contains 30 conventional inner product neurons connected in a ring structure with input and lateral on-center off-surround connections. We model the normalized I element response as $exp[-(x/a)^b]$, where a and b are parameters that determine the specific nonlinear response. This provides insight into the sensitivity of the ION model to nonlinearities in the I element response, without being overly specific to one given device. A suitable choice of a and b does provide a close fit to the inversion region of the normalized experimental characteristic of a liquid crystal light valve (LCLV). By adjusting the parameters a and b, four different nonlinear inversion curves were simulated in this network. In the simulation a compensating attenuator was used before the I element instead of after it. The N element response, which provides a close approximation to the normalized increasing portion of the LCLV response (Fig. 2), is modeled as $1 - exp[-(x/0.43)^{1.2}]$. Fig. 3 shows the computer simulated responses of this network. Each resolvable row of the figure represents a 1-D simulation on a distinct 1-D input. Thirty different binary inputs were each simulated at four different input signal levels. Fig. 3(b) gives the ideal output, and (d) simulates a response that is close to the experimental response of our LCLV. Deviation from linearity of the I element is measured by normalized mean square error (*nmse*), which is defined as $\int[v(i) - \hat{v}(i)]^2 \mathrm{d}i / \int v(i)^2 \mathrm{d}i$, where $v(i)$ and $\hat{v}(i)$ are the output value of the linear and simulated nonlinear characteristic curves. The input level i ranged from 0 to 0.7. Our LCLV characteristic has an *nmse* of 50%, which does not perform well. If proper input attenuation of the I element is included, the network performs correctly. Four nonlinear curves are simulated, each with optimal input attenuation; we find that deviations from linearity that give an *nmse* of approximately 15% (measured after input

attenuation) can be tolerated. For more extremely nonlinear devices, a bias point and limited region of operation can be used.

The fan-in and fan-out of the ION, neglecting interconnection effects such as crosstalk, can be calculated as follows. (Interconnection effects are important but are not peculiar to the ION model.) We assume binary neurons. As shown in Fig. 1(c), the output of the i-th neuron can be formulated as $I_r + \Delta I_s^{(N)} V_i$, where $V_i \in \{0,1\}$ is the output state of the neuron i and I_r is the residual output of element N. Let the fan-in and fan-out of each neuron be N_{in} and N_{out} respectively. The summed inputs to neuron j can be grouped into two terms, a noise term caused by residual outputs (I_r) of the optical neurons and the signal term. Consider the worst case, i.e. all weights are close to one and only one input is active. If we assume each neuron must be able to discriminate a change in any one of its input lines, then the signal term must at least be greater than the noise term. This is a reasonable assumption for networks with small fan-in and fan-out. Thus the maximum fan-in is

$$N_{in}^{(max)} = \frac{\Delta I_s^{(N)}}{I_r} = \text{extinction ratio of element N.} \tag{4}$$

The fan-out is calculated from the I element, as shown in Fig. 1(b). The ratio of the maximum input (a_1) to the minimum input $I_s^{(N)}/N_{out}^{(max)}$ is the fan-in $N_{in}^{(max)}$, where $N_{out}^{(max)}$ is the maximum fan-out over all neurons, thus

$$N_{out}^{(max)} = \frac{I_s^{(N)}}{a_1} N_{in}^{(max)} \approx \frac{\Delta I_s^{(N)}}{a_1} N_{in}^{(max)} \tag{5}$$

where the approximation holds when the extinction ratio of the N element is large. For networks with large fan-in, we assume instead that the neuron can discriminate a change in a constant fraction β of its input signals. In this case, there are no such limitations on N_{in} and N_{out}. Instead, $1/\beta$ is limited by the extinction ratio of the N element, and the fan-out is still related to the fan-in of the network by

Eq. (5). For example, in many networks a $1/\beta$ ranging from 10-100 may be sufficient for the optical neuron while the maximum fan-in may be $10^3 - 10^4$. During implementation of the ION model, with many optical devices I_s can be varied with the intensity of the read beam, which effectively increases the gain and permits a larger fan-out.

As an example, a conceptual diagram of an implementation of a single layer feedback net is shown in Fig. 4. It utilizes a single 2-D SLM for both I and N elements. The output of the I element is imaged onto the input of the N element, after passing through a ND filter as the (uniform) attenuation. A uniform bias beam is also input to the N element. The N element output is fed back through an interconnection hologram to the inputs of both I and N elements, representing inhibitory and excitatory lateral connections, respectively.

In the remainder of this letter we will present a variant of the ION model that incorporates bipolar neuron outputs in the case of fully connected networks. The operation of the network is given by

$$\hat{V}_i = \psi[\sum_{j=1}^{N} W_{ij} V_j] \tag{6}$$

where $V_j \in [-1, +1]$ is output of the j^{th} neuron, $W_{ij} \in [-1, 1]$ is the normalized weight from neuron j to neuron i and N is the number of neurons. A special case of this is the bipolar binary neuron used by Amari (1972) [5] ($V_j \in \{-1, +1\}$). In this case, the nonlinear output function $\psi(x)$ is equal to 1 for $x \geq 0$, otherwise it is -1.

By a complementary offset scheme, Eq. (6) can be rewritten as

$$\frac{1}{2}(1 + \hat{V}_i) = \psi[\sum_{j=1}^{N} \frac{(1 - W_{ij})}{2}\frac{(1 - V_j)}{2} + \sum_{j=1}^{N} \frac{(1 + W_{ij})}{2}\frac{(1 + V_j)}{2}] \tag{7}$$

where $\psi(x)$ is the nonlinear output function of the neuron. All terms in parentheses are positive and can be represented by intensities. The neuron input and output are in the form $(1 + V_i)/2$, and

the I element is used to generate the $(1 - V_i)/2$ term. A Hopfield net [6] is identical except the neuron outputs $V_i \in \{0, 1\}$; for this we can replace $(1 + V_i)/2$ with V_i and $(1 - V_i)/2$ with $\tilde{V}_i$ in Eq. (7), where $\tilde{V}_i$ is the complement of V_i, and is generated by the I element.

Most of this work was presented at the 1987 Annual Meeting of Optical Society of America [7].

References

[1] A. F. Gmitro and G. R Gindi, *Proc. IEEE First Int. Conf. on Neural Networks*, III-599, San Diego (1987).

[2] R. D. Te Kolste and C. C. Guest, *Proc. IEEE First Int. Conf. on Neural Networks*, III-625, San Diego (1987).

[3] M. Wang and A. Freeman, "Neural function", *Little, Brown and Company* press (1987).

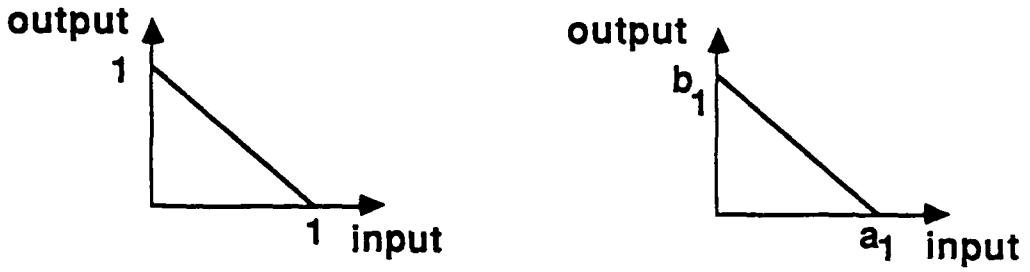
[4] S. A. Ellias and S. Grossberg, *Biol. Cybernetics* **20**, 69 (1975).

[5] S-I Amari, *IEEE Trans. on Computers* **C-21**, 1197 (1972).

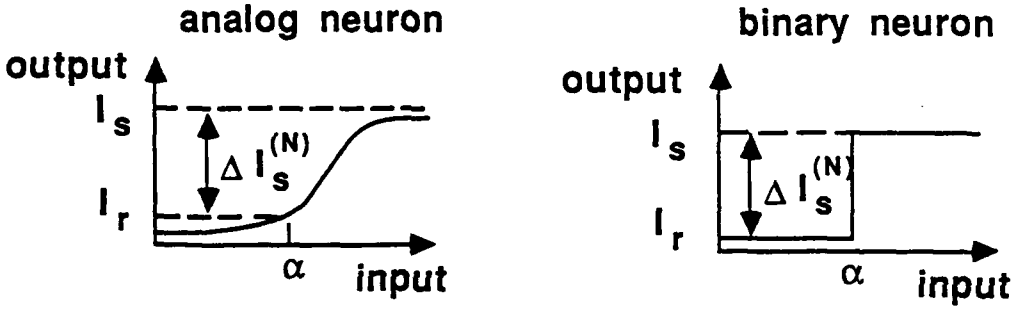
[6] J. J. Hopfield, *Proc. Natl. Acad. Sci. USA* **79**, 2554 (1982).

[7] B. K. Jenkins and C. H. Wang, *J. of Opt. Soc. Amer. (A)* **4**, 127A, paper PD6 (Dec. 1987).

(a) The I (Inhibitory) element (b) The Unnormalized I Element:



(c) The N (Nonlinear) Element



(d). The ION Structure:

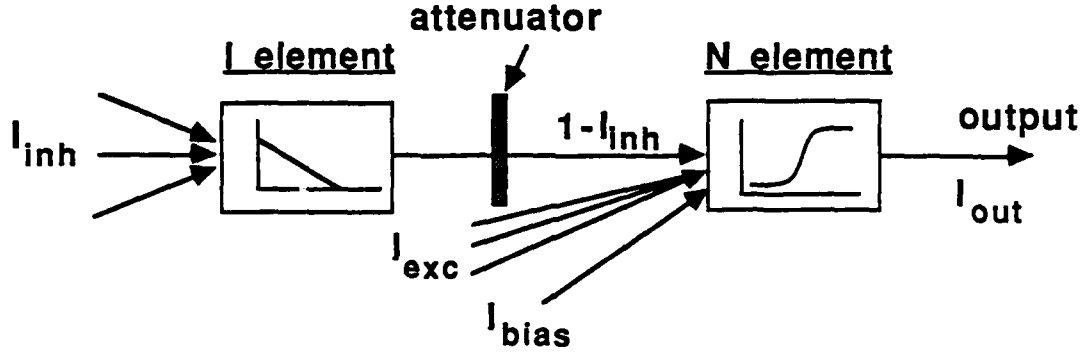


Fig. 1 The ION: (a)-(c) its components, and (d) its structure.

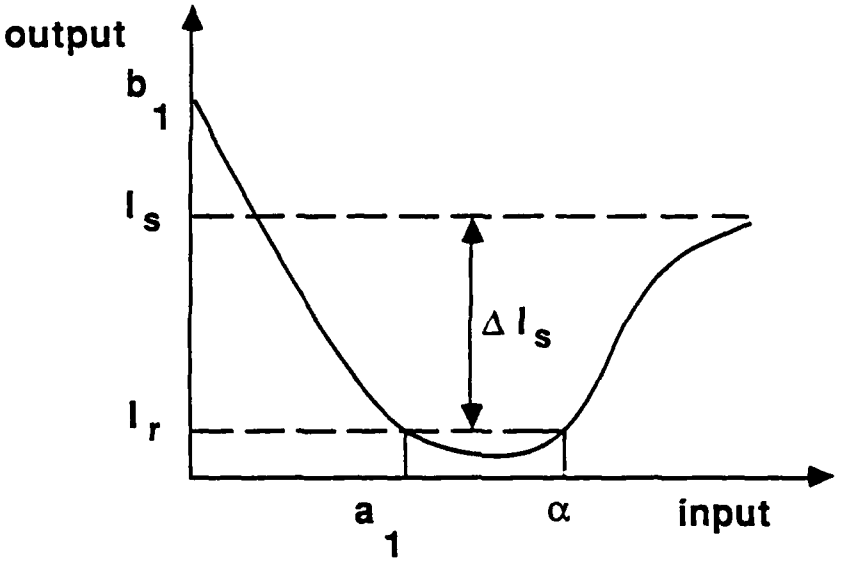


Fig. 2 Characteristic of a Huges twisted-nematic liquid crystal light valve, a possible device for the homogeneous ION model.

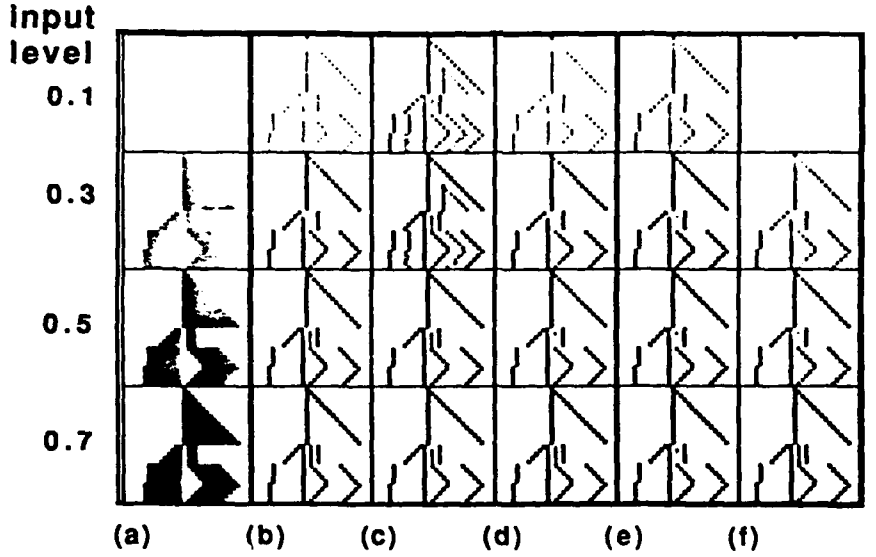


Fig. 3 Simulations of imperfect I elements in the 1-D on-center off-surround competitive network. (s1 is the input attenuation factor for the I element; mse is the normalized mean square error deviation from linearity of the I element response.) (a) network input. (b)-(f) network outputs for:
(b) linear I element
(c) a=0.46,b=2.1,s1=1.0,mse=19%
(d) a=0.25,b=1.2,s1=2.5,mse=4%
(e) a=0.33,b=1.5,s1=1.5,mse=14%
(f) a=0.16,b=0.9 ,s1=5.0,mse=7%

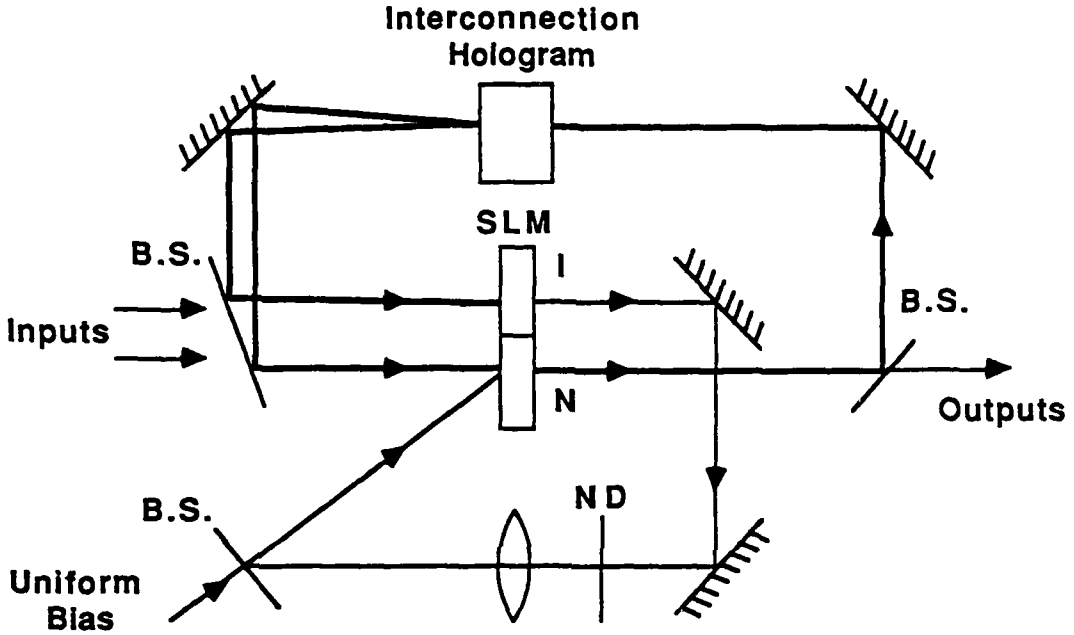


Fig. 4 Single layer feedback net using a single SLM to implement both I and N elements.

Proc. SPIE, vol. 882, Jan. 1988.

Potential Difference Learning and Its Optical Architecture

C. H. Wang and B. K. Jenkins

Siginal and Image Processing Institute, Department of Electrical Engineering,
University of Southern California, Los Angeles, CA 90089-0272

ABSTRACT

A learning algorithm based on temporal difference of membrane potential of the neuron is proposed for self-organizing neural networks. It is independent of the neuron nonlinearity, so it can be applied to analog or binary neurons. Two simulations for learning of weights are presented; a single layer fully-connected network and a 3-layer network with hidden units for a distributed semantic network. The results demonstrate that this potential difference learning (PDL) can be used with neural architectures for various applications. Unlearning based on PDL for the single layer network is also discussed. Finally, an optical implementation of PDL is proposed.

1. INTRODUCTION

Most of the unsupervised learning algorithms are based on Hebb's hypothesis [1], which depends on the correlated activity of the pre- and postsynaptic nerve cells. For steady input patterns, Hebb's rule will suffer from weight overflow. Von der Malsburg (1973) [2] solved this by adding the constraint that the sum of the weights of a neuron is constant. This concept led to competitive learning, developed by Grossberg (1976) [3], and Rumelhart and Zipser (1985) [4]. They also assumed a winner-take-all algorithm (Fukushima 1975 [5]) to enhance the synaptic weight modification between neurons. Biologically, the sum of the weights of a neuron can likely be changed by the supply of some chemical substance. In this paper we propose a learning algorithm, potential difference learning (PDL), based on temporal difference of the neuron membrane potential. Because PDL is based on the membrane potential, it is independent of the nonlinear threshold function of the neuron. Its temporal characteristic prevents weight overflow and permits unlearning without access to individual weights.

In an artificial neural system, unlearning can provide for real time reprogramming and modification of the distributed storage for stable recollection, or equivalently, modification of the energy surface in an energy minimization problem. Hopfield proposed unlearning to reduce the accessibility of spurious states [6]. Our unlearning emphasizes reprogrammability and local modification of the energy surface for stable partial retrieval. The unlearning in PDL is done by presenting a sequence of patterns and global gain control; reversing the sign of the learning gain is not necessary. The distinction of learning and unlearning in PDL is in the data sequence and value of the gain constant for different phases.

The main advantages of potential difference learning are spontaneous learning without weight overflow for steady state input patterns and unlearning. Other features of PDL include contrast learning, temporally correlated and uncorrelated learning, learning independently of neuron type and ease of physical implementation.

2. POTENTIAL DIFFERENCE LEARNING AND ITS PROPERTIES

Like most learning rules, potential difference learning requires only local information for synapse modification. Given a neuron with n inputs, PDL is given by:

$$\underline{w}(k+1) = \Phi[\underline{w}(k) + K_a \alpha^{-1}(k) \cdot \Delta p(k) \cdot \underline{x}(k)] \quad (1)$$

$$\Delta p(k) \equiv \underline{w}^T(k)\underline{x}(k) - \underline{w}^T(k-1)\underline{x}(k-1) \quad (2)$$

$$y(k) = \Psi[\underline{w}^T(k)\underline{x}(k) - \theta(k)] \quad (3)$$

* Presented at SPIE's O-E Lase '88, Los Angeles, California, 10-15 January, 1988. "Neural Network Models for Optical Computing", SPIE vol. 882.

Where $\underline{w}(k)$ and $\underline{x}(k)$ are the input weights and stimuli respectively; they are represented as n x 1 vectors. $p(k)$ and $y(k)$ are the neuron potential and output value at time instant k. $\theta(k)$ is the threshold of the neuron and $K_a\alpha^{-1}(k)$ denotes the learning gain constant with K_a as the global gain constant and $\alpha^{-1}(k)$ as the adaptive gain constant. The weight $\underline{w}(k)$ is bounded by the function $\Phi(\cdot)$, which represents the physical limitation of synapses. Distinct from other learning models, PDL is independent of the output nonlinear function $\Psi(\cdot)$ of the neuron.

PDL has the following properties:

- (1). Self-organization: similar to Hebb's rule, PDL can modify the weights of synapses according to the input patterns.
- (2). Contrast learning: Weight modification is initiated by potential *difference*, which is caused by difference in input. Since most sensory preprocessing is differential in nature, PDL can provide a good approach for feature extraction when it is combined with neural architectures.
- (3). Unlearning: This can be used to erase stored states, to alter the energy surface or reprogram a network. PDL can provide unlearning capability by applying suitable pattern sequences to generate a negative potential difference Δp. This is discussed below.
- (4). Temporally correlated or uncorrelated learning: By varying the training sequences, the neuron can learn absolute patterns or temporal differences between training patterns. The temporal difference property may be useful in sensory information processing.
- (5). Ease of implementation: The PDL uses only local information to update the weights and only one differencer per neuron is needed to calculate the potential difference. The complexity is low when it is compared with differential Hebbian learning (Kosko 1986) [8] or drive-reinforcement learning (Klopf 1986) [9].
- (6). Independence from neuron nonlinearity: The learning rule is evoked by the potential change only, so various non-linear functions can be imposed on the neuron to make different types of neurons. Due to this feature, weight modification can still occur when the output is saturated or clamped as long as the weight is not saturated.

A variant of PDL is given by

$$\underline{w}(k+1) = \Phi[\underline{w}(k) + K_a\alpha^{-1}(k) \cdot \Delta y(k) \cdot \underline{x}(k)] \tag{4}$$

which replaces potential difference $\Delta p(k)$ with output difference $\Delta y(k)$. This can be used when the neuron potential is not physically available. The tradeoff is that weight modification no longer occurs when the neuron output is saturated. Equation (4) is similar in appearance to supervised learning (Widrow-Hoff rule), but here $\Delta y(k)$ refers to the temporal difference of the neuron output, instead of the spatial output error.

Other learning algorithms have been proposed based on the following:

$$\underline{w}(k+1) = \Phi[\underline{w}(k) + K_a\alpha^{-1}(k) \cdot \Delta y(k) \cdot \Delta \underline{x}(k)] \tag{5}$$

with Δ representing different forms of temporal difference [7], [8], [9]. The use of $\Delta\underline{x}(k)$ instead of $\underline{x}(k)$ and more complex definitions of time average in these learning rules causes a higher implementation complexity.

Due to the fact that the PDL rule is embedded in the neurons, we need some lateral interconnections between the neurons of the same layer to enhance the competitive or cooperative modification of synapses. One example is to use the winner-take-all algorithm [5]:

$$\underline{w}_j(k+1) = \Phi[\underline{w}_j(k) + K_a\alpha^{-1}(k) \cdot \Delta p_j(k) \cdot \underline{x}_j(k) \cdot \delta(y_j(k))] \tag{6}$$

where the subscript j denotes neuron j, and $\delta[y_j(k)] = 1$ if j^{th} neuron wins in his neighborhood, otherwise it is zero.

3. COMPUTER SIMULATIONS

First a single layer fully interconnected neural network, as described by Amari [10], Hopfield [11] and others, is simulated. Four input patterns [12], each 20 bits long, are presented to the external input of the network. The learning rule is our PDL with K_a=0.01 and $\alpha^{-1}(k)$=1. The neurons are binary with bipolar coding (+1,-1). The weights are initially set to zero. For each iteration, the four inputs were presented in sequence. Four iterations were performed. The resulting weight matrix is shown in Fig. 1 (a). PDL produces a near symmetric weight matrix, which is quite similar to the result obtained using the familiar sum of outer products, as shown in Fig. 1 (b). If a partial input is applied to the trained network, we can get full retrieval after several iterations, dependent on the hamming distance from the partial input.

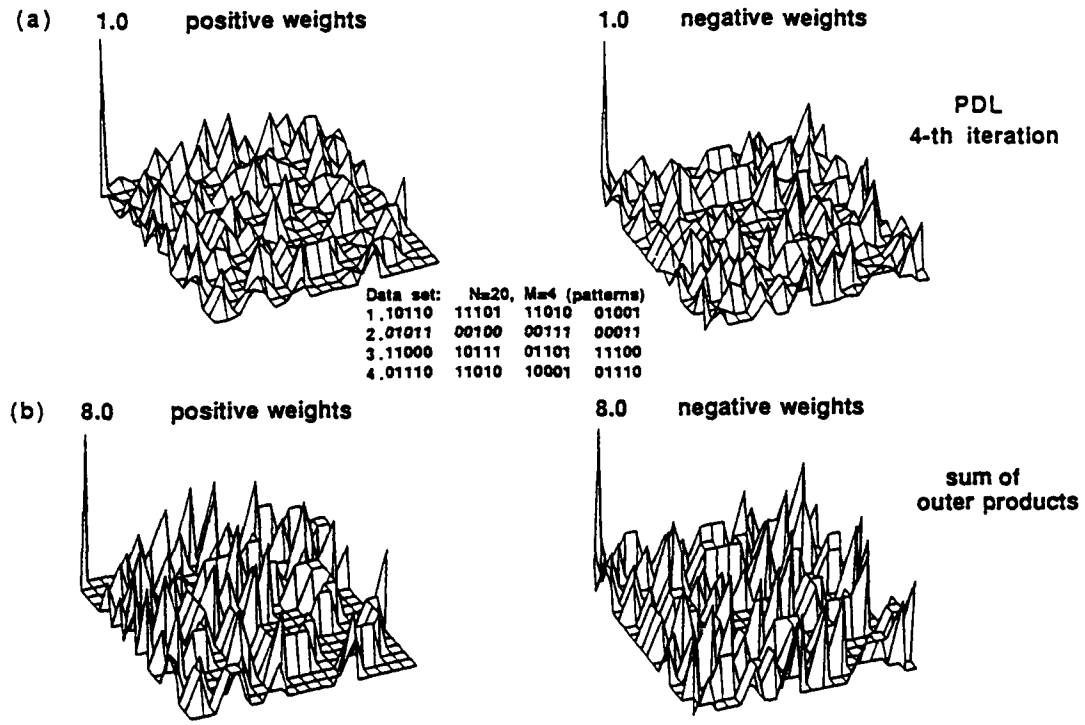


Fig. 1 Comparison Between Outer Product T(i,j) Matrix and PDL (20 neurons, 4 patterns)

The unlearning procedure of this network is divided into two stages. (1). Apply the data to be erased to the network with low (zero) global gain constant. (2). Use the same gain constant as for learning. In each step, present the input with one bit complemented; allow Δp to decrease to zero; restore that bit and complement the next bit for the next step. After all bits have been complemented, one iteration is completed. Starting with the trained weights of Fig. 1(a), two of the stored vectors (pattern #3 and #4) were erased using this unlearning procedure. We erase pattern #4 in five iterations, then erase pattern #3 in another five iterations. After each iteration, we test the convergence of the erased pattern. The resulting network would not converage to the erased states after just three iterations. For five iterations of unlearning, the weight matrix, Fig. 2(a), is very close to the original weight matrix that stored only pattern #1 and #2 as shown in Fig. 2(b). To measure the performance of unlearning, the resulting weight matrix is normalized by dividing it by a factor F, which is

$$F = \frac{\sqrt{\sum_{i,j} T_U^2(i,j)}}{\sqrt{\sum_{i,j} T_I^2(i,j)}} \tag{7}$$

where $T_U(i,j)$ is the resulting weight matrix after unlearning and $T_I(i,j)$ is the ideal weight matrix. Then

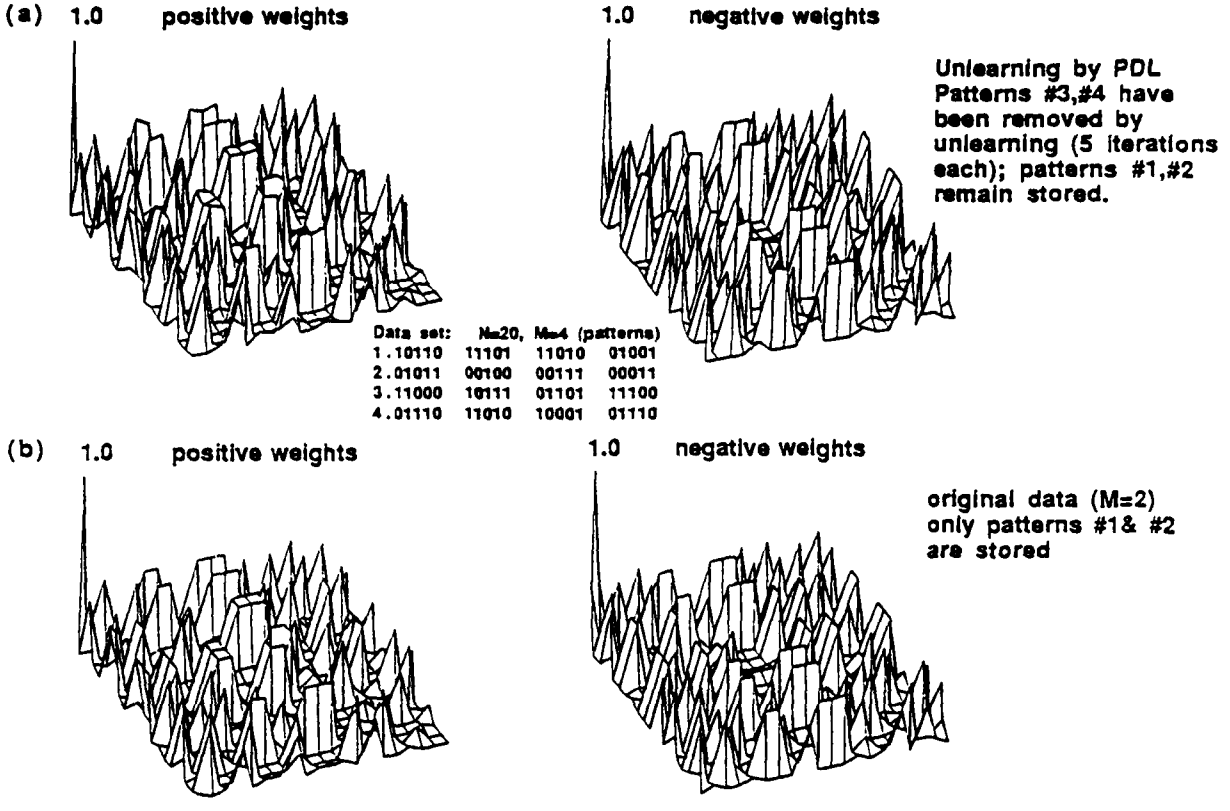


Fig. 2 Unlearning of PDL from M=4 to M=2, N=20. Weight matrices (a) after unlearning, (b) ideal result.

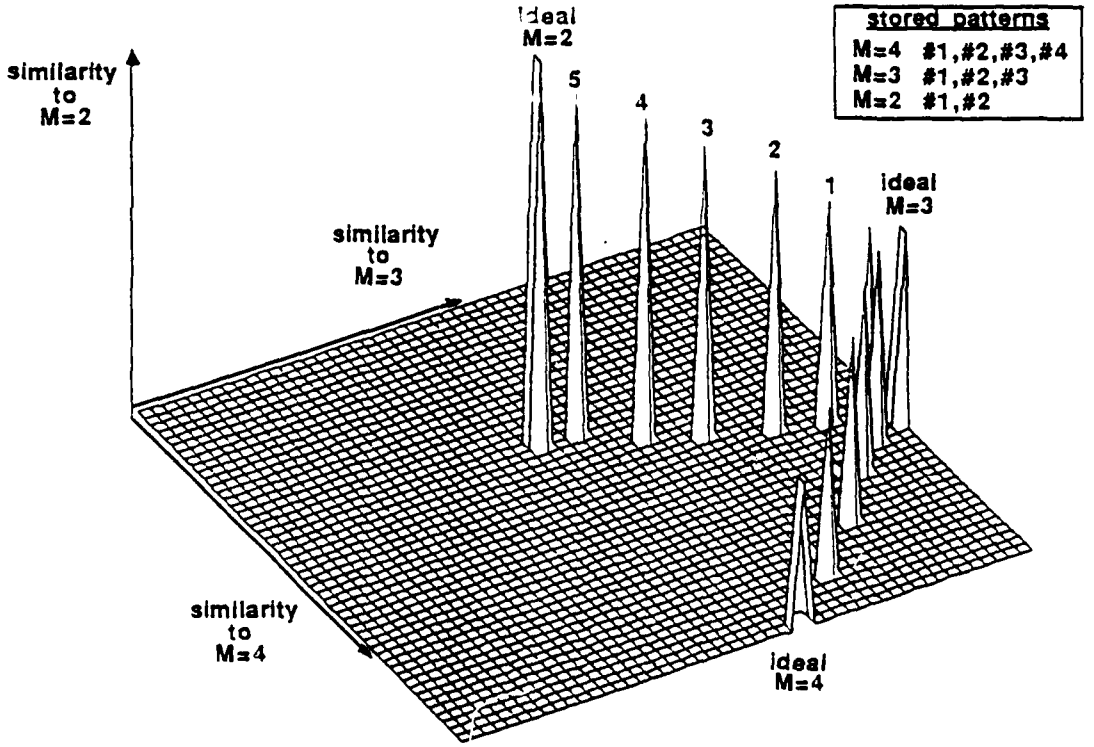


Fig. 3 Unlearning of PDL from M=4 to M=2 , N=20. Plot of similarity measures after each iteration. (5 iterations total for each unlearned pattern; numbered in sequence for unlearning of pattern #3). The ideal expected results for M=3 and M=2 are labelled.

a similarity measure, which is defined as the ratio of the matrix 1-norm [13] of these two weight matrices, is applied to evaluate the performance. Fig. 3 shows the similarity measure of the weight matrix after each iteration when unlearning using PDL from initially M=4 stored vectors to M=2 stored vectors.

The second example is a 3-layer network with two fully interconnected visible layers and a hidden layer. The purpose of this network is to do associative mapping between two visible layers by using one hidden layer. The visible layers are fully connected and the interconnections between the visible layers and the hidden layer are shown in Fig. 4. The visible layers use binary neurons to interface with the environment, while the hidden layer uses analog neurons. One neuron is used to calculate the average output, $u(k)$, of the hidden layer; then the competitive network of Fig.5, which is used in the hidden layer, reinforces those neurons with stronger output.

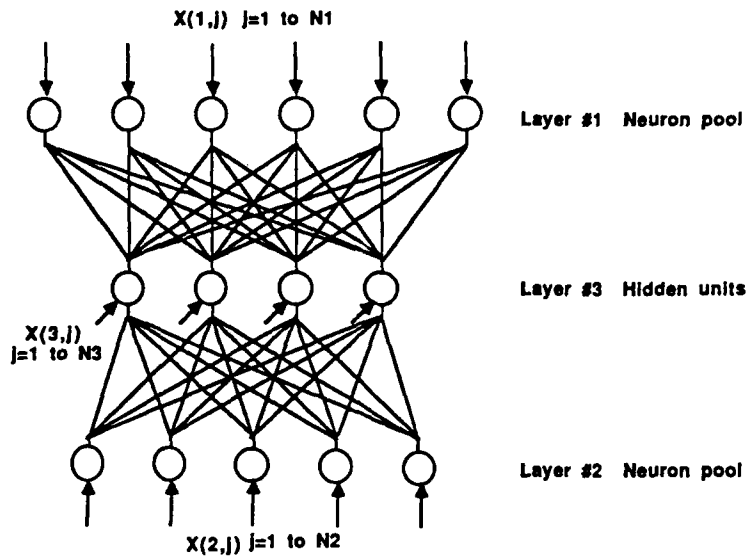


Fig. 4 Interconnections between hidden layer and visible layers. This interconnections are bidirectional with possibly different weights in each direction. Each visible layer is fully connected.

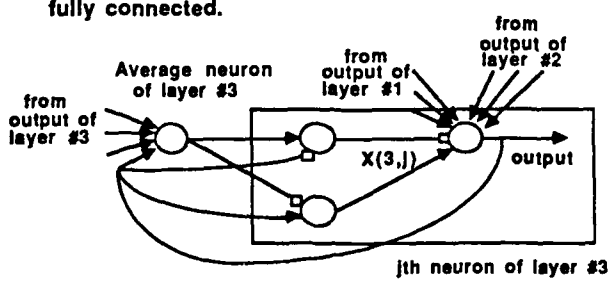


Fig. 5 competitive interconnections of the hidden layer.

The operation of the hidden layer is

$$u(k) = \sum_{j=1}^{N_3} \frac{1}{N_3} y_j^{(3)}(k) \tag{8}$$

$$y_j^{(3)}(k+1) = \psi\{\sum_{i=1}^{N_1} w_{ij}^{(1)} y_i^{(1)}(k) + \sum_{i=1}^{N_2} w_{ij}^{(2)} y_i^{(2)}(k) + \psi[y_j^{(3)}(k) - u(k)] - \psi[u(k) - y_j^{(3)}(k)]\} \tag{9}$$

where $\psi(x)$ is 1 for $x \geq 1$, and is 0 for $x < 0$, else it is x. Superscripts denote the layer number and N_1, N_2, N_3 represent the number of neurons in layer 1, 2, and 3. $w_{ij}^{(l)}$ for $l = 1, 2$ represents the weight from the i^{th} neuron of layer l to the j^{th} neuron of the hidden layer. Initially, the weights of the visible layers are set

to zero and the weights between the visible layers and the hidden layer are set to small random values. The visible layers are trained separately during the first phase, while the learning gain of the hidden layer is set to zero. In the second phase, the learning gain of the hidden layer and the visible layers is nonzero; we apply the corresponding patterns at these visible layers to train the hidden layer and the visible layers. After the learning phases, applying a partial input at the visible layer #1 will retrieve the full information at the same layer and associated data at the other visible layer. We have performed computer simulation of this network with 20 neurons in layer #1, 16 neurons in layer #2, and 10 neurons in the hidden layer. Eight patterns were stored, four into layer #1 and four into layer #2, and associations between pairs of these patterns were learned. The network randomly selected a set of one or more "representation" neurons in the hidden layer to form an association between each pair of patterns. Some of the sets of neurons for different pairs of patterns were disjoint, and some were partially overlapping. Table 1 shows the hidden neurons selected by the network for each associated pair of patterns. The last column in the table shows the pattern retrieved upon presentation of each layer #1 pattern. Since each set of representation neurons usually consists of multiple neurons, some fault tolerance is provided. However, when these sets overlap some interference can result during retrieval of associated patterns. This imperfect mapping results from the "soft" competitive network that was used.

Table 1 Simulation results of network of Fig. 4 and Fig. 5.

pattern stored in layer #1	resulting representation neurons in hidden layer	pattern stored in layer #2	pattern retrieved in layer #2
$\underline{a}_1$	1,4	$\underline{b}_1$	$\underline{b}_1$
$\underline{a}_2$	2,9	$\underline{b}_2$	$\underline{b}_2$
$\underline{a}_3$	2,3,5	$\underline{b}_3$	$\underline{b}_3$
$\underline{a}_4$	1,4,6	$\underline{b}_4$	$\underline{b}_1$

4. OPTICAL ARCHITECTURE OF PDL

A conceptual diagram of an optical implementation of PDL is shown in Fig. 6; it is somewhat similar to Fisher's associative processor [14]. Two spatial light modulators (SLMs) are used, one for storage of the weight matrix and one for generation of $\Delta\underline{w}(k)$. In addition, two 1-D storage devices and one 1-D threshold device are used.

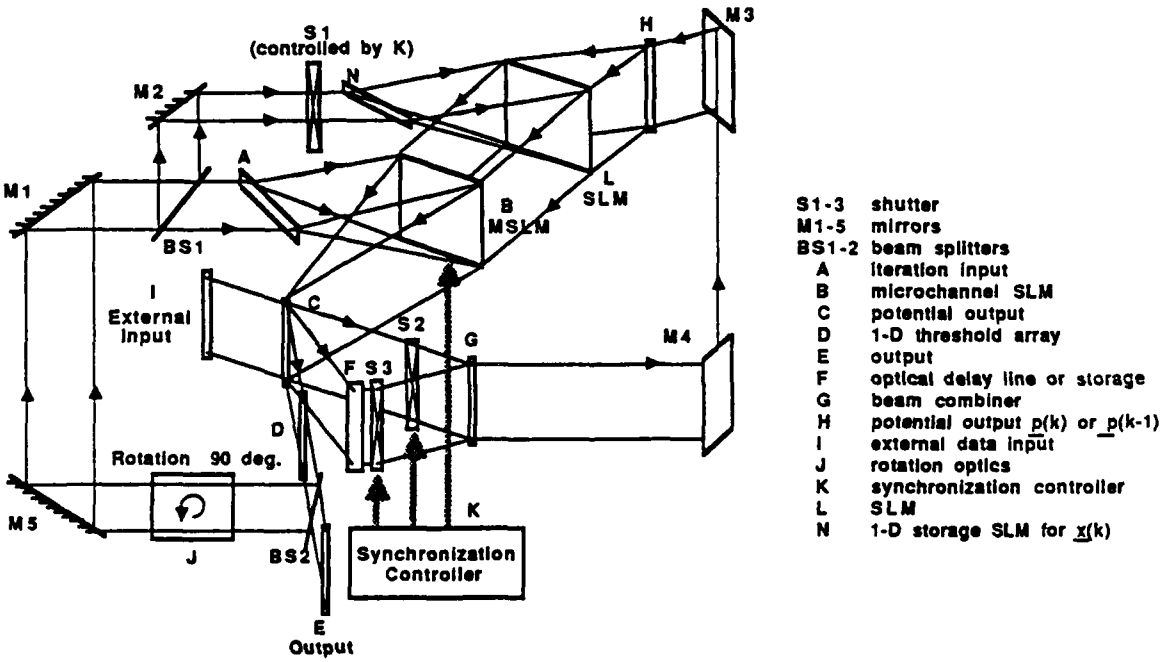


Fig. 6 Conceptual diagram of an optical implementation of potential difference learning.

"A" is the input $\underline{x}(k)$ from the previous iteration, which is expanded vertically to illuminate the weight storage "B". The reflected output from the microchannel SLM "B" is collected horizontally. This represents a vector, each component of which is $\sum w_{ij}x_j$. It is then combined with external input "I" to produce potential output at position "C". The output at "C" is split into three. The first one passes through 1-D threshold device "D" to generate outputs of the neurons. The second output of "C" passes through delay element F and shutter S3, yielding $\underline{p}(k-1)$ at "G". The third output path from "C" passes through shutter S2 to yield $\underline{p}(k)$ at "G". Only one of the shutter arrays S2 and S3 can be turned on at a time. "G" is a beam combiner and its output, either $\underline{p}(k)$ or $\underline{p}(k-1)$, reflects off mirrors M4, M3 and is expanded horizontally to illuminate the write side of SLM "L". A beam with intensity $\underline{x}(k)$ illuminates the read side of SLM "L" (which is read in reflection), to form outerproduct $\underline{p}(k)\underline{x}^T(k)$ or $\underline{p}(k-1)\underline{x}^T(k)$ for a 1-D array of neurons. At the first phase, $\underline{p}(k)\underline{x}^T(k)$ is added to the storage SLM "B". Then $\underline{p}(k-1)\underline{x}^T(k)$ is applied to "B", which is operated in subtraction mode during the second phase. These two steps calculate the potential difference and update the weights stored in "B".

During retrieval phase, partial input is applied to external input "I" and is then passed through threshold device "D", rotation optics "J", mirror M5, M1 and beam splitter BS1 to position "A" to perform vector-matrix computation of potential. Part of the iterated feedback signal $\underline{x}(k)$ reflects off BS1, M2 and is enabled by shutter S1 to store in 1-D storage SLM "N", which is used to form the outerproduct during the learning phase. Mirrors M1 and M5 are used, as shown, to implement feedback within a single layer network. For a multilayer network M1 and M5 can be removed (or replaced with beamsplitters) to send outputs to and receive signals from other layers.

5. CONCLUSIONS

This PDL provides a number of interesting features along with a moderate implementation complexity. It is a general technique that can be applied to different neuron types and different network models. Our simulations indicate that it learns correctly in a variety of networks. We also described an unlearning technique for the case of a fully connected network used as an associative memory, which does not require any sign reversal of the learning gain or any global access to the weights. Applications of PDL include low level processing such as extraction of features.

References

[1] D. O. Hebb, "Organization of behavior", *John Wiley & sons* press, (1949)

[2] Chr. von der Malsburg, "Self-organization of orientation sensitive cells in the striate cortex", *Kybernetik*, 14, 85-100 (1973)

[3] S. Grossberg, "Adaptive pattern classification and universal recoding: I. parallel development and coding of neural feature detectors", *Biol. Cybernetics*, 23, 121-134 (1976)

[4] D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning", *Cognitive Science*, 9, 75-112 (1985)

[5] K. Fukushima, "Cognitron: A self-organizing multilayered neural network", *Biol. Cybernetics*, 20, 121-136 (1975)

[6] J. J. Hopfield, D. I. Feinstein and R. G. Palmer, "Unlearning has a stabilizing effect in collective memories", *Nature*, 304, 158-159, July (1983)

[7] G. Chauvet, "Habituation rules for a theory of the cerebellar cortex", *Biol. Cybernetics*, 55, 201-209 (1986)

[8] Bart Kosko, "Differential Hebbian Learning", *Proc. of conf. on Neural Networks for Computering*, Snowbird, Utah, 277-282 (1986)

[9] A. H. Klopf, "A drive-reinforcement model of single neuron function: an alternative to the Hebbian neuronal model", *Proc. of conf. on Neural Networks for Computering*, Snowbird, Utah, 265-269 (1986)

[10] S-I. Amari, "Learning patterns and pattern sequences by self-organizing nets of threshold elements", *IEEE trans. on Computers*, C-21(11), 1197-1206 (1972)

[11] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational ability", *Proc. Natl. Acad. Sci. USA*, 79, 2554-2558 (1982)

[12] N. H. Farhat, D. Psaltis, A. Prata and E. Paek, "Optical implementation of the Hopfield model", *Appl. Optics*, 24, 1469-1475 (1985)

[13] C. T. Chen, Linear System Theory and Design, 57-59, Holt, Rinehart and Winston, 1984

[14] A. D. Fisher et. al., "Implementation of adaptive associative optical computing elements", *Proc. SPIE*, 625, 196-204 (1986)

Implementation Considerations of a Subtracting Incoherent Optical Neuron

C. H. Wang and B. K. Jenkins

Signal and Image Processing Institute, Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089-0272

ABSTRACT

The incoherent optical neuron (ION) subtracts inhibitory inputs from excitatory inputs optically by utilizing separate device responses. Those factors that affect the operation of the ION are discussed here, such as nonlinearity of the inhibitory element, input noise, device noise, system noise and crosstalk. A computer simulation of these effects is performed on a version of Grossberg's on-center off-surround competitive neural network.

1 Introduction

The need to process positive and negative signals optically in optical neural network has been pursued in the past few years. Existing techniques such as intensity bias [1] or weight bias method suffer from input dependent bias or thresholds that must vary from neuron to neuron. A technique described by Te Kolste and Guest [2] eliminites most of these drawbacks in the special case of fully connected networks.

The *incoherent optical neuron* (ION) [3, 4] model uses separate device responses for inhibitory and excitatory inputs. This is modeled after the biological neuron that processes the excitatory and inhibitory signals by different mechanisms (e.g. chemical-selected receptors and ion-selected gate channels) [5, 6, 7, 8]. By using this architecture, we can realize general optical neuron units with thresholding.

The ION comprises two elements: an inhibitory (I) element and a nonlinear output (N) element. The inhibitory element provides inversion of the sum of the inhibitory signals; the nonlinear element operates on the excitatory signals, the inhibitory element output, and an optical bias to produce the output. The inhibitory element is linear; the nonlinear threshold of the neuron is provided entirely by the output nonlinear device. Fig. 1(a) and 1(c) shows the characteristic curve of the I and N elements respectively. The structure of the ION model is illustrated in Fig. 1(d). The input/output relationships for the normalized I and N elements respectively, are given by:

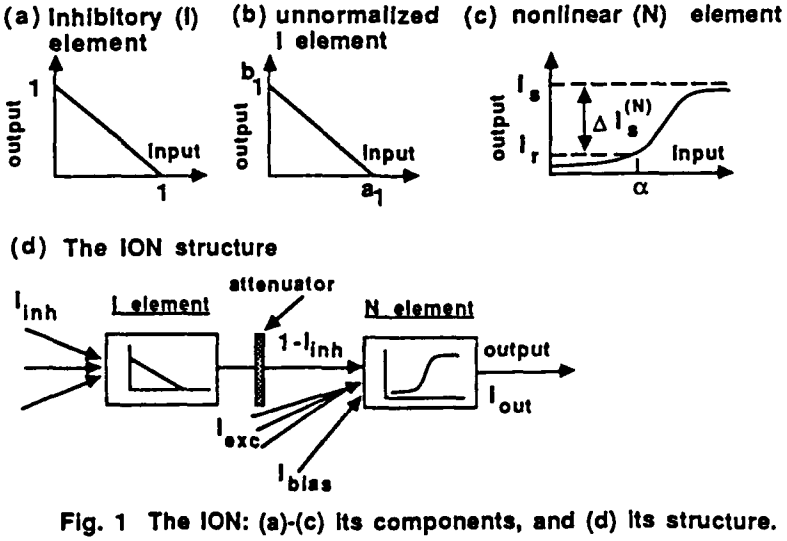


Fig. 1 The ION: (a)-(c) its components, and (d) its structure.

* Proc. IEEE International Conference on Neural Networks, San Diego, CA, July 1988. vol II, 403-410

$$I_{out}^{(I)} = \hat{I}_{inh} = 1 - I_{inh} \tag{1}$$

$$I_{out}^{(N)} = \psi(\hat{I}_{inh} + I_{exc} + I_{bias} - \alpha) \tag{2}$$

where I_{inh} and I_{exc} represent the total inhibitory and excitatory inputs, I_{bias} is the bias term for the N element, which can be varied to change the threshold, and α is the offset of the characteristic curve of the N element. $\psi(\cdot)$ represents the output nonlinear function of the N element. If we choose I_{bias} to be $\alpha - 1$, the output of the N element is

$$I_{out}^{(N)} = \psi(I_{exc} - I_{inh}) \tag{3}$$

which is the desired subtraction. In general, the I element will not be normalized (Fig 1(b)), in which case the offset and slope of its response can be adjusted using I_{bias} and an attenuating element (ND filter), respectively. The unnormalized I element must have gain greater than or equal to 1. A positive threshold (θ) can be implemented by lowering the bias term by the same amount θ. Similarly, a negative threshold is realized by increasing the bias term by θ.

The ION model can be implemented using separate devices for the I and N elements (heterogeneous case), or by using a single device with a nonmonotonic response to implement both elements (homogeneous case). Possible devices include bistable optical arrays [9, 10, 11, 12] and SLMs such as liquid crystal light valves (LCLV) [13]. A single Huges liquid crystal light valve can be used to implement both elements (Fig. 2).

Several factors that affect the realization of a neural network based on the ION concept, are examined here. These include deviation from linearity within the inhibitory element, residual noise of the optical device, input noise, drift of the operation point of the device, and system noise. A noise model for the ION is proposed and a computer simulation of these effects on a version of Grossberg's on-center off-surround type network [14] is performed.

2 Factors that Affect the ION Operation

2.1 Nonlinearity in I Element

In order to perform subtraction correctly, we need a linear I element. Fig. 2 shows the typical input output characteristic curve of the LCLV [15], which is nonlinear in the inversion region. In this region, the characteristic curve can be modeled as

$$I_{out}^{(I)} = 1 - I_{inh} - E_r(I_{inh}) \tag{4}$$

where $E_r(I_{inh})$ denotes the error term, which can be treated as an input dependent deterministic noise. If the transfer curve is time varying, then it can be treated as temporally correlated random noise.

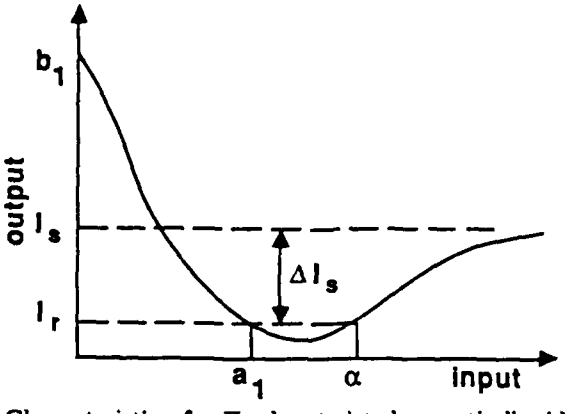


Fig. 2 Characteristic of a Hughes twisted-nematic liquid crystal light valve. The negative slope region can implement the I element, and the positive slope region, with appropriate input optical bias, can implement the N element.

2.2 Noise

Here we use "noise" to mean "any undesired signals", including perturbation of the operating point of the device, non-uniformity of the device, variation in operating characteristics from device to device due to production variation, environmental effects etc. Some of these effects are global (they affect all neuron units on a device identically), others are localized (each neuron unit behaves differently; the noise on neighboring neuron units on a device may be independent or correlated, depending on the source of the noise). Both temporal and spatial characteristics of the noise need to be included. The effect of noise on an additive lateral inhibitory network was discussed by Stirk, Rovnyak and Athale [16]. Here, we construct a noise model for the ION by considering the origin and impact of the noise sources.

The possible noise sources in the ION model can be classified into four categories: input noise, device noise, system noise and coupling noise. The input noise includes environmental background noise, residual output of the optical devices etc. Essentially, they are not zero mean and vary slowly with time. The device noise is mainly caused by uncertainty in the device's characteristics, for example drift of the operating point and variation of gain, due to temperature or other effects. The system noise has global effect on all neuron units on an optical device and includes fluctuations in the optical source. Finally, the coupling noise (crosstalk) is due to poor isolation between the optical neuron units, crosstalk from the interconnection network, and imperfect learning. As noted in [16], alignment inaccuracies and imperfect focussing and collimating optics also cause localized crosstalk. Coupling noise is signal dependent.

2.3 Noise Model for the ION

Device Input Noise

Let the environmental background noise for the I and N elements be denoted by $N_b^{(I)}$ and $N_b^{(N)}$ respectively. The total residual output noise, caused by the optical devices to the input of an incoherent optical neuron, is $N_r = \sum_{j=1}^{N_{in}} W_{ij} I_r / N_{out}$, which is weight-dependent and varies slowly with time due to learning. W_{ij} is the interconnection strength from neuron j to neuron i. I_r is the residual output of the optical device (Fig. 1(c)). N_{in} and N_{out} denote the fan-in and fan-out of the optical neuron unit respectively. Perturbation of the weights can be treated as an input dependent noise source as $N_w = \sum \Delta W_{ij} \cdot x_j$, where each ΔW_{ij} is independent. For the interconnection network, imperfect learning of the weights, nonuniformity of the weights, residual weights after reprogramming and perturbation of the reference beam intensity will cause weight noise. Then the output of the ION for the case of normalized characteristics is

$$I_{out} = \psi\{[1 - (I_{inh} + N_b^{(I)} + N_r^{(I)} + N_w^{(I)})] + [I_{exc} + N_b^{(N)} + N_r^{(N)} + N_w^{(N)}] + (\alpha - 1) - \alpha\} \qquad (5)$$

If the background noise is space invariant and the I and N element have the same device area, the terms $N_b^{(I)}$ and $N_b^{(N)}$ will cancel out. The residual noise terms $N_r^{(I)}$ and $N_r^{(N)}$, and weight noise $N_w^{(I)}$ and $N_w^{(N)}$ generally do not cancel.

Device Noise

There are two possible noise sources in the I element, as illustrated in Fig. 3(a) and (b): shift (drift) and gain variation in the device characteristics, which are denoted as $N_d^{(I)}$ and $N_g^{(I)}$ respectively. For the output N element, the gain variation (Fig. 3(e)) only modifies the nonlinearity of the element N. If this gain variation is a slowly varying effect, it will have little effect on the dynamic behavior of the network; so for the N element we only consider the drift effect. Let's denote it by $N_d^{(N)}$. Two different drifts in the N element are possible, horizontal drift ($N_{dh}^{(N)}$) (Fig. 3(c)) and vertical drift ($N_{dv}^{(N)}$) (Fig 3(d)). The vertical drift of one neuron unit becomes an additive noise at the input of the next neuron unit, and so will be approximated by including it in the residual noise term above. The horizontal drift has the same effect as a perturbation in the bias term, denoted by N_{bp}.

If the gain variation is small, the output of the I element can be expressed as $(1 + N_g) - (1 + N_g) I_{inh}$, where N_g denotes the gain noise.

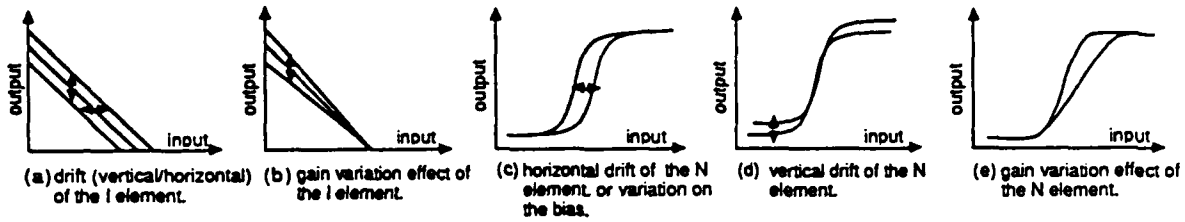


Fig. 3 Modeling the device noise of an incoherent optical neuron.

System Noise

System noise has a global effect on the ION. If it is caused by an uncertainty in the optical source, it causes a variation in the characteristic curve of the device and a perturbation in the bias term. In the case of an LCLV, a perturbation in the reading beam intensity produces a gain variation in the I element and a combination of gain variation and horizontal drift in the N element (or equivalently, essentially a re-scaling of its input axis). Gain variation of the I element was discussed above. The difference is that the device gain variation is local, ie. it varies from one neuron unit to another on a given device, while the variation in gain due to system noise is global.

The device noise and system noise can be modeled as

$$I_{out} = \psi\{(1 + N_g^{(I)})[1 + N_d^{(I)} - I_{inh}] + I_{exc} + N_{dh}^{(N)} + (\alpha - 1 + N_{bp}) - \alpha\} \tag{6}$$

Crosstalk

Crosstalk can be caused by the physical construction of the interconnection network (e.g., coupling between different holograms, diffraction in the detection (neuron unit inputs) plane, inaccurate alignment and focussing). It can also be caused by imperfect learning or reprogramming of the synaptic weights, where the perturbation of different weights are correlated. In general, crosstalk is signal dependent and varies from one neuron unit to another on a given device. *It can be modeled as an input noise to the* I and N elements. It is excluded from our current simulations because it is signal dependent.

Based on the above discussion, these noise terms can be grouped into additive (N_I^+) and multiplicative (N_I^*) noise of the I element and additive noise (N_N^+) of the output element N. The general noise model of the ION can be written as

$$I_{out} = \psi\{(1 + N_I^*)[1 - I_{inh} + N_I^+] + I_{exc} + N_N^+ - 1\} \tag{7}$$

where N_I^+ is the sum of the drift noise ($N_d^{(I)}$), background noise ($N_b^{(I)}$), residual noise ($N_r^{(I)}$), weight noise ($N_w^{(I)}$) and crosstalk noise ($N_c^{(I)}$); N_I^* is the gain noise of I element; and N_N^+ is the sum of the background ($N_b^{(N)}$) and residual input noise, horizontal shift noise ($N_{dh}^{(N)}$), weight noise and crosstalk noise of the N element, and bias noise (N_{bp}).

3 Computer Simulation

3.1 Compensation for the Nonlinearity of the I Element

To assess the effect of imperfect device responses for the I element, we have performed simulations on a variant of Grossberg's on-center off-surround competitive network [14] for edge detection (Fig. 4). The network contains 30 inner-product type neurons connected in a ring structure with input and lateral on-center off-surround connections. Fig. 5 shows several modeled nonlinear characteristic curves for the I element; these approximate the normalized response of a Hughes liquid-crystal light valve. An attenuator (neutral density filter) can be placed in front of the I element to reduce the overall gain (by effectively re-scaling the input axis) to bring it closer to the ideal response. Fig. 6 shows computer simulations of the network responses based on nonlinear curve #2 for different input attenuations and input levels. As shown here, the attenuation has a tolerance of approximately ±20%. For extremely nonlinear responses we expect an input bias and a limited region of

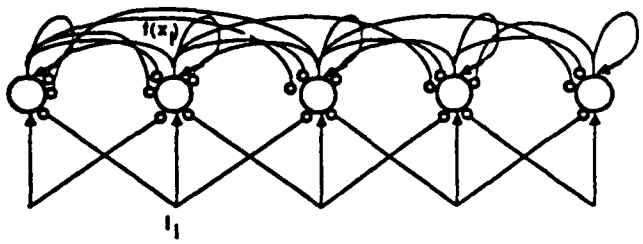
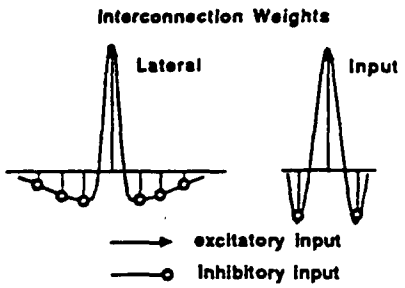


Fig. 4 On-center off-surround competitive neural network.



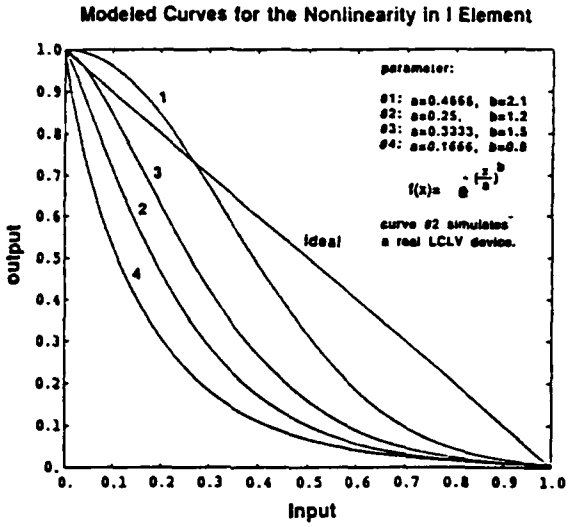


Fig. 5 Four curves used for simulating the effect of nonlinearities in the I element.

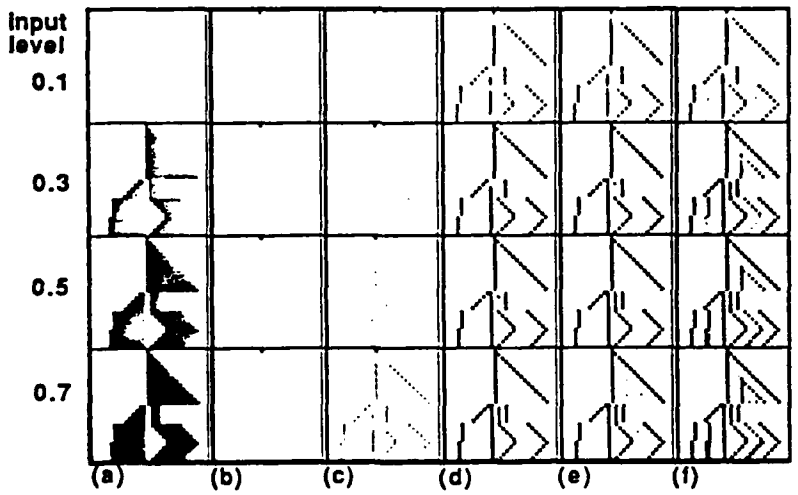


Fig. 6 Network responses for different attenuation factors, S_1, at the input to the nonlinear inhibitory element.
a) input patterns. b) $S_1 = 1.0$ (no attenuation), c) $S_1 = 1.5$, d) $S_1 = 2.5$, e) $S_1 = 3.0$, f) $S_1 = 3.5$. The ideal output is essentially identical to (d).

operation to provide a sufficiently linear response. In our simulations we used an attenuator but no input bias to the I element; the region of operation for these curves was seen to extend over most of the input range of the device [3,4].

3.2 Noise Effect

We use the same network to test effect of noise (of course the results are actually network dependent) to get an idea of the noise imunity and robustness to physical imperfections of the ION model. In the computer simulation, each of the three noise sources in Eq. (7) are assumed independently distributed Gaussian with zero mean. We define the maximum perturbation, p, of the noise source as twice the standard deviation, expressed as a percentage of the input signal level. A normalized mean square error (*nmse*) is used to measure the acceptability of the result. Although it is not a perfect measure, a *nmse* less than 0.1-0.15 generally looks acceptable for the network response for our input test pattern.

Fig. 7(a) shows the *nmse* vs. percentage of maximum noise perturbation for the input level of 0.7 and for noise that is correlated over different time periods T. The noise sources for each neuron are assumed independent and identically distributed (*iid*). Each noise source is temporally correlated with its previous values, as given by $N(t+1) = \sum_{i=1}^{T} h_i \cdot N(t+1-i)$. The correlation coefficients h_i decrease linearly with i (to $h_T = 0$). In Fig. 7(a), all three noise sources in our model are present and have the same variance. If the acceptance *nmse* criterion is 0.15, a perturbation of ±10% on each noise source yields an acceptable result in all cases. For $T = 50$, the *nmse* increases as the input level and noise variance increase as shown in Fig. 7(b). The network responses are shown in Fig. 8 for temporally correlated noise with perturbation of ±10%.

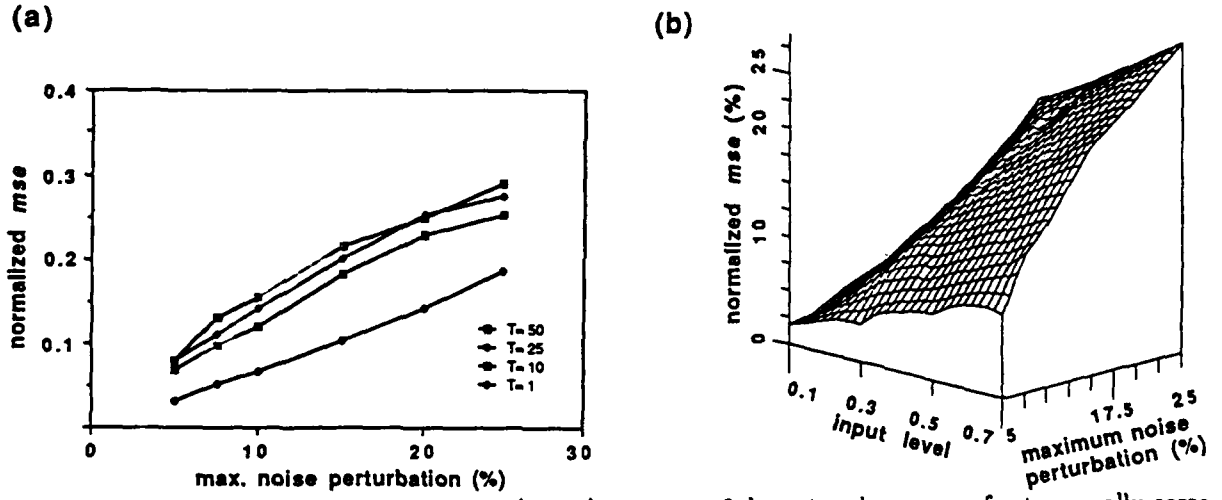


Fig. 7 Normalized mean square error (*nmse*) measure of the network response for temporally correlated noise. Three noise sources, N_I^+, N_I^*, and N_N^+, are simulated.

a) Normalized mean square error of the net output vs. maximum noise perturbation p for correlation periods (T) ranging from 1 to 50. The input level is 0.7.

b) Output *nmse* plot for different noise perturbation and input levels (T=50).

In some cases, the noise is spatially correlated. We simulated the network with spatially correlated noise. The spatial correlation is assumed to have a Gaussian profile. Fig 9(a) and (b) are the responses for a spatial correlation range of 5 and 13 respectively, while Fig 9(c) and (d) show the responses for spatially *and* temporally correlated noise.

Drift of the device characteristic is a global effect. Fig 10 simulates slowly varying and quickly varying drift on this network. Fig. 11 shows the effect of local gain variation that is spatially correlated. A ±25% perturbation in drift is apparently acceptable, and a $\pm 15 - 20\%$ perturbation in gain is acceptable.

4 Discussions and Conclusions

We have summarized sources of noise for the ION and proposed a noise model. From the result of the computer simulation, it seems that the example network performs much better for quickly varying (ie. temporally

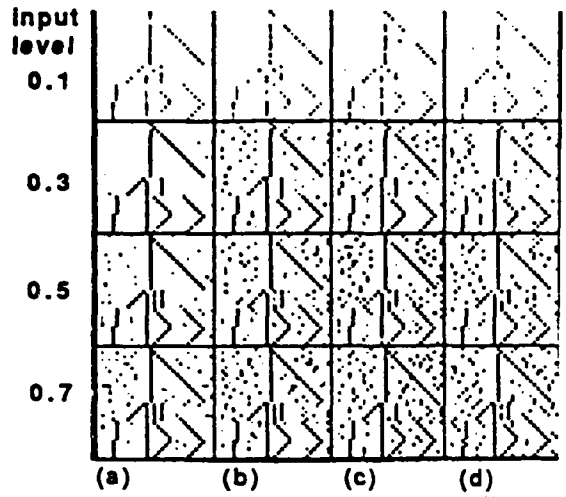


Fig. 8 The network response for temporally correlated noise sources. Three noise are simulated with the same maximum noise perturbation of ±10%. T is the correlation period of the noise and *nmse* is the normalized mean square error of the output. Here, the given value of *nmse* corresponds to the maximum input level case.

a) T=1, *nmse* = 0.07, b) T=10, *nmse* = 0.12, c) T=25, *nmse* = 0.14, d) T=50, *nmse* = 0.16.

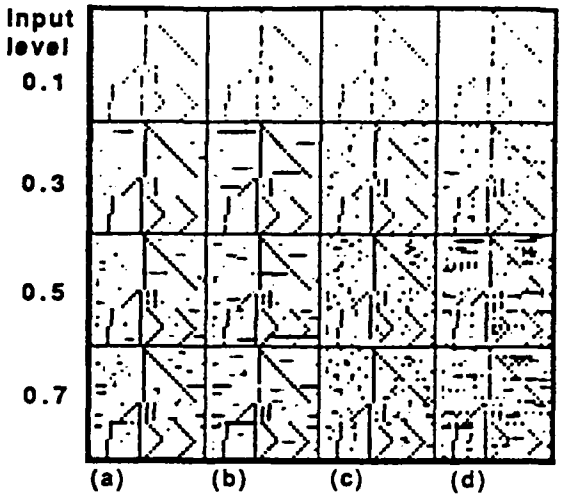


Fig. 9 Simulation result of spatially and temporally correlated noise. *sc* is the spatial correlation range. Three noise sources are simulated simulateneously with p = ±10% and correlation period T.

a)-b) only spatially correlated noise with a) *sc* = 3, *nmse* = 0.08, b) *sc* = 13, *nmse* = 0.13, c)-d) are the result of spatially and temporally correlated noise (T=25) with c) *sc* = 3, *nmse* = 0.14, d) *sc* = 13, *nmse* = 0.18.

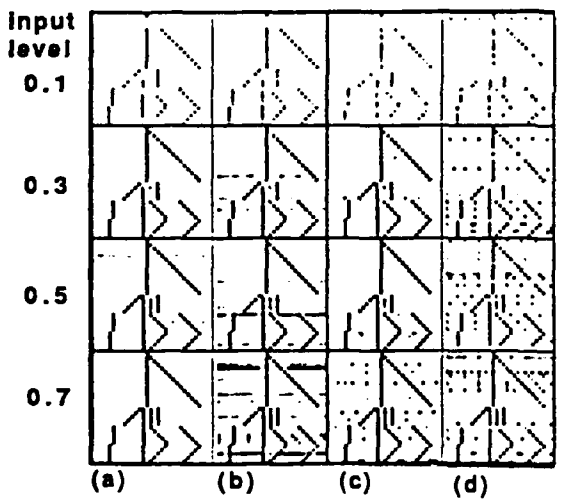


Fig. 10 Effect of device drift in the ION. p is the maximum perturbation of the noise source. T is the temporal correlation period of the drift. The drift effect is uniform over all neuron units. a) & b) simulate high frequency drift (T=1) with a) p = ±10%, *nmse* = 0.02, b) p = ±25%, *nmse* = 0.09, c)-d) are the low frequency drift (T=50) with c) p = ±10%, *nmse* = 0.07, d) p = ±25%, *nmse* = 0.11.

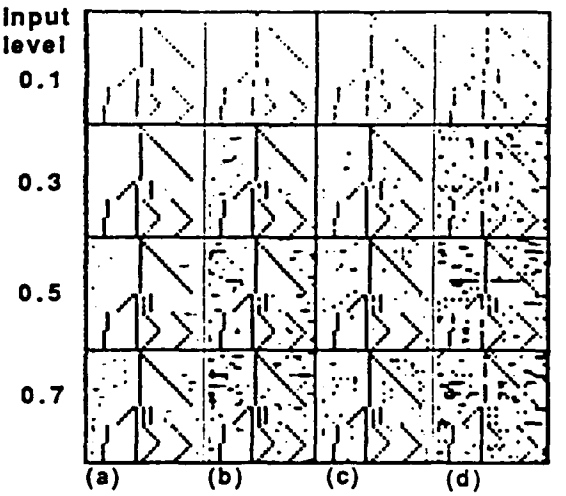


Fig. 11 The gain variation effect of the ION. *sc* and p are the correlation range and maximum perturbation percentage of the noise source. a)-b) high frequency gain variation with a) T=1, *sc* = 3, , p = ±10%, *nmse* = 0.04, b) T=1, *sc* = 3, p = ±25%, *nmse* = 0.11. c)-d) low frequency variation with c) T=25, *sc* = 9, p = ±10%, *nmse* = 0.09, d) T=25, *sc* = 9, p = ±25%, *nmse* = 0.18.

uncorrelated) noise than for temporally correlated (more slowly varying) noise. Due to the static input pattern and the competitive nature of the network, once the noise term has survived a number of iterations, then it will continue to get stronger and will not die out. We speculate that if the input to the network is time varying, then the slowly varying noise source is effectively an offset response of the network and might be adaptively overcome by the network, while the quickly varying noise interacts with the input patterns and is more difficult to compensate.

For noise that is correlated, we have found that the qualitative effect of each of the three noise sources (additive inhibitory, multiplicative inhibitory, and additive excitatory) on the output of the net is essentially the same. Since one of the noise terms, N_N^+, is the same for a conventional neuron implementation as for the ION, it appears that an ION implementation is not significantly different from a conventional neuron implementation in terms of immunity to noise and device imperfections, for a given technology. We also see that the output is affected primarily by the *variance* of the noise and by the degree of *spatial and temporal correlation*, but apparently *not* by the source of the noise. We conjecture that this result is not peculiar to the ION model, but is true of other neuron implementations as well.

References

[1] A. F. Gmitro and G. R. Gindi, "Optical neurocomputer for implementation of the Marr-Poggio stereo algorithm" *Proc. of IEEE First Int. Conf. on Neural Network*, San Diego, vol III, pp599-606, 1987.

[2] R. D. Te Kolste and C. C. Guest, "Optical competitive neural network with optical feedback", *Proc. of IEEE First Int. Conf. on Neural Network*, San Diego, vol III, pp 625-629, 1987.

[3] B. K. Jenkins and C. H. Wang, "Requirements for an incoherent optical neuron that subtracts", *J. of Opt. Soci. Am. (A)*, vol 4, No. 13, 127A, paper PD6, Dec. 1987.

[4] B. K. Jenkins and C. H. Wang, "Model for an incoherent optical neuron that subtracts", submitted to *Optics Letters*, 1988.

[5] M. Wang and A. Freeman, "Neural function", *Little, Brown and Company* press, 1987.

[6] C. F. Stevens, "The neuron", *Scientic American*, pp55-65, Aug. 1979.

[7] G. M. Shepherd, "Microcircuits in the nervous system", *Scientic American*, pp93-103, Feb. 1978.

[8] R. D. Keynes, "Ion channels in the nerve-cell membrane", *Scientic American*, pp126-135, March 1979.

[9] A. C. Walker, "Application of bistable optical logic gate arrays to all-optical digital parallel processing", *Applied Optics*, vol. 25, No. 10, pp1578-1585, May 1986.

[10] S. D. Smith, "Optical bistability, photonic logic, and optical computation", *Applied Optics*, vol. 25, No. 10, pp1550-1564, May 1986.

[11] S. D. Smith, A. C. Walker, et. al., "Cascadable digital optical logic circuit elements in the visible and infrared: demonstration of some first all-optical circuits", *Applied Optics*, vol. 25, No. 10, pp1586-1593, May 1986.

[12] F. A. P. Tooley, S. D. Smith, and C. T. Seaton, "High gain signal amplification in an InSb transphasor at 77 K", *Appl. Phys. Lett.*, vol 43, pp9-, 1983.

[13] W. P. Bleha et. al., "Application of the liquid crystal light valve to real-time optical data processing", *Optical Engineering*, vol 17, No. 4, pp371-384, July/Aug., 1978.

[14] S. A. Ellias and S. Grossberg, "Pattern formation, contrast control and oscillations in short term memory of shunting on-center off-surround networks", *Biol. Cybernetics*, vol 20, pp69-98, 1975.

[15] B. K. Jenkins, A. A. Sawchuk, T. C. Strand et. al., "Sequential optical logic implementation", *Appl. Optics*, vol 23, No. 19, pp3455-3464, 1 Oct. 1984.

[16] C. W. Stirk, S. M. Rovnyak and R. A. Athale, "Effects of system noise on an optical implementation of an additive lateral inhibitory network", *IEEE first International Conf. on Neural Network*, San Diego, vol III, pp615-624, 1987.

I hereby submit a communication related to topic number 5

1. nonlinear phenomena
2. nonlinear and electro-optical components
3. SLM
4. interconnections
5. optical processors
6. hybrid processors
7. neural processors
8. massively parallel architectures
9. symbolic substitution
10. applications
11. physical limits

Enclosed is a summary in no more than 2 pages and a 25-word abstract :

Title SUPERPOSITION IN OPTICAL COMPUTING

	first author	coauthors
name	B. KEITH JENKINS	C. LEE GILES
company	USC, MC-0272	AFOSR/NE
address	LOS ANGELES, CA 90089-0272 USA	BOLLING AFB, D.C. 20332-6448

Standard audiovisual aids available are :
overhead and 5 cm x 5 cm slides projectors.

Note that the final manuscript is due september 2, 1988.

Do not forget to fill in and mail your registration and accommodation forms.

signature and date :

B Keith Jenkins

3/11/88

Superposition in Optical Computing

B. Keith Jenkins
Signal and Image Processing Institute
University of Southern California, Los Angeles, California 90089-0272

and

C. Lee Giles
Air Force Office of Scientific Research/NE, Bolling AFB, D.C. 20332-6448

ABSTRACT

The property of superposition in optics is not present in electronics, and can be utilized in the implementation of optical interconnections, shared memory, and gates.

Superposition in Optical Computing

B. Keith Jenkins
Signal and Image Processing Institute
University of Southern California, Los Angeles, California 90089-0272

and

C. Lee Giles
Air Force Office of Scientific Research/NE, Bolling AFB, D.C. 20332-6448

SUMMARY

Fundamental differences in the properties of electrons and photons provide for expected differences in computational systems based on these elements. Some, such as the relative ease with which optics can implement regular, massively parallel interconnections are well known. In this paper we examine how the property of superposition of optical signals in a linear medium can be exploited in building an optical or hybrid optical/electronic computer. This property enables many optical signals to pass through the same point in space at the same time without causing mutual interference or crosstalk. Since electrons do not have this property, this may shed more light on the role that optics could play in computing. We will separately consider the use of this property in interconnections, memory, and gates.

Interconnections. A technique for implementing hybrid space-variant/space-invariant optical interconnections from one 2-D array to another (or within the same array) has been described [1]. It utilizes two holograms in succession, where the first hologram serves as an array of facets that each address facets in the second hologram. The superposition property allows many optical beams to pass through a facet in the second hologram, permitting many input nodes to effectively share the same routing "wire" to output nodes. This decreases the complexity (space-bandwidth product) of both holograms.

Using this as a model for interconnections in parallel computing, a comparison can be made between the complexity of these optical interconnections with those of electronic VLSI for various interconnection networks [2]. It is found that in general the optical interconnections have an equal or lower space complexity than electronic interconnections, with the difference becoming more pronounced as the connectivity increases. Also, a slight variation in a given network can further reduce the space complexity in the optics case. An example is a hypercube ($O(n^2)$ in VLSI, $O(n \log n)$ in optics) [2] vs. a 2-D cellular hypercube (twice as many connections, at least $O(n^2)$ in VLSI, yet $O(n)$ in optics).

Shared memory. The same superposition principle can be applied to memory cells, where many optical beams can read the same memory location simultaneously. This concept is useful in building a parallel shared memory machine.

For this concept, we consider abstract models of parallel computation based on shared memories [3]. The reason for this approach is to abstract out inherent limitations of electronic technology (such as limited interconnection capability); in designing an architecture one would adapt the abstract model to the limitations of optical systems. In Fig. 1 we see a typical shared memory model where individual processing elements (PE's) have variable simultaneous access to an individual memory cell.

In general, these shared memory are not physically realizable because of actual fan-in limitations. As an electronic example, the ultracomputer [4] is an architectural manifestation of a shared memory model, and uses a hardwired Omega network between the PE's and memories; it simulates the shared memory model with a time penalty of $O(\log^2 n)$.

Optical systems could in principle be used to implement this parallel memory read capability. As a simple example, a single 1-bit memory cell can be represented by one pixel of a 1-D or 2-D array; the bit could be represented by the state (opaque or transparent) of the memory cell. Many optical beams can simultaneously read the contents of this memory cell without contention, by the superposition property. A system based on this concept includes an array of memory cells,

an interconnection network, and an array of PE's. The interconnection network is needed between the PE's and the memory, and must allow any PE to communicate with any memory cell, preferably in one step, and with no contention. A regular crossbar is not sufficient for this because fan-in to a given memory cell must be allowed. Optical systems can potentially implement crossbars that also allow this fan-in (e.g., some of the systems described in [5]).

Gates. Since the superposition property of optics only applies in linear media, it cannot in general be used for gates, which are (by definition) nonlinear. However, for important special cases superposition can allow many optical gates to be replaced with one optical switch.

Consider an aperture whose state (opaque or transparent) is controlled by an optical beam, with again many optical beams being able to read its state simultaneously. Here the aperture is being used as a switch or relay, and the control beam opens or closes the switch. If b represents the control beam and a_i the signal beams, this in effect computes $b \cdot a_i$ or $\overline{b} \cdot a_i$, depending on which state of b closes the switch, where $\cdot$ denotes the AND operation (Fig. 2).

Using this concept, a set of gates with a common input in an SIMD machine can be replaced with one optical switch or "superimposed gate". It also obviates the need for broadcasting the instructions to all PE's; instead, a fan-in of all signals to a common control switch is performed.

These superimposed gates are not true 3-terminal devices, since the a_i inputs are not regenerated. As a result, a design constraint must be adhered to: these a_i signals should not go through too many superimposed gates in succession without being regenerated by a conventional gate. Note, however the following features. The total switching energy required for a given processing operation is reduced, because N gates are replaced with one superimposed gate. This is important because it is likely that the total switching energy will ultimately be the limiting factor on the switching speed and number of gates in an optical computer. Also, it permits an increase in computing speed since some of the gates are effectively passive, and reduces requirements on the device used to implement the optical gates.

In summary, architectures for optical computing must incorporate the capabilities of optics as opposed to electronics. A familiar but important inherent difference lies in the superposition property of optical beams, which can be expoited in opitcal interconnections, gates, and memory.

REFERENCES

[1] B.K. Jenkins, et al., *Applied Optics*, Vol. 23, No. 19, pp. 3465-3474, 1984.

[2] C.L. Giles and B.K. Jenkins, *Proc. Twentieth Annual Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, Calif., IEEE Cat. No. 87CH2461-2, 513-517 (1986).

[3] B.K. Jenkins and C.L. Giles, *Proc. SPIE*, **625** 22-29 (1986)

[4] J.T. Schwartz, *A.C.M. Trans on Prog. Lang. and Sys.* **2**, No. 4, 484-521 (1980).

[5] A.A. Sawchuk, B.K. Jenkins, C.S. Raghavendra, and A. Varma, *Computer* **20**, No. 6, 50-60 (1987).

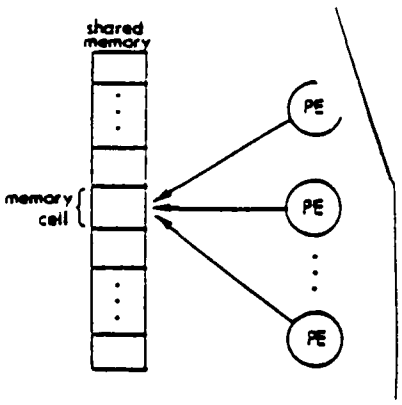


Fig. 1. Conceptual diagram of shared memory models.

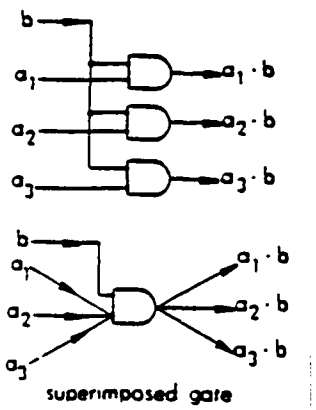


Fig. 2. One optical relay or superimposed gate versus individual gates with a common input.

Proc. First ICNN, June 1987, San Diego, CA

A Novel Approach to Image Restoration Based on a Neural Network[1]

Y. T. Zhou, R. Chellappa and B. K. Jenkins

Signal and Image Processing Institute
Department of EE-Systems
University of Southern California

Abstract

A novel approach for restoration of gray level images degraded by a known shift invariant blur function and additive noise is presented using a neural computational model. A neural network model is employed to represent an image whose gray level function is the simple sum of the neuron state variables. The restoration procedure consists of two stages: estimation of the parameters of the neural network model and reconstruction of images. During the first stage, image noise is suppressed and the parameters are estimated. The restoration is then carried out iteratively in the second stage by using a dynamic algorithm to minimize the energy function of an appropriate neural network. Owing to the model's fault-tolerant nature and computation capability, a high quality image is obtained using this approach. A practical algorithm with reduced computational complexity is also presented. Several computer simulation examples involving synthetic and real images are given to illustrate the usefulness of our method.

1 Introduction

Image restoration is an important problem in early vision processing to recover an ideal high quality image from a degraded recording. Restoration techniques are applied to remove (1) system degradations such as blur due to optical system aberrations, atmospheric turbulence, motion and diffraction; and (2) statistical degradations due to noise. Over the last 20 years, various methods such as the inverse filter, Wiener filter, Kalman filter, SVD pseudoinverse and many other model based approaches, have been proposed for image restoration. One of the major drawbacks of most of the image restoration algorithms is the computational complexity, so much so that many simplifing assumptions have been made to obtain computationally feasible algorithms. An artificial neural network system that can perform extremely rapid parallel computation seems to be very attractive for image processing applications; preliminary investigations to various problems such as pattern recognition and image processing are very promising [1].

In this paper, we use a neural network model containing redundant neurons to restore gray level images degraded by a known shift invariant blur function and noise. It is based on the model described in [2] [3] using a simple sum number representation [4]. The image gray levels are represented by the simple sum of the neuron state variables which take binary values of 1 or 0. The observed image is degraded by a shift invariant function and noise. The restoration procedure consists of two stages: estimation of the parameters of the neural network model, and reconstruction of images. During the first stage, the image noise is suppressed and the parameters are estimated. The restoration is then carried out by using a dynamic iterative algorithm to minimize the energy function of the neural network. Owing to the model's fault-tolerant nature and computation capability, a high quality image is obtained using our approach. We illustrate the usefulness of this approach by using both synthetic and real images degraded by a known shift-invariant blur function with or without noise.

[1]This research work is partially supported by the AFOSR Contract No. F-49620-87-C-0007.

2 Image Representation Using a Neural Network

We use a neural network containing redundant neurons for representing the image gray levels. The model consists of $L^2 \times M$ mutually interconnected neurons, where L is the size of image and M is the maximum value of the gray level function. Let $V = \{v_{i,k}, 1 \leq i \leq L^2, 1 \leq k \leq M\}$ be a binary state set of the neural network with $v_{i,k}$ (1 for firing and 0 for resting) denoting the state of the (i,k)th neuron. Let $T_{i,k,j,l}$ denote the strength (possibly negative) of the interconnection between neuron (i,k) and neuron (j,l). We require symmetry

$$T_{i,k,j,l} = T_{j,l,i,k} \quad for \quad 1 \leq i,j \leq L^2 \ and \ 1 \leq l,k \leq M$$

We also insist that the neurons have self-feedback, i.e. $T_{i,k,i,k} \neq 0$. In this model, each neuron (i,k) randomly and asynchronously receives inputs $\sum T_{i,k,j,l} v_{j,l}$ from all neurons and a bias input $I_{i,k}$

$$u_{i,k} = \sum_{j}^{L^2} \sum_{l}^{M} T_{i,k,j,l} v_{j,l} + I_{i,k} \tag{1}$$

Each $u_{i,k}$ is fed back to corresponding neurons after thresholding

$$v_{i,k} = g(u_{i,k}) \tag{2}$$

where $g(x)$ is a nonlinear function whose form can be taken as

$$g(x) = \begin{cases} 1 & if \ x \geq 0 \\ 0 & if \ x < 0. \end{cases} \tag{3}$$

In this model, the state of each neuron is updated by using the latest information about other neurons.

The image is described by a finite set of gray level functions $\{x(i,j), 1 \leq i,j \leq L\}$ with $x(i,j)$ (positive integer number) denoting the gray level of the cell (i,j). The image gray level function can be represented by a simple sum of the neuron state variables as

$$x(i,j) = \sum_{k=1}^{M} v_{m,k} \tag{4}$$

where $m = i \times L + j$. Here the gray level functions have degenerate representations. Use of this redundant number representation scheme yields advantages such as fault-tolerance and convergence to the solution [4].

If we scan the 2-D image by rows and stack them as a long vector, then the degraded image vector can be written as

$$\underline{Y} = H\underline{X} + \underline{N} \tag{5}$$

where H is the $L^2 \times L^2$ point spread function (or blur) matrix, and $\underline{X}$, $\underline{Y}$ and $\underline{N}$ are the $L^2 \times 1$ long original, degraded and noise vectors, respectively. This is similar to the simultaneous equations solution of [4], but differs in that (5) includes a noise term.

The shift-invariant blur function can be written as a convolution over a small window, for instance, it takes the form

$$h(k,l) = \begin{cases} \frac{1}{2} & if \ k = 0, \ l = 0 \\ \frac{1}{16} & if \ |k|, \ |l| \leq 1, \ (k,l) \neq (0,0) \end{cases} \tag{6}$$

accordingly, the "blur matrix" H will be a block Toeplitz or block circulant matrix (if the image has periodic boundaries).

3 Estimation of Model Parameters

The neural model parameters, the interconnection strengths and the bias inputs, can be determined in

terms of the energy function of the neural network. As defined in [2], the energy function of the neural network can be written as

$$E = -\frac{1}{2}\sum_{i=1}^{L^2}\sum_{j=1}^{L^2}\sum_{k=1}^{M}\sum_{l=1}^{M} T_{i,k;j,l}\, v_{i,k}\, v_{j,l} + \sum_{i=1}^{L^2}\sum_{k=1}^{M} I_{i,k}\, v_{i,k} \tag{7}$$

In order to use the spontaneous energy-minimization process of the neural network, we reformulate our restoration problem as one of minimizing an energy function defined as

$$E = \frac{1}{2}\|\underline{Y} - H\underline{X}\|^2 \tag{8}$$

where $\|\underline{Z}\|$ is the L_2 norm of $\underline{Z}$. By comparing the terms in the expansion of (8) with the corresponding terms in (7), we can determine the interconnection strengths and the bias inputs as

$$T_{i,k;j,l} = -\sum_{p=1}^{L^2} h_{p,i}\, h_{p,j} \tag{9}$$

and

$$I_{i,k} = \sum_{p=1}^{L^2} y_p\, h_{p,i}. \tag{10}$$

From (9), one can see that the interconnection strengths are determined by the shift-invariant blur function. Hence, $T_{i,k;j,l}$ can be computed without error provided the blur function is known. However, the bias inputs are functions of observation, the degraded image. If the image is degraded by shift-invariant blur function only, then $I_{i,k}$ can be estimated perfectly. Otherwise, the degraded image needs to be preprocessed to suppress the noise if the signal to noise ratio (SNR), defined by

$$SNR = 10\,\log_{10}\frac{\sigma_s^2}{\sigma_n^2} \tag{11}$$

where σ_s^2 and σ_n^2 are variances of signal and noise, respectively, is low.

4 Restoration

Restoration is carried out by the neuron evaluation and image construction procedure. Once the parameters $T_{i,k;j,l}$ and $I_{i,k}$ are obtained using (9) and (10), each neuron can randomly and asynchronously evaluate its state and readjust accordingly using (1) and (2). When one quasi-minimum energy point is reached, the image can be constructed by (4).

However, this neural network has self-feedback, i.e. $T_{i,k;i,k} \neq 0$, as a result of a transition the energy function E does not decrease monotonically. This is explained as follows. Define the state change $\Delta v_{i,k}$ of neuron (i,k) and energy change ΔE as

$$\Delta v_{i,k} = v_{i,k}^{new} - v_{i,k}^{old} \quad and \quad \Delta E = E^{new} - E^{old}$$

Consider the energy function

$$E = -\frac{1}{2}\sum_{i=1}^{L^2}\sum_{j=1}^{L^2}\sum_{k=1}^{M}\sum_{l=1}^{M} T_{i,k;j,l}\, v_{i,k}\, v_{j,l} - \sum_{i=1}^{L^2}\sum_{k=1}^{M} I_{i,k}\, v_{i,k}, \tag{12}$$

Then the change ΔE due to a change $\Delta v_{i,k}$ is given by

$$\Delta E = -(\sum_{j=1}^{L^2}\sum_{l=1}^{M} T_{i,k;j,l}\, v_{j,l} + I_{i,k})\, \Delta v_{i,k} - \frac{1}{2}\, T_{i,k;i,k}\, (\Delta v_{i,k})^2 \qquad (13)$$

which is not always negative. For instance, if

$$v_{i,k}^{old} = 0, \quad u_{i,k} = \sum_{j=1}^{L^2}\sum_{l=1}^{M} T_{i,k;j,l}\, v_{j,l} + I_{i,k} > 0,$$

and the threshold function is as in (3), then $v_{i,k}^{new} = 1$ and $\Delta v_{i,k} > 0$. Thus, the first term in (13) is negative. But

$$T_{i,k;i,k} = -\sum_{p=1}^{L^2} h_{p,i}^2 < 0.$$

leading to

$$-\frac{1}{2}\, T_{i,k;i,k}\, (\Delta v_{i,k})^2 > 0.$$

When the first term is less than the second term in (13), then $\Delta E > 0$ (we have observed this in our experiments).

Thus, depending on whether convergence to a local minimal or a global minimal is desired, we can design a deterministic or stochastic decision rule. The deterministic rule is to take a new state $v_{i,k}^{new}$ of neuron (i,k) if the energy change ΔE due to state change $\Delta v_{i,k}$ is less than zero. If ΔE due to state change is > 0, no state change is affected. We have also designed a stochastic rule similar to the one used in simulated annealing techniques [5] [6]. The details of this stochastic scheme are given as follows:

Define a Boltzmann distribution by

$$\frac{p_{new}}{p_{old}} = e^{\frac{-\Delta E}{T}}$$

where p_{new} and p_{old} are the probabilities of the new and old global state, respectively, ΔE is the energy change and T is the parameter which acts like temperature. A new state $v_{i,k}^{new}$ is taken if

$$\frac{p_{new}}{p_{old}} > 1, \quad or\ if\ \ \frac{p_{new}}{p_{old}} \leq 1\ \ but\ \ \frac{p_{new}}{p_{old}} > \xi$$

where ξ is a random number uniformly distributed in the interval [0,1].

The restoration algorithm can then be summarized as

1. Set the initial state of the neurons.
2. Update the state of all neurons randomly and asynchronously according to the decision rule.
3. Check the energy function; if energy does not change anymore, go to next step; otherwise, go back to step 2.
4. Construct an image using (4).

5 A Practical Algorithm

The algorithm described above is difficult to simulate on a conventional computer due to high computational complexity even for images of reasonable size. For instance, if we have an $L \times L$ image with M gray levels, then L^2M neurons and $\frac{1}{2}L^4M^2$ interconnections are required and L^4M^2 additions and multiplications are needed at each iteration. Therefore, the space and time complexities are $O(L^4M^2)$ and $O(L^4M^2K)$, respectively, where K $O(10)$–$O(100)$ is the number of iterations. When $L = 256$ and

$M = 256$, the space and time complexities will be $O(10^{14})$ and $O(10^{15})$–$O(10^{15})$, respectively. However, simplification is possible if the neurons are sequentially updated .

In order to simplify the algorithm, we begin by reconsidering (1) and (2) of the neural network. Noting that the interconnection strengths given in (9) are independent of subscripts k and l and the bias inputs given in (10) are independent of subscript k, the M neurons used to represent the same image gray level function have the same interconnection strengths and bias inputs. Hence, one set of interconnection strengths and one bias input are sufficient for every gray level function, i.e. the dimensions of the interconnection matrix T and bias input matrix I can be reduced by a factor of M^2. From (1) all inputs received by a neuron, say, the (i,k)th neuron can be written as

$$\begin{aligned} u_{i,k} &= \sum_{j}^{L^2} T_{i,.j,.} \left(\sum_{l}^{M} v_{j,l}\right) + I_{i,.} \\ &= \sum_{j}^{L^2} T_{i,.j,.}\, x_j + I_{i,.} \end{aligned} \tag{14}$$

where we have used (4) and x_j is the gray level function of the jth image pixel. Equation (14) suggests that we can use a multivalue number to replace the simple sum number. Since the interconnection strengths are determined by the blur function only as shown in (9), it is easy to see that if the blur function is local, then most interconnection strengths are zeros so that the neurons are locally connected. Therefore, most elements of the interconnection matrix T are zeros. If the blur function is shift invariant taking the form in (6), then the interconnection matrix is block Toeplitz so that only a few elements need to be stored. Based on the value of inputs $u_{i,k}$, the state of the (i,k)th neuron is updated by applying a decision rule. The state change of the (i,k)th neuron in turn causes the gray level function x_i to change

$$x_i^{new} = \begin{cases} x_i^{old} & if\ \ \Delta v_{i,k} = 0 \\ x_i^{old} + 1 & if\ \ \Delta v_{i,k} = 1 \\ x_i^{old} - 1 & if\ \ \Delta v_{i,k} = -1 \end{cases} \tag{15}$$

where $\Delta v_{i,k} = v_{i,k}^{new} - v_{i,k}^{old}$ is the state change of the (i,k)th neuron. The supscripts "new" and "old" are for after and before updating, respectively. We use x_i to respresent the gray level value as well as the output of M neurons representing x_i. Assuming that the neurons of the network are sequentially visited, it is straightforward to prove that the updating procedure can be reformulated as

$$u_{i,k} = \sum_{j}^{L^2} T_{i,.j,.}\, x_j + I_{i,.} \tag{16}$$

$$\Delta v_{i,k} = g(u_{i,k}) = \begin{cases} \Delta v_{i,k} = 0 & if\ \ u_{i,k} = 0 \\ \Delta v_{i,k} = 1 & if\ \ u_{i,k} > 0 \\ \Delta v_{i,k} = -1 & if\ \ u_{i,k} < 0 \end{cases} \tag{17}$$

$$x_i^{new} = \begin{cases} x_i^{old} + \Delta v_{i,k} & if\ \ \Delta E < 0 \\ x_i^{old} & if\ \ \Delta E \geq 0 \end{cases} \tag{18}$$

Note that the stochastic decision rule can also be used in (18). In order to limit the gray level function to the range 0 to 255, after each updating step we have to check the value of the gray level function x_i^{new}. Equations (16), (17) and (18) give a much simpler algorithm. This algorithm is summarized below:

1. Take the degraded image as the initial value.
2. Sequentially visit all numbers (image pixels). For each number, use (16), (17) and (18) to update it repeatedly until no further change, i.e. if $\Delta v_{i,k} = 0$ or energy change $\Delta E \geq 0$, then move to next one.
3. Check the energy function; if energy does not change anymore, a restored image is obtained;

otherwise, go back to step 2 for another iteration.

The calculations of the inputs $u_{i,k}$ of the (i,k)th neuron and the energy change ΔE can be simplified furthermore. When we update the same image gray level function repeatedly, the inputs received by the current neuron (i,k) can be computed by making use of the previous result

$$u_{i,k} = u_{i,k-1} + \Delta v_{i,k}\, T_{i,i,k,k} \tag{19}$$

where $u_{i,k-1}$ is the inputs received by the $(i,k-1)$th neuron. The energy change ΔE due to the state change of the (i,k)th neuron can be calculated as

$$\Delta E = -u_{i,k}\, \Delta v_{i,k} - \frac{1}{2}\, T_{i,i,k,k}\, (\Delta v_{i,k})^2 \tag{20}$$

If the blur function is shift invariant, all these simplifications reduce the space and time complexities significantly from $O(L^4M^2)$ and $O(L^4M^2K)$ to $O(L^2)$ and $O(ML^2K)$, respectively. Since every gray level function needs only a few updating steps after the first iteration, the computation at each iteration is $O(L^2)$. The resulting algorithm can be easily simulated on mini-computers for images, as large 512×512.

6 Computer Simulations

The practical algorithm described in the previous section was applied to the synthetic and real images on a Sun-3/160 Workstation. In all cases, only the deterministic decision rule was used. The results are summarized in Figure 1 and 2.

Figure 1 shows the results for the synthetic image. The original image shown in Figure 1(a) is of size 32 x 32 with 3 gray levels. The image was degraded by convolving with a 3 x 3 blur function as in (6) using a circulant boundary condition; 22 dB white Gaussian noise was added after convolution. A perfect image was obtained after 6 iterations without preprocessing. We set the state of all neurons to equal 1, i.e. firing as initial condition.

Figure 2(a) shows the original girl image. The original image is of size 256 x 256 with 256 gray levels. The variance of the original image is 2826.128. It was degraded by a 5 x 5 uniform blur function. A small amount of quantization noise was introduced by quantizing the convolution results to 8 bits. The noisy blurred image is shown in Figure 2(b). For comparison purpose, Figure 2(c) shows the output of an inverse filter [7], completely overridden by the amplified noise and the ringing effects due to the ill conditioned of the blur matrix H. Since the blur matrix H corresponding to the 5 x 5 uniform blur function is not singular, the pseudoinverse filter [7] and the inverse filter have the same output. The restored image by using our approach is shown in Figure 2(d). In order to eliminate the ringing effect, due to the boundary conditions, we took the 4 pixel wide boundaries from the original image and updated the interior region (248 x 248) of the image only. The blurred imgage was used as an initial condition for accelerating the convergence. The total number of iterations was 213 (when the energy function did not change anymore). The square error (i.e. energy function) defined in (8) is 0.02543 and the square error between the original and restored imges is 66.5027.

7 Conclusion

This paper has introduced a novel approach to restore gray level images degraded by a shift invariant blur function and additive noise. The restoration procedure consists of two steps: parameter estimation and image reconstruction. In order to reduce the computational complexity, a practical algorithm which is equivalent to the original one is developed under the assumption that the neurons are sequentially visited. The image is generated iteratively by updating the neurons representing the image gray levels

via a simple sum scheme. As no matrices are inverted, the serious problem of ringing due to the ill conditioned blur matrix H and noise overriding caused by inverse filter or pseudoinverse inverse filter can be avoided. For the case of 2-D uniform blur plus noise, the neural network based approach give high quality images whereas the inverse filter and pseudoinverse filter yield poor results. We see from the experimental results that the error defined by (8) is small while the error between the original image and the restored image is relatively large. This is because the neural network decreases energy according to (8) only. Another reason is that when the blur matrix is singular or near singular, the mapping from $\underline{X}$ to $\underline{Y}$ is not one to one, therefore, the error measure (8) is not reliable anymore. Thus, we have to point out that our approach will not work very well when the bluuring matrix is singular. In our experiments, when the window size of a uniform blur function is 3×3, the ringing effect was eliminated by leaving the boundaries of the degraded image without processing. When the window size is 5×5, the ringing effect was significantly reduced by using the original image boundaries.

References

[1] N. H. Farhat, D. Psaltis, A. Prata, and E. Paek, "Optical Implementation of the Hopfield Model", *Applied Optics*, vol. 24, No.10, pp. 1469–1475, 15 May 1985.

[2] J. J. Hopfield and D. W. Tank, "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, vol. 52, pp. 114–152, 1985.

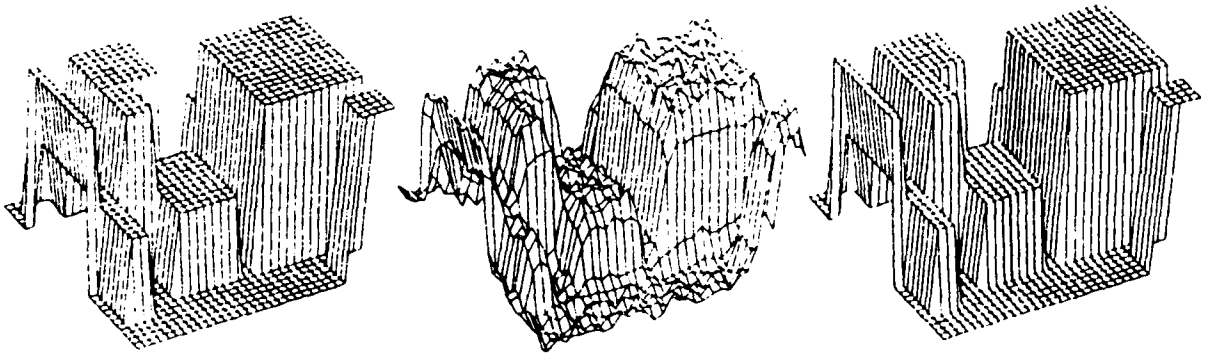
[3] J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554–2558, April 1982.

[4] M. Takeda and J. W. Goodman, "Neural Networks for Computation: Number Representations and Programming Complexity", *Applied Optics*, vol. 25, No. 18, pp. 3033–3046, Sept. 1986.

[5] N. Metropolis et al., "Equations of State Calculations by Fast Computing Machines", *J. Chem. Phys.*, vol. 21, pp. 1087–1091, 1953.

[6] S. Kirkpatrick et al., "Optimization by Stimulated Annealing", *Science*, vol. 220, pp. 671–680, 1983.

[7] W. K. Pratt, *Digital Image Processing*, John Wiley and Sons, New York, New York, 1978.



(a) Original image. (b) Degraded image. (c) Results after 6 iterations.

Figure 1: Restoration of noisy blurred synthetic image.

(a) Original girl image.

(b) Image degraded by 5 × 5 uniform blur and quantization noise

(c) Rsetored image using inverse filter.

(d) Restored image using our approach.

Figure 2: Restoration of noisy blurred real image and comparison.

Image Restoration Using a Neural Network

YI-TONG ZHOU, STUDENT MEMBER, IEEE, RAMA CHELLAPPA, SENIOR MEMBER, IEEE, ASEEM VAID, AND B. KEITH JENKINS, MEMBER, IEEE

Abstract—A new approach for restoration of gray level images degraded by a known shift-invariant blur function and additive noise is presented using a neural computational network. A neural network model is employed to represent a possibly nonstationary image whose gray level function is the simple sum of the neuron state variables. The restoration procedure consists of two stages: estimation of the parameters of the neural network model and reconstruction of images. During the first stage, the parameters are estimated by comparing the energy function of the network to a constrained error function. The nonlinear restoration method is then carried out iteratively in the second stage by using a dynamic algorithm to minimize the energy function of the network. Owing to the model's fault-tolerant nature and computation capability, a high-quality image is obtained using this approach. A practical algorithm with reduced computational complexity is also presented. Several computer simulation examples involving synthetic and real images are given to illustrate the usefulness of our method. The choice of the boundary values to reduce the ringing effect is discussed, and comparisons to other restoration methods such as the SVD pseudoinverse filter, minimum mean-square error (MMSE) filter, and modified MMSE filter using the Gaussian Markov random field model are given. Finally, a procedure for learning the blur parameters from prototypes of original and degraded images is outlined.

I. Introduction

RESTORATION of a high-quality image from a degraded recording is an important problem in early vision processing. Restoration techniques are applied to remove 1) system degradations such as blur due to optical system aberrations, atmospheric turbulence, motion, and diffraction; and 2) statistical degradations due to noise. Over the last 20 years, various methods such as the inverse filter [1], Wiener filter [1], Kalman filter [2], SVD pseudoinverse [1], [3], and many other model-based approaches have been proposed for image restorations. One of the major drawbacks of most of the image restoration algorithms is the computational complexity, so much so that many simplifying assumptions such as wide sense stationarity (WSS), availability of second-order image statistics have been made to obtain computationally feasible algorithms. The inverse filter method works only for extremely high signal-to-noise ratio images. The Wiener filter is usually implemented only after the wide sense stationary assumption has been made for images. Furthermore, knowledge of the power spectrum or correlation matrix of the undegraded image is required. Often times, additional assumptions regarding boundary conditions are made so that fast orthogonal transforms can be used. The Kalman filter approach can be applied to nonstationary image, but is computationally very intensive. Similar statements can be made for the SVD pseudoinverse filter method. Approaches based on noncausal models such as the noncausal autoregressive or Gauss Markov random field models [4], [5] also make assumptions such as WSS and periodic boundary conditions. It is desirable to develop a restoration algorithm that does not make WSS assumptions and can be implemented in a reasonable time. An artificial neural network system that can perform extremely rapid computations seems to be very attractive for image restoration in particular and image processing and pattern recognition [6] in general.

In this paper, we use a neural network model containing redundant neurons to restore gray level images degraded by a known shift-invariant blur function and noise. It is based on the method described in [7]–[9] using a simple sum number representation [10]. The image gray levels are represented by the simple sum of the neuron state variables which take binary values of 1 or 0. The observed image is degraded by a shift-invariant function and noise. The restoration procedure consists of two stages: estimation of the parameters of the neural network model and reconstruction of images. During the first stage, the parameters are estimated by comparing the energy function of the neural network to the constrained error function. The nonlinear restoration algorithm is then implemented using a dynamic iterative algorithm to minimize the energy function of the neural network. Owing to the model's fault-tolerant nature and computation capability, a high-quality image is obtained using this approach. In order to reduce computational complexity, a practical algorithm, which has equivalent results to the original one suggested above, is developed under the assumption that the neurons are sequentially visited. We illustrate the usefulness of this approach by using both synthetic and real images degraded by a known shift-invariant blur function with or without noise. We also discuss the problem of choosing boundary values and introduce two methods to reduce the ringing effect. Comparisons to other restoration methods such as the SVD pseudoinverse filter, the minimum mean-square error (MMSE) filter, and the modified MMSE filter using a Gaussian Markov random field model are given using real images. The advantages of the method developed in this paper are: 1) WSS assumption is not required

Manuscript received February 22, 1988. This work was supported in part by AFOSR Contract F-49620-87-C-0007 and AFOSR Grant 86-0196.

The authors are with the Signal and Image Processing Institute, Department of Electrical Engineering—Systems, University of Southern California, Los Angeles, CA 90089.

IEEE Log Number 8821366.

0096-3518/88/0700-1141$01.00 © 1988 IEEE

for the images, 2) it can be implemented rapidly, and 3) it is fault tolerant.

In the above, the interconnection strengths (also called weights) of the neural network for image restoration are known from the parameters of the image degradation model and the smoothing constraints. We also consider learning of the parameters for the image degradation model and formulate it as a problem of computing the parameters from samples of the original and degraded images. This is implemented as a secondary neural network. A different scheme is used to represent multilevel activities for the parameters; some of its properties are complementary to those of the simple sum scheme. The learning procedure is accomplished by running a greedy algorithm. Some results of learning the blur parameters are presented using synthetic and real image examples.

The organization of this paper is as follows. A network model containing redundant neurons for image representation and the image degradation model is given in Section II. A technique for parameter estimation is presented in Section III. Image generation using a dynamic algorithm is described in Section IV. A practical algorithm with reduced computational complexity is presented in Section V. Computer simulation results using synthetic and real degraded images are given in Section VI. Choice of the boundary values is discussed in Section VII. Comparisons to other methods are given in Section VIII. A procedure for learning the blur parameters from prototypes of original and degraded images is outlined in Section IX, and conclusions and remarks are included in Section X.

II. A Neural Network for Image Representation

We use a neural network containing redundant neurons for representing the image gray levels. The model consists of $L^2 \times M$ mutually interconnected neurons where L is the size of image and M is the maximum value of the gray level function. Let $V = \{ v_{i,k}$ where $1 \le i \le L^2$, $1 \le k \le M \}$ be a binary state set of the neural network with $v_{i,k}$ (1 for firing and 0 for resting) denoting the state of the (i, k)th neuron. Let $T_{i,k;j,l}$ denote the strength (possibly negative) of the interconnection between neuron (i, k) and neuron (j, l). We require symmetry:

$$T_{i,k;j,l} = T_{j,l;i,k} \quad \text{for } 1 \le i, j \le L^2 \text{ and } 1 \le l, k \le M.$$

We also allow for neurons to have self-feedback, i.e., $T_{i,k;i,k} \neq 0$. In this model, each neuron (i, k) randomly and asynchronously receives inputs $\Sigma T_{i,k;j,l} v_{j,l}$ from all neurons and a bias input $I_{i,k}$:

$$u_{i,k} = \sum_{j}^{L^2} \sum_{l}^{M} T_{i,k;j,l} v_{j,l} + I_{i,k}. \tag{1}$$

Each $u_{i,k}$ is fed back to corresponding neurons after thresholding:

$$v_{i,k} = g(u_{i,k}) \tag{2}$$

where $g(x)$ is a nonlinear function whose form can be taken as

$$g(x) = \begin{cases} 1 & \text{if } x \ge 0 \\ 0 & \text{if } x < 0. \end{cases}$$

In this model, the state of each neuron is updated by the latest information about other neurons.

The image is described by a finite set of gray level functions $\{x(i, j)$ where $1 \le i, j \le L\}$ with $x(i, j)$ (positive integer number) denoting the gray level of the pixel (i, j). The image gray level function can be represented as a simple sum of the neuron state variables as

$$x(i, j) = \sum_{k=1}^{M} v_{m,k}$$

where $m = (i - 1) \times L + j$. Here the gray level functions have degenerate representations. Use of this redundant number representation scheme yields advantages such as fault tolerance and faster convergence to the solution.

By using the lexicographic notation, the image degradation model can be written as

$$Y = HX + N$$

where H is the "blur matrix" corresponding to a blur function, N is the signal independent white noise, X and Y are the original and degraded images, respectively. Furthermore, H and N can be represented as

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,L^2} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,L^2} \\ \vdots & \vdots & \cdots & \vdots \\ h_{L^2,1} & h_{L^2,2} & \cdots & h_{L^2,L^2} \end{bmatrix}$$

and

$$N = \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_L \end{bmatrix} = \begin{bmatrix} n_1 \\ n_2 \\ \vdots \\ n_{L^2} \end{bmatrix},$$

$$N_i = \begin{bmatrix} n(i, 1) \\ n(i, 2) \\ \vdots \\ n(i, L) \end{bmatrix} = \begin{bmatrix} n_{(i-1)\times L+1} \\ n_{(i-1)\times L+2} \\ \vdots \\ n_{i\times L} \end{bmatrix}$$

respectively. Vectors X and Y have similar representations. Equation (5) is similar to the simultaneous equations solution of [10], but differs in that it includes a noise term.

The shift-invariant blur function can be written as a convolution over a small window, for instance, it can

the form

$$h(k,l) = \begin{cases} \frac{1}{2} & \text{if } k = 0,\ l = 0 \\ \frac{1}{16} & \text{if } |k|, |l| \le 1,\ (k,l) \ne (0,0); \end{cases} \tag{8}$$

accordingly, the "blur matrix" H will be a block Toeplitz or block circulant matrix (if the image has periodic boundaries). The block circulant matrix corresponding to (8) can be written as

$$H = \begin{bmatrix} H_0 & H_1 & 0 & \cdots & 0 & H_1 \\ H_1 & H_0 & H_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ H_1 & 0 & 0 & \cdots & H_1 & H_0 \end{bmatrix} \tag{9}$$

where

$$H_0 = \begin{bmatrix} \frac{1}{2} & \frac{1}{16} & 0 & \cdots & 0 & \frac{1}{16} \\ \frac{1}{16} & \frac{1}{2} & \frac{1}{16} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \frac{1}{16} & 0 & 0 & \cdots & \frac{1}{16} & \frac{1}{2} \end{bmatrix},$$

$$H_1 = \begin{bmatrix} \frac{1}{16} & \frac{1}{16} & 0 & \cdots & 0 & \frac{1}{16} \\ \frac{1}{16} & \frac{1}{16} & \frac{1}{16} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ \frac{1}{16} & 0 & 0 & \cdots & \frac{1}{16} & \frac{1}{16} \end{bmatrix} \tag{10}$$

and $\mathbf{0}$ is null matrix whose elements are all zeros.

III. Estimation of Model Parameters

The neural model parameters, the interconnection strengths, and bias inputs can be determined in terms of the energy function of the neural network. As defined in [7], the energy function of the neural network can be written as

$$E = -\frac{1}{2}\sum_{i=1}^{L^2}\sum_{j=1}^{L^2}\sum_{k=1}^{M}\sum_{l=1}^{M} T_{i,k;j,l} v_{i,k} v_{j,l} - \sum_{i=1}^{L^2}\sum_{k=1}^{M} I_{i,k} v_{i,k}. \tag{11}$$

In order to use the spontaneous energy-minimization process of the neural network, we reformulate the restoration problem as one of minimizing an error function with constraints defined as

$$E = \frac{1}{2}\left\| Y - H\hat{X} \right\|^2 + \frac{1}{2}\lambda \left\| D\hat{X} \right\|^2 \tag{12}$$

where $\| Z \|$ is the L_2 norm of Z and λ is a constant. Such a constrained error function is widely used in the image restoration problems [1] and is also similar to the regularization techniques used in early vision problems [11]. The first term in (12) is to seek an $\hat{X}$ such that $H\hat{X}$ approximates Y in a least squares sense. Meanwhile, the second term is a smoothness constraint on the solution $\hat{X}$. The constant λ determines their relative importance to achieve both noise suppression and ringing reduction.

In general, if H is a low-pass distortion, then D is a high-pass filter. A common choice of D is a second-order differential operator which can be approximated as a local window operator in the 2-D discrete case. For instance, if D is a Laplacian operator

$$\nabla = \frac{\partial^2}{\partial i^2} + \frac{\partial^2}{\partial j^2} \tag{13}$$

it can be approximated as a window operator

$$\frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}. \tag{14}$$

Then D will be a block Toeplitz matrix similar to (9).

Expanding (12) and then replacing x_i by (4), we have

$$\begin{aligned} E &= \frac{1}{2}\sum_{p=1}^{L^2}\left(y_p - \sum_{i=1}^{L^2} h_{p,i}x_i\right)^2 + \frac{1}{2}\lambda\sum_{p=1}^{L^2}\left(\sum_{i=1}^{L^2} d_{p,i}x_i\right)^2 \\ &= \frac{1}{2}\sum_{i=1}^{L^2}\sum_{j=1}^{L^2}\sum_{k=1}^{M}\sum_{l=1}^{M}\sum_{p=1}^{L^2} h_{p,i}h_{p,j}v_{i,k}v_{j,l} \\ &\quad + \frac{1}{2}\lambda\sum_{i=1}^{L^2}\sum_{j=1}^{L^2}\sum_{k=1}^{M}\sum_{l=1}^{M}\sum_{p=1}^{L^2} d_{p,i}d_{p,j}v_{i,k}v_{j,l} \\ &\quad - \sum_{i=1}^{L^2}\sum_{k=1}^{M}\sum_{p=1}^{L^2} y_p h_{p,i} v_{i,k} + \frac{1}{2}\sum_{p=1}^{L^2} y_p^2. \end{aligned} \tag{15}$$

By comparing the terms in (15) to the corresponding terms in (11) and ignoring the constant term $\frac{1}{2}\sum_{p=1}^{L^2} y_p^2$, we can determine the interconnection strengths and bias inputs as

$$T_{i,k;j,l} = -\sum_{p=1}^{L^2} h_{p,i}h_{p,j} - \lambda\sum_{p=1}^{L^2} d_{p,i}d_{p,j} \tag{16}$$

and

$$I_{i,k} = \sum_{p=1}^{L^2} y_p h_{p,i} \tag{17}$$

where $h_{i,j}$ and $d_{i,j}$ are the elements of the matrices H and D, respectively. Two interesting aspects of (16) and (17) should be pointed out: 1) the interconnection strengths are independent of subscripts k and l and the bias inputs are independent of subscript k, and 2) the self-connection $T_{i,k;i,k}$ is not equal to zero which requires self-feedback for neurons.

From (16), one can see that the interconnection strengths are determined by the shift-invariant blur function, differential operator, and constant λ. Hence, $T_{i,k;j,l}$ can be computed without error provided the blur function

is known. However, the bias inputs are functions of the observed degraded image. If the image is degraded by a shift-invarian: blur function only, then $I_{i,k}$ can be estimated perfectly Otherwise, $I_{i,k}$ is affected by noise. The reasoning behind this statement is as follows. By replacing y_p by $\sum_{i=1}^{L^2} h_{p,i}x_i + n_p$, we have

$$I_{i,k} = \sum_{p=1}^{L^2}\left(\sum_{i=1}^{L^2} h_{p,i}x_i + n_p\right)h_{p,i} = \sum_{p=1}^{L^2}\sum_{i=1}^{L^2} h_{p,i}x_i h_p + \sum_{p=1}^{L^2} n_p h_{p,i}. \quad (18)$$

The second term in (18) represents the effects of noise. If the signal-to-noise ratio (SNR), defined by

$$\text{SNR} = 10\log_{10}\frac{\sigma_s^2}{\sigma_n^2} \quad (19)$$

where σ_s^2 and σ_n^2 are variances of signal and noise, respectively, is low, then we have to choose a large λ to suppress effects due to noise. It seems that in the absence of noise, the parameters can be estimated perfectly, ensuring exact recovery of the image as error function E tends to zero. However, the problem is not so simple because the restoration performance depends on both the parameters and the blur function when a mean-square error or least square error such as (12) is used. A discussion about the effect of blur function is given in Section X.

IV. Restoration

Restoration is carried out by neuron evaluation and an image construction procedure. Once the parameters $T_{i,k;j,l}$ and $I_{i,k}$ are obtained using (16) and (17), each neuron can randomly and asynchronously evaluate its state and readjust accordingly using (1) and (2). When one quasi-minimum energy point is reached, the image can be constructed using (4).

However, this neural network has self-feedback, i.e., $T_{i,k;i,k} \neq 0$. As a result, the energy function E does not always decrease monotonically with a transition. This is explained below. Define the state change $\Delta v_{i,k}$ of neuron (i, k) and energy change ΔE as

$$\Delta v_{i,k} = v_{i,k}^{\text{new}} - v_{i,k}^{\text{old}} \quad \text{and} \quad \Delta E = E^{\text{new}} - E^{\text{old}}.$$

Consider the energy function

$$E = -\frac{1}{2}\sum_{i=1}^{L^2}\sum_{j=1}^{L^2}\sum_{k=1}^{M}\sum_{l=1}^{M} T_{i,k;j,l}v_{i,k}v_{j,l} - \sum_{i=1}^{L^2}\sum_{k=1}^{M} I_{i,k}v_{i,k}. \quad (20)$$

Then the change ΔE due to a change $\Delta v_{i,k}$ is given by

$$\Delta E = -\left(\sum_{j=1}^{L^2}\sum_{l=1}^{M} T_{i,k;j,l}v_{j,l} + I_{i,k}\right)\Delta v_{i,k} - \frac{1}{2}T_{i,k;i,k}(\Delta v_{i,k})^2 \quad (21)$$

which is not always negative. For instance, if

$$v_{i,k}^{\text{old}} = 0, \quad u_{i,k} = \sum_{j=1}^{L^2}\sum_{l=1}^{M} T_{i,k;j,l}v_{j,l} + I_{i,k} > 0$$

and the threshold function is as in (3), then $v_{i,k}^{\text{new}} = 1$ $\Delta v_{i,k} > 0$. Thus, the first term in (21) is negative. B

$$T_{i,k;i,k} = -\sum_{p=1}^{L^2} h_{p,i}^2 - \lambda\sum_{p=1}^{L^2} d_{p,i}^2 < 0$$

with $\lambda > 0$, leading to

$$-\frac{1}{2}T_{i,k;i,k}(\Delta v_{i,k})^2 > 0.$$

When the first term is less than the second term in then $\Delta E > 0$ (we have observed this in our experin which means E is not a Lyapunov function. C quently, the convergence of the network is not guara [12].

Thus, depending on whether convergence to a minimum or a global minimum is desired, we can d a deterministic or stochastic decision rule. The dete istic rule is to take a new state $v_{i,k}^{\text{new}}$ of neuron (i, k) energy change ΔE due to state change $\Delta v_{i,k}$ is less zero. If ΔE due to state change is > 0, no state ci is affected. One can also design a stochastic rule si to the one used in stimulated annealing techniques [14]. The details of this stochastic scheme are giv follows.

Define a Boltzmann distribution by

$$\frac{p_{\text{new}}}{p_{\text{old}}} = e^{-\Delta E/T}$$

where p_{new} and p_{old} are the probabilities of the ne old global state, respectively, ΔE is the energy ch and T is the parameter which acts like temperature. state $v_{i,k}^{\text{new}}$ is taken if

$$\frac{p_{\text{new}}}{p_{\text{old}}} > 1 \text{ or if } \frac{p_{\text{new}}}{p_{\text{old}}} \leq 1 \text{ but } \frac{p_{\text{new}}}{p_{\text{old}}} > \xi$$

where ξ is a random number uniformly distributed interval [0, 1].

The restoration algorithm is summarized as belov

Algorithm 1:

1) Set the initial state of the neurons.

2) Update the state of all neurons randomly and chronously according to the decision rule.

3) Check the energy function; if energy doe change, go to step 4); otherwise, go back to step 2

4) Construct an image using (4).

V. A Practical Algorithm

The algorithm described above is difficult to sim on a conventional computer owing to high computat complexity, even for images of reasonable size. Fo stance, if we have an $L \times L$ image with M gray le then L^2M neurons and $\frac{1}{2}L^4M^2$ interconnections ar quired and L^4M^2 additions and multiplications are n

at each iteration. Therefore, the space and time complexities are $O(L^4M^2)$ and $O(L^4M^2K)$, respectively, where K, typically 10–100, is the number of iterations. Usually, L and M are 256–1024 and 256, respectively. However, simplification is possible if the neurons are sequentially updated.

In order to simplify the algorithm, we begin by reconsidering (1) and (2) of the neural network. As noted earlier, the interconnection strengths given in (16) are independent of subscripts k and l and the bias inputs given in (17) are independent of subscript k; the M neurons used to represent the same image gray level function have the same interconnection strengths and bias inputs. Hence, one set of interconnection strengths and one bias input are sufficient for every gray level function, i.e., the dimensions of the interconnection matrix T and bias input matrix I can be reduced by a factor of M^2. From (1), all inputs received by a neuron, say the (i, k)th neuron, can be written as

$$u_{i,k} = \sum_{j}^{L^2} T_{i,\cdot;j,\cdot} \cdot \left(\sum_{l}^{M} v_{j,l} \right) + I_{i,\cdot} = \sum_{j}^{L^2} T_{i,\cdot;j,\cdot} \cdot x_j + I_{i,\cdot} \tag{22}$$

where we have used (4) and x_j is the gray level function of the jth image pixel. The symbol "$\cdot$" in the subscripts means that the $T_{i,\cdot;j,\cdot}$ and $I_{i,\cdot}$ are independent of k. Equation (22) suggests that we can use a multivalue number to replace the simple sum number. Since the interconnection strengths are determined by the blur function, the differential operator, and the constant λ as shown in (16), it is easy to see that if the blur function is local, then most interconnection strengths are zeros and the neurons are locally connected. Therefore, most elements of the interconnection matrix T are zeros. If the blur function is shift invariant taking the form in (8), then the interconnection matrix is block Toeplitz so that only a few elements need to be stored. Based on the value of inputs $u_{i,k}$, the state of the (i, k)th neuron is updated by applying a decision rule. The state change of the (i, k)th neuron in turn causes the gray level function x_i to change:

$$x_i^{\text{new}} = \begin{cases} x_i^{\text{old}} & \text{if } \Delta v_{i,k} = 0 \\ x_i^{\text{old}} + 1 & \text{if } \Delta v_{i,k} = 1 \\ x_i^{\text{old}} - 1 & \text{if } \Delta v_{i,k} = -1 \end{cases} \tag{23}$$

where $\Delta v_{i,k} = v_{i,k}^{\text{new}} - v_{i,k}^{\text{old}}$ is the state change of the (i, k)th neuron. The superscripts "new" and "old" are for after and before updating, respectively. We use x_i to represent the gray level value as well as the output of M neurons representing x_i. Assuming that the neurons of the network are sequentially visited, it is straightforward to show that the updating procedure can be reformulated as

$$u_{i,k} = \sum_{j}^{L^2} T_{i,\cdot;j,\cdot} \cdot x_j + I_{i,\cdot} \tag{24}$$

$$\Delta v_{i,k} = g(u_{i,k}) = \begin{cases} \Delta v_{i,k} = 0 & \text{if } u_{i,k} = 0 \\ \Delta v_{i,k} = 1 & \text{if } u_{i,k} > 0 \\ \Delta v_{i,k} = -1 & \text{if } u_{i,k} < 0 \end{cases} \tag{25}$$

$$x_i^{\text{new}} = \begin{cases} x_i^{\text{old}} + \Delta v_{i,k} & \text{if } \Delta E < 0 \\ x_i^{\text{old}} & \text{if } \Delta E \geq 0. \end{cases} \tag{26}$$

Note that the stochastic decision rule can also be used in (26). In order to limit the gray level function to the range 0–255 after each updating step, we have to check the value of the gray level function x_i^{new}. Equations (24), (25), and (26) give a much simpler algorithm. This algorithm is summarized below.

Algorithm 2:

1) Take the degraded image as the initial value.

2) Sequentially visit all numbers (image pixels). For each number, use (24), (25), and (26) to update it repeatedly until there is no further change, i.e., if $\Delta v_{i,k} = 0$ or energy change $\Delta E \geq 0$; then move to the next one.

3) Check the energy function; if energy does not change anymore, a restored image is obtained; otherwise, go back to step 2) for another iteration.

The calculations of the inputs $u_{i,k}$ of the (i, k)th neuron and the energy change ΔE can be simplified furthermore. When we update the same image gray level function repeatedly, the input received by the current neuron (i, k) can be computed by making use of the previous result

$$u_{i,k} = u_{i,k-1} + \Delta v_{i,k} T_{i,\cdot;i,\cdot} \tag{27}$$

where $u_{i,k-1}$ is the inputs received by the $(i, k-1)$th neuron. The energy change ΔE due to the state change of the (i, k)th neuron can be calculated as

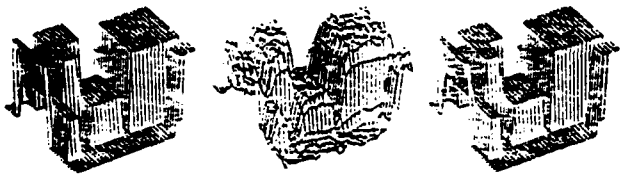
$$\Delta E = -u_{i,k} \Delta v_{i,k} - \tfrac{1}{2} T_{i,\cdot;i,\cdot} (\Delta v_{i,k})^2. \tag{28}$$

If the blur function is shift invariant, all these simplifications reduce the space and time complexities significantly from $O(L^4M^2)$ and $O(L^4M^2K)$ to $O(L^2)$ and $O(ML^2K)$, respectively. Since every gray level function needs only a few updating steps after the first iteration, the computation at each iteration is $O(L^2)$. The resulting algorithm can be easily simulated on minicomputers for images as large as 512 × 512.

VI. Computer Simulations

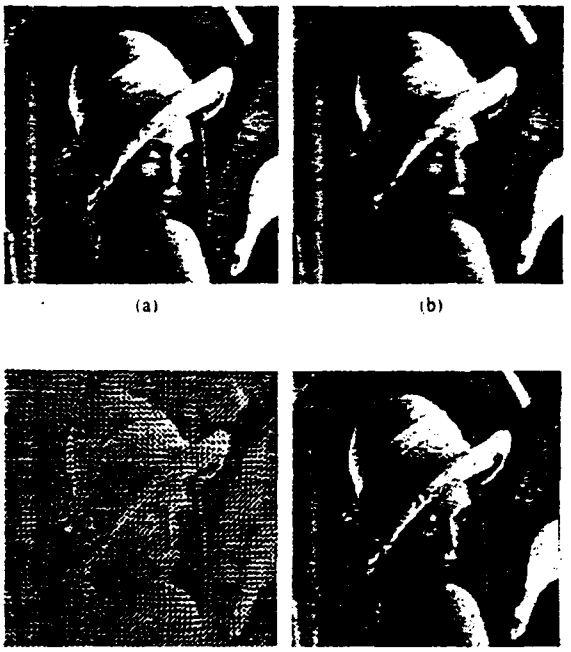
The practical algorithm described in the previous section was applied to synthetic and real images on a Sun-3/160 Workstation. In all cases, only the deterministic decision rule was used. The results are summarized in Figs. 1 and 2.

Fig. 1 shows the results for a synthetic image. The original image shown in Fig. 1(a) is of size 32 × 32 with three gray levels. The image was degraded by convolving with a 3 × 3 blur function as in (8) using circulant boundary conditions: 22 dB white Gaussian noise was added after convolution. A perfect image was obtained after six



(a) (b) (c)

Fig. 1. Restoration of noisy blurred synthetic image. (a) Original image. (b) Degraded image. (c) Result after six iterations.



(a) (b)

(c) (d)

Fig. 2. Restoration of noisy blurred real image. (a) Original girl image. (b) Image degraded by 5 × 5 uniform blur and quantization noise. (c) The restored image using inverse filter. (d) The restored image using our approach.

iterations without preprocessing. We set the initial state of all neurons to equal 1, i.e., firing, and chose $\lambda = 0$ due to the well conditioning of the blur function.

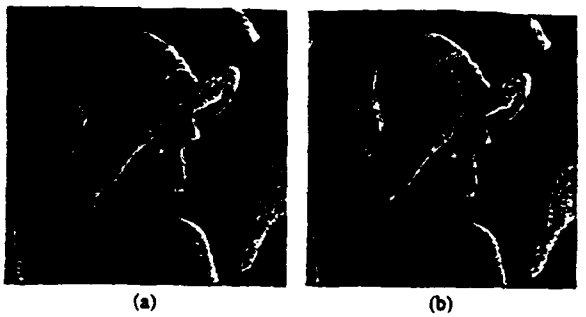
Fig. 2(a) shows the original girl image. The original image is of size 256 × 256 with 256 gray levels. The variance of the original image is 2797.141. It was degraded by a 5 × 5 uniform blur function. A small amount of quantization noise was introduced by quantizing the convolution results to 8 bits. The noisy blurred image is shown in Fig. 2(b). For comparison purpose, Fig. 2(c) shows the output of an inverse filter [15], completely overridden by the amplified noise and the ringing effects due to the ill-conditioned blur matrix H. Since the blur matrix H corresponding to the 5 × 5 uniform blur function is not singular, the pseudoinverse filter [15] and the inverse filter have the same output. The restored image by using our approach is shown in Fig. 2(d). In order to avoid the ringing effects due to the boundary conditions, we took 4 pixel wide boundaries, i.e., the first and last four rows and columns, from the original image and updated the interior region (248 × 248) of the image only. The noisy blurred image was used as an initial condition for acc erating the convergence. The constant λ was set to z because of small noise and good boundary values. restored image in Fig. 2(d) was obtained after 213 it tions. The square error (i.e., energy function) define (12) is 0.02543 and the square error between the orig and the restored image is 66.5027.

VII. Choosing Boundary Values

As mentioned in [16], choosing boundary values common problem for techniques ranging from determ istic inverse filter algorithms to stochastic Kalman filt In these algorithms, boundary values determine the en solution when the blur is uniform [17]. The same prob occurs in the neural network approach. Since the 5 uniform blur function is ill conditioned, improper bou ary values may cause ringing which may affect the stored image completely. For example, appending ze to the image as boundary values introduces a sharp e at the image border and triggers ringing in the resto image even if the image has zero mean. Another pr dure is to assume a periodic boundary. When the left (and right (bottom) borders of the image are differen sharp edge is formed and ringing results even though degraded image has been formed by blurring with p odic boundary conditions. The drawbacks of these assumptions for boundary values were reported in [[2], [18] for the 2-D Kalman filtering technique. We tested our algorithm using these two assumptions boundary values; the results indicate the restored ima were seriously affected by ringing.

In the last section, to avoid the ringing effect, we t 4 pixel wide borders from the original image as bound values for restoration. Since the original image is available in practice always, an alternative to elimi the ringing effect caused by sharp false edges is to use blurred noisy boundaries from the degraded image. 3(a) shows the restored image using the first and last rows and columns of the blurred noisy image in Fig. as boundary values. In the restored image, there still ists some ringing due to the naturally occurring sh edges in the region near the borders in the original ima but not due to boundary values. A typical cut of the stored image to illustrate ringing near the borders is sh in Fig. 4. To remove the ringing near the borders cau by naturally occurring sharp edges in the original ima we suggest the following techniques.

First, divide the image into three regions: border, border, and interior region as shown in Fig. 5. For th × 5 uniform blur case, the border region will be 4 pi wide due to the boundary effect of the bias input I_i (17), and the subborder region will be 4 or 8 pixels w In fact, the width of the subborder region will be im dependent. If the regions near the border are smooth, t the width of the subborder region will be small or e zero. If the border contains many sharp edges, the w will be large. For the real girl image, we chose the w



(a) (b)

(c)

Fig. 3. Results using blurred noisy boundaries. (a) Blurred noisy boundaries. (b) Method 1. (c) Method 2.

Fig. 4. One typical cut of the restored image using the blurred noisy boundaries. Solid line for original image, dashed line for blurred noisy image, and dashed and dotted line for restored image.

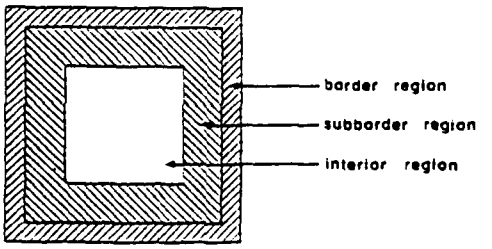


Fig. 5. Border, subborder, and interior regions of the image.

of the subborder region to be 8 pixels. We suggest using one of the following two methods.

Method 1: In the case of small noise, such as quantization error noise, the blurred image is usually smooth. Therefore, we restricted the difference between the restored and blurred image in the subborder region to a certain range to reduce the ringing effect. Mathematically, this constraint can be written as

$$\|\hat{x}_i - y_i\| \leq T \quad \text{for } i \in \text{subborder region} \tag{29}$$

where T is a threshold and $\hat{x}_i$ is the restored image gray value. Fig. 3(b) shows the result of using this method with $T = 10$.

Method 2: This method simply sets λ in (12) to zero in the interior region and nonzero in the subborder region, respectively. Fig. 3(c) shows the result of using this method with $\lambda = 0.09$. In this case, D was a Laplacian operator.

Owing to checking all restored image gray values in the subborder region, Method 1 needs more computation than Method 2. However, Method 2 is very sensitive to the parameter λ, while Method 1 is not so sensitive to the parameter λ. Experimental results show that both Methods 1 and 2 reduce the ringing effect significantly by using the suboptimal blurred boundary values.

VIII. Comparisons to Other Restoration Methods

Comparing the performance of different restoration methods needs some quality measures which are difficult to define owing to the lack of knowledge about the human visual system. The word "optimal" used in the restoration techniques usually refers only to a mathematical concept, and is not related to response of the human visual system. For instance, when the blur function is ill conditioned and the SNR is low, the MMSE method improves the SNR, but the resulting image is not visually good. We believe that human objective evaluation is the best ultimate judgment. Meanwhile, the mean-square error or least square error can be used as a reference.

For comparison purposes, we give the outputs of the inverse filter, SVD pseudoinverse filter, MMSE filter, and modified MMSE filter using the Gaussian Markov random field (GMRF) model [19], [5].

A. Inverse Filter and SVD Pseudoinverse Filter

An inverse filter can be used to restore an image degraded by a space-invariant blur function with high signal-to-noise ratio. When the blur function has some singular points, an SVD pseudoinverse filter is needed; however, both filters are very sensitive to noise. This is because the noise is amplified in the same way as the signal components to be restored. The inverse filter and SVD pseudoinverse filter were applied to an image degraded by the 5×5 uniform blur function and quantization noise (about 40 dB SNR). The blurred and restored images are shown in Fig. 2(b) and (c), respectively. As we mentioned before, the outputs of these filters are completely overridden by the amplified noise and ringing effects.

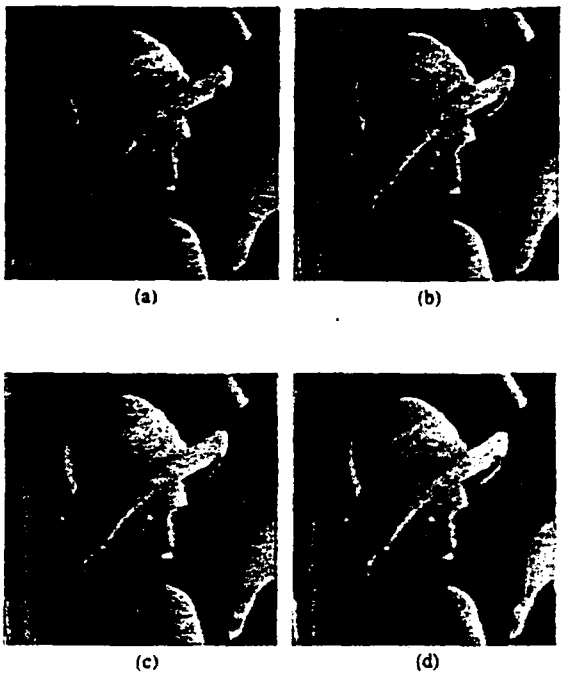


Fig. 6. Comparison to other restoration methods. (a) Image degraded by 5 × 5 uniform blur and 20 dB SNR additive white Gaussian noise. (b) The restored image using the MMSE filter. (c) The restored image using the modified MMSE filter. (d) The restored image using our approach.

B. MMSE and Modified MMSE Filters

The MMSE filter is also known as the Wiener filter (in the frequency domain). Under the assumption that the original image obeys a GMRF model, the MMSE filter (or Wiener filter) can be represented in terms of the GMRF model parameters and the blur function. In our implementation of the MMSE filter, we used a known blur function, unknown noise variance, and the GMRF model parameters estimated from the blurred noisy image by a maximum likelihood (ML) method [19]. The image shown in Fig. 6(a) was degraded by 5 × 5 uniform blur function and 20 dB SNR additive white Gaussian noise. The restored image is shown in Fig. 6(b).

The modified MMSE filter in terms of the GMRF model parameters is a linear weighted combination of a Wiener filter with a smoothing operator (such as a median filter) and a pseudoinverse filter to smooth the noise and preserve the edge of the restored image simultaneously. Details of this filter can be found in [5]. We applied the modified MMSE filter to the same image used in the MMSE filter above with the same model parameters. The smoothing operator is a 9 × 9 cross shape median filter. The resulting image is shown in Fig. 6(c).

The result of our method is also shown in Fig. 6(d). The D we used in (12) was a Laplacian operator as in (13). We chose $\lambda = 0.0625$ and used 4 pixel wide blurred noisy boundaries for restoration. The total number of iterations was 20. The improvement of mean-square error between the restored image and the original image for each method is shown in Table I. In the table, the "MMSE (o)" denotes that the parameters were estimated from the

TABLE I
MEAN-SQUARE ERROR IMPROVEMENT

Method	MMSE	MMSE (o)	Modified MMSE	Ne
Mean-square error	1.384 dB	2.139 dB	1.893 dB	1.6

original image. The restored image using "MMSE (is very similar to Fig. 6(a). As we mentioned before comparison of the outputs of the different restor methods is a difficult problem. The MMSE filter vis gives the worst output which has the smallest mean-sc error for the MMSE (o) case. The result of our me is smoother than that of the MMSE filter. Althoug output of the modified MMSE filter is smooth in fl gions, it contains some artifacts and snake effects a edges due to using a large sized median filter.

IX. Parameter Learning for Linear Image B Model

Apart from fine-grain parallelism, fast (and prefe automatic) adaptation of a problem-solving network t ferent instances of a problem is a primary motivatio using a network solution. For pattern recognition an sociative memory applications, this weight trainir done by distributed algorithms that optimize a dist measure between sample patterns and network respo However, in feedback networks, general problems involve learning higher order correlations (like the e sive OR) or combinatorial training sets (like the Trav Salesperson problem) are difficult to solve and may exponential complexity. In particular, techniques for ing a compact training set do not exist.

A. Learning Model

For model-based approaches to "neural" pro solving, the weights of the main network are comp from the parameters of the model. The learning pro can then be solved by a parallel, distributed algorithr estimating the model parameters from samples of th puts and desired outputs. This algorithm can be in mented on a secondary network. An error function fo "learning" network must be constructed, which will be problem-dependent.

For the linear shift-invariant blur model (5), the p lem is that of estimating the parameters correspondi the blur function in a $K \times K$ small window center each pixel. Rewrite (5) as

$$y(i,j) = z(i,j)^t h + n(i,j) \qquad i,j = 1, 2, \cdots$$

where t denotes the transpose operator and $z(i,j)$ a are $K^2 \times 1$ vectors corresponding to original image ples in a $K \times K$ window centered at (i,j) and blur f tion, respectively.

For instance, for $K = 3$, we have

$$h = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ \vdots \\ h_9 \end{bmatrix} = \begin{bmatrix} h(-1,-1) \\ h(-1,0) \\ h(-1,1) \\ \vdots \\ h(1,1) \end{bmatrix} \tag{31}$$

and

$$z(i,j) = \begin{bmatrix} z(i,j)_1 \\ z(i,j)_2 \\ z(i,j)_3 \\ \vdots \\ z(i,j)_9 \end{bmatrix} = \begin{bmatrix} x(i-1,j-1) \\ x(i-1,j) \\ x(i-1,j+1) \\ \vdots \\ x(i+1,j+1) \end{bmatrix}. \tag{32}$$

We can use an error function for estimation of h, as in the restoration process, because the roles of data $\{x(i,j)\}$ and parameter h are simply interchanged in the learning process. Therefore, an error function is defined as

$$E = \sum_{(i,j)\in S} \left[y(i,j) - h^t z(i,j) \right]^2 \tag{33}$$

where S is a subset of $\{(i,j), i,j = 1, 2, \cdots, L\}$ and $y(i,j)$ and $z(i,j)$ are training samples taken from the degraded and original images, respectively. The network energy functions is given by

$$E = -\sum_{k=1}^{K^2}\sum_{l=1}^{K^2} w_{kl} h_k h_l - \sum_{k=1}^{K^2} \theta_k h_k \tag{34}$$

where h_k are the multilevel parameter activities and w_{kl} and θ_k are the symmetric weights and bias inputs, respectively. From (33) and (34), we get the weights and bias inputs in the familiar outer-product forms:

$$w_{kl} = -\sum_{(i,j)\in S} z(i,j)_k \, z(i,j)_l \tag{35}$$

$$\theta_k = 2 \sum_{(i,j)\in S} z(i,j)_k \, y(i,j). \tag{36}$$

A greedy, distributed neural algorithm is used for the energy minimization. This leads to a localized multilevel number representation scheme for a general network.

B. Multilevel Greedy Distributed Algorithm

For a K^2 neuron second-order network, we choose Γ discrete activities $\{f_i, i = 0, 1, \cdots, \Gamma - 1\}$ in any arbitrary range of activities (e.g., $[0, 1]$) where we shall assume without loss of generality that $f_i > f_{i-1}$ for all i. Then, between any two activities f_m and f_n for the kth neuron, we can locally and asynchronously choose the one which results in the lowest energy given the current state of the other neurons because

$$E_{h_k = f_m} - E_{h_k = f_n} = \left[\theta_k - \zeta_k - (f_m + f_n) w_{k,k} \right] \cdot \left[f_m - f_n \right] \tag{37}$$

where

$$\zeta_k = \sum_{i,i\neq k}^{K^2} w_{i,k} h_i$$

is the current weighted sum from the other neuron activities. Thus, we choose level m over n for $m > n$ if

$$\zeta_k > \theta_k - (f_m + f_n) w_{k,k}. \tag{38}$$

Some properties of this algorithm follow.

1) Convergence is assured as long as the number of levels is not decreasing with time (i.e., assured if coarse to fine).

2) Self-feedback terms are included as level-dependent bias input terms.

3) The method can be easily extended to higher order networks (e.g., based on cubic energies). Appropriate lower order level-dependent networks (like the extra bias input term above) must then be implemented.

The multilevel lowest energy decision can be implemented by using variations of feedforward min-finding networks (such as those summarized in [20]). The space and time complexity of these networks are, in general, $O(\Gamma)$ and $O(\log \Gamma)$, respectively. However, in the quadratic case, it is easy to verify from (38) that we need only implement the decision between all *neighboring* levels in the set $\{f_i\}$; this requires exactly Γ neurons with level-dependent inputs. The best activity in the set is then proportional to the sum of the Γ neuron outputs so that the time complexity for the multilevel decision can be made $O(1)$. This means that this algorithm is similar in implementation complexity (e.g., the number of problem-dependent global interconnects required) to the simple sum energy representation used in [10] and in this paper. Also, in the simple sum case, visiting the neurons for each pixel in sequence will result in conditional energy minimization. Otherwise, from the implementation point of view, the two methods have some properties that are complementary. For example, we have the following.

1) The simple sum method requires asynchronism in the update steps for each pixel, while the greedy method does not.

2) The level-dependent terms arise as *inputs* in the greedy method as compared to *weights* in the simple sum method.

C. Simulation Results

The greedy algorithm was used with the weights from (35) and (36) to estimate the parameters from original and blurred sample points. A 5×5 window was used with two types of blurs: uniform and Gaussian. Both real and synthetic images were used, with and without additive Gaussian noise.

TABLE II
RESULTS FOR PARAMETER LEARNING. THE NUMBER Γ OF DISCRETE ACTIVITIES IS 256 FOR ALL TESTS. *A*: ARBITRARY CHOICE OF PIXELS FROM IMAGE. *L*: PIXELS CHOSEN FROM THRESHOLDED LAPLACIAN

Image	Noise	Blur	Samples	Methods	Iterations	MSE
Synthetic		Gaussian	68	*A*	49	0.000023
Synthetic		*Uniform*	100	*A*	114	0.000011
Real		Uniform	50	*A*	94	0.00353
Real		Uniform	100	*L*	85	0.00014
Real	20 dB	*Uniform*	100	*A*	72	0.00232
Real	20 dB	Uniform	100	*L*	83	0.00054

The estimated parameters for all types of blur matrices were numerically very close to the actual values when synthetic patterns were used. The network took longest to converge with a uniform blur function. The levels chosen for the discrete activity set $\{f_i\}$ were 128–256 equally spaced points in [0, 1] with 50–100 sample points from the image. Results for various cases are summarized in Table II.

When the sample pixels were randomly chosen, the errors increased by two orders of magnitude for a real image [Fig. 2(b)] as compared to synthetic ones. This is due to the smooth nature of real images. To solve this problem, sample points were chosen so as to lie close to *edges* in the image. This was done by thresholding the Laplacian of the image. Using sample points above a certain threshold for estimation improved the errors by an order of magnitude. The results were not appreciably degraded with 20 dB noise in the samples.

X. CONCLUSION

This paper has introduced a new approach for the restoration of gray level images degraded by a shift-invariant blur function and additive noise. The restoration procedure consists of two steps: parameter estimation and image reconstruction. In order to reduce computational complexity, a practical algorithm (Algorithm 2), which has equivalent results to the original one (Algorithm 1), is developed under the assumption that the neurons are sequentially visited. The image is generated iteratively by updating the neurons representing the image gray levels via a simple sum scheme. As no matrices are inverted, the serious problem of ringing due to the ill-conditioned blur matrix H and noise overriding caused by inverse filter or pseudoinverse inverse filter are avoided by using suboptimal boundary conditions. For the case of a 2-D uniform blur plus small noise, the neural network-based approach gives high-quality images compared to some of the existing methods. We see from the experimental results that the error defined by (12) is small, while the error between the original image and the restored image is relatively large. This is because the neural network decreases energy according to (12) only. Another reason is that when the blur matrix is singular or ill conditioned, the mapping from X to Y is not one to one; therefore, the error measure (12) is not reliable anymore. In our experiments, when the window size of a uniform blur function is 3 × 3, the ringing effect was eliminated by using blurred boundary values without any smoothing constraint. the window size is 5 × 5, the ringing effect was re with the help of the smoothing constraint and subop boundary conditions. We have also shown that a sr secondary network can effectively be used for estir the blur parameters; this provides a more efficient ing technique than Boltzman machine learning on th mary network.

REFERENCES

[1] H. C. Andrews and B. R. Hunt, *Digital Image Restoration*. wood Cliffs, NJ: Prentice-Hall, 1977.

[2] J. W. Woods and V. K. Ingle, "Kalman filtering in two dime Further results," *IEEE Trans. Acoust., Speech, Signal Proc* vol. ASSP-29, pp. 188–197, Apr. 1981.

[3] W. K. Pratt, *Digital Image Processing*. New York: Wiley,

[4] R. Chellappa and R. L. Kashyap, "Digital image restoratio spatial interaction models," *IEEE Trans. Acoust., Speech. Processing*, vol. ASSP-30, pp. 461–472, June 1982.

[5] H. Jinchi and R. Chellappa, "Restoration of blurred and nois using Gaussian Markov random field models," in *Proc. C form. Sci. Syst.*, Princeton Univ., Princeton, NJ, 1986, pp.

[6] N. H. Farhat, D. Psaltis, A. Prata, and E. Paek, "Optical imr tation of the Hopfield model," *Appl. Opt.*, vol. 24, pp. 146 May 15, 1985.

[7] J. J. Hopfield and D. W. Tank, "Neural computation of dec optimization problems," *Biol. Cybern.*, vol. 52, pp. 141–15

[8] J. J. Hopfield, "Neural networks and physical systems with e collective computational abilities," *Proc. Nat. Acad. Sci. U* 79, pp. 2554–2558, Apr. 1982.

[9] S.-I. Amari, "Learning patterns and pattern sequences by se nizing nets of threshold elements," *IEEE Trans. Comput.*, vo pp. 1197–1206, Nov. 1972.

[10] M. Takeda and J. W. Goodman, "Neural networks for comp Number representations and programming complexity," *App* vol. 25, pp. 3033–3046, Sept. 1986.

[11] T. Poggio, V. Torre, and C. Koch, "Computational vision gularization theory," *Nature*, vol. 317, pp. 314–319, Sept.

[12] J. P. LaSalle, *The Stability and Control of Discrete Processes* York: Springer-Verlag, 1986.

[13] N. Metropolis *et al.*, "Equations of state calculations by fa puting machines," *J. Chem. Phys.*, vol. 21, pp. 1087–1091.

[14] S. Kirkpatrick *et al.*, "Optimization by stimulated annealin *ence*, vol. 220, pp. 671–680, 1983.

[15] W. K. Pratt *et al.*, "Visual discrimination of stochastic fields," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-8, p 814, Nov. 1978.

[16] J. W. Woods, J. Biemond, and A. M. Tekalp, "Boundary val lem in image restoration," in *Proc. Int. Conf. Acoust., Spee nal Processing*, Tampa, FL, Mar. 1985, pp. 692–695.

[17] M. M. Sondhi, "The removal of spatially invariant degrad *Proc. IEEE*, vol. 60, pp. 842–853, July 1972.

[18] J. Biemond, J. Rieske, and J. Gerbrand, "A fast Kalman images degraded by both blur and noise," *IEEE Trans. Speech, Signal Processing*, vol. ASSP-31, pp. 1248–1256, Oc

[19] R. Chellappa and H. Jinchi, "A nonrecursive filter for edge

ing image restoration," in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, Tampa, FL, Mar. 1985, pp. 652–655.
[20] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, pp. 4–22, Apr. 1987.

Yi-Tong Zhou (S'84) received the B.S. degree in physics from the East China Normal University, Shanghai, China, and the M.S. degree in electrical engineering from the University of Southern California, Los Angeles, in 1982 and 1983, respectively.

He is currently a Research Assistant at the Signal and Image Processing Institute, University of Southern California, Los Angeles, and is working toward the Ph.D. degree in electrical engineering. His research interests include image processing, computer vision, neural network algorithms, optical computing, and biomedical signal processing. He has published about a dozen technical papers in these areas.

Rama Chellappa (S'78–M'79–SM'83), for a photograph and biography, see this issue, p. 1066.

Aseem Vaid was born in Jammu, India, on January 8, 1963. He received the B.Tech. degree in electrical engineering in May 1985 from the Indian Institute of Technology, New Delhi.

Currently he is working on the Ph.D. degree at the University of Southern California, Los Angeles. His research interests are in neural networks and optical computing.

B. Keith Jenkins (M'85) received the B.S. degree in applied physics from the California Institute of Technology, Pasadena, in 1977, and the M.S. and Ph.D. degrees in electrical engineering from the University of Southern California, Los Angeles, in 1979 and 1984, respectively.

He was employed at Hughes Aircraft Company, El Segundo, CA, from 1977 to 1979 where he worked on holography for use in head-up displays. From 1984 to 1987 he was a Research Assistant Professor in the Department of Electrical Engineering, University of Southern California, where he presently is Assistant Professor of Electrical Engineering. He has also participated in advisory panels to government and industry, and has been a consultant to JPL, TRW, and Odetics. His research has included work in the areas of optical digital computing, neural networks and their optical implementation, learning algorithms, optical interconnection networks, parallel computation models and complexity, computer-generated holography, and optical 3-D position sensing systems.

Dr. Jenkins is a member of the Optical Society of America and the Association for Computing Machinery. He was awarded The Northrop Assistant Professor of Engineering at USC in 1987, and is a recipient of the 1988 NSF Presidential Young Investigator Award.

To be presented at the Twenty Second Annual Asilomar Conference on Signals, Systems and Computers, Pacific Grove, California, November 1988.

Stochastic Learning Networks for Texture Segmentation*

B.S.Manjunath and R.Chellappa
University of Southern California
Department of EE-Systems
324, Powell Hall of Engineering
Los Angeles, California 90089
Telephone No.: (213)-743-8559.

May 24, 1988

Author for corresondence:
Prof. Rama Chellappa at the above address.

Abstract

In this paper we describe Neural Network based algorithms for segmentation of textured gray level images. We formulate the problem as one of minimizing an energy function, derived through the representation of textures as Markov Random Fields (MRF). We use the Gauss Markov Random Field (GMRF) to represent the texture intensities and an Ising model to characterize the label distribution. The resulting non-convex energy function is minimized using a Hopfield neural network. The solution obtained is a local optimum in general and may not be satisfactory in many cases. Although stochastic algorithms like simulated annealing have a potential of finding a global optimum, they are computationally expensive. We suggest an alternate approach based on the theory of learning automata which introduces stochastic learning into the iterations of the Hopfield network. A probabilty distribution over the possible label configurations is defined and the probabilities are updated depending on the final stable states reached by the neural network. The performance of this rule in classifying some real textured images is given. The results are similar to those obtained using simulated annealing but our algorithm needs fewer number of iterations.

*Partially supported by the AFSOR grant no. 86-0196.

SUMMARY

Texture segmentation is an important problem in computer vision as most of the real scenes consist of textures. Understanding such images is critical in many applications and fast algorithms to segment and classify images will be very useful in this context. Speed is an important criterion if these algorithms are to be implemented in real time for applications like robotic vision. The inherent parallelism of neural networks provides an interesting architecture for such problems and there have been some attempts in using neural networks for texture discrimination [5]. In this paper we model this problem as one of minimizing an energy function which is derived by modelling the texture as a Gauss Markov Random Field (GMRF) [2] and the texture label distribution using an Ising model [7].

The image is represented by an $M \times M$ intensity array and the individual sites are indexed by $s = (i, j)$. The intensity at site $s = (i, j)$ is denoted by y_s. We obtain an energy $U_1(s, k)$ relating the intensity at site s with a texture calss k by constructing a square window W_s centered at s and computing the negative of the likelihood of the pixels within that window. It is assumed that the region inside the window belongs to a single texture class k. The energy corresponding to the prior distribution of the class labels $U_2(s, l_s)$ where l_s denotes the label assigned to site s, is obtained using an Ising model. We are intersted in maximizing the posterior distribution of the texture classes given the intensity array. Finding an optimal solution requires an exhaustive search over possible label configurations which is practically impossible. It is well known that stochastic relaxation algorithms like simulated annealing [3] can find the global optimum if proper cooling schedules are followed but these algorithms are computationally very expensive. Approximate solutions can be obtained using deterministic relaxation techniques. In this paper we consider two algorithms , one based on minimizing an energy function using a Hopfield network and the other using stochastic learning and compare the results with that of simulated annealing.

Hopfield Network

A Neural Network for the classification of textures based on the above image model is described in this section . The neurons in the network are assumed to be binary and are arranged in a 3-D array . The elements are indexed by the subscripts (i, j, k) , $(1 \leq i, j \leq M, 1 \leq k \leq L)$,where $M \times M$ is the image size and L is the number of texture classes . Thus we have L layers each having M^2 neurons . The (i, j)-th neuron in layer k corresponds to the pixel site (i, j) taking the label k . A *column* of this network consists of the L neurons in the L layers for a site (i, j).

The connections are local and there are no inter-layer connections . Within any layer ,except at the boundaries , each neuron is connected to its 8 nearest neighbours. Since any pixel can have only one label , one neuron should be active in each column of the network . A simple *winner-*

takes-all circuit can be used for each column to select only one of the L neurons to represent the label for the corresponding site . Another alternative is to introduce a constraint in the energy equation for the network as in Hopfield and Tank [4]. If the k-th neuron in a column is active it means that the corresponding site has the label k.

The Energy function to be minimized can be written as

$$E = \frac{A}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{L}(U_1((i,j),k) - w(L(k)))V_{ijk} - \frac{\beta}{2}\sum_{k=1}^{L}\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{(i',j')\in N_{ij}} V_{i'j'k}V_{ijk} \qquad (1)$$

where N_{ij} denotes the neighbourhood of site (i,j) , V_{ijk} is the output of neuron at site (i,j) in the k-th layer , and A is a constant. The standard Hopfield energy equation is [4]

$$E = -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{L}\sum_{i'=1}^{M}\sum_{j'=1}^{M}\sum_{k'=1}^{L} T_{ijk;i'j'k'}V_{ijk}V_{i'j'k'} - \frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M}\sum_{k=1}^{L} I_{ijk}V_{ijk} \qquad (2)$$

From (1) and (2) we can identify the parameters of the network as

$$T_{ijk;i'j'k'} = \begin{cases} \beta & \text{if } (i',j') \in N_{ij}, \forall k \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

and the bias current

$$I_{ijk} = -A(U_1((i,j),k) - w(L(k)))$$

The input-output relation can be stated as follows . Let u_{ijk} be the potential of neuron (i,j,k).(Note :k is the layer number) , then

$$u_{ijk} = \sum_{i'=1}^{M}\sum_{j'=1}^{M}\sum_{k=1}^{L} T_{ijk;i'j'k'}V_{i'j'k'} + I_{ijk} \qquad (4)$$

and

$$V_{ijk} = \begin{cases} 1 & \text{if } u_{ijk} = \max_l\{u_{ijl}\} \\ 0 & \text{otherwise} \end{cases} \qquad (5)$$

Convergence : In (3) we have no self feedback ,i.e. $T_{ijk;ijk} = 0, \forall i,j,k$ and all the connections have equal strengths . The updating scheme ensures that at each stage the energy decreases . Since the energy is bounded, the convergence of the above system is assured but the stable state will in general be a local optimum .

This neural model is one version of the Iterated Conditional Mode algorithm (ICM) of Besag [1] , which maximizes the conditional proability of the labels given the intensity array and the labels of the neighbouring pixels at each iteration. ICM is a local deterministic relaxation

algorithm and very easy to implement. We observe that in general any algorithm based on MRF models can be easily mapped on to Hopfield type Neural networks with local interconnections. However this algorithm is very sensitive to the initial configuration and the solutions obtained are not satisfactory in general. In the next section we consider an alternate scheme which combines stochastic learning and deterministic relaxation and because of its stochastic decision making is not sensitive to the initial states.

Stochastic learning in neural networks :

We now describe an algorithm which introduces stochastic learning into the neural network model for texture discrimination. This is motivated from the theory of learning automata [6]. It consists of a two stage process with learning and relaxation alternating with each other and because of its stochastic nature has the potential of escaping the local minima.

The learning part of the system consists of a team of automata $\{A_s\}$, one automaton for each pixel site. Each automaton A_s at site s maintains a time varying probability vector $\mathbf{p_s} = [p_{s_1}, \ldots, p_{s_L}]$ where p_{s_k} is the probability of assigning the texture class k to the pixel site s. Initially all these probabilities are equal. At the beginning of each cycle the learning system will choose a label configuration based on this probability distribution and present it to the Hopfield neural network described above as an initial state. The neural network will then converge to a stable state. The probabilities for the labels in the stable configuration are increased according to the following updating rule : Let k_s be the label selected for the site $s = (i, j)$ in the stable state in the n-th cycle. Let $\lambda(n)$ denote a reinforcement signal received by the learning system in that cycle.then,

$$\begin{aligned} p_{s_{k_s}}(n+1) &= p_{s_{k_s}}(n) + a\lambda(n)[1 - p_{s_{k_s}}] \\ p_{s_j}(n) &= p_{s_j}(n)[1 - a\lambda(n)] \;,\; \forall j \neq k_s \end{aligned} \tag{6}$$

for all $s = (i, j), 1 \leq i, j \leq M$.

In the above equation 'a' determines the learning rate of the system. The reinforcement signal determines whether the new state is good compared to the previous one in terms of the energy function. Using the new probabilities, a new initial state is generated randomly for the relaxation network and the process repeats. The above learning rule is called Linear Reward-Inaction rule in the learning automata terminology.

Experimental results and conclusions :

We have tested the above algorithms in classifying some real textured images. The parameter values for the different texture classes were precomputed and stored in a library. These are used in calculating the different energy functions. The bias values $w(.)$ are choosen by trial and error but

they can also be estimated from the image data. We have experimented with different β ranging from 0.3 to 2.0 and also with β depending on the order of the neighborhood. We have used images consisting of two and six texture classes and the results for the six texture class problem are shown in figure 1. The Hopfield network solution has a misclassification of about 14% when started with a random configuration. We observed that random initial states give better results compared to the ones starting from Maximum likelihood estimates [2]. The learning scheme has an error of about 6.8% compared to 6.3% obtained using simulated annealing, but the number of iterations were considerably more in the case of simulated annealing. In general stochastic algorithms seem to perform better than any deterministic sheme.

Currently we are working on extending these methods to do hierarchical segmentation and the initial results are quite promising.

References :

1. Besag ,J ," On the Statistical Analysis of Dirty Pictures" , *Journal of Royal Statistical Society* B,vol. 48 No. 6,pp 259-302, 1986.

2. Chellappa,R. and Chatterjee, S., " Classification of Textures Using Gauss Markov Random Fields", *IEEE Trans. Acous., Speech, Signal Process.*, vol. ASSP-33,no. 4, pp. 959-963, August 1985.

3. Geman,S and Geman,D. ," Stochastic Relaxation , Gibbs Distributions, and Bayesian Restoration of Images ", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.6 ,pp. 721-741, November 1984.

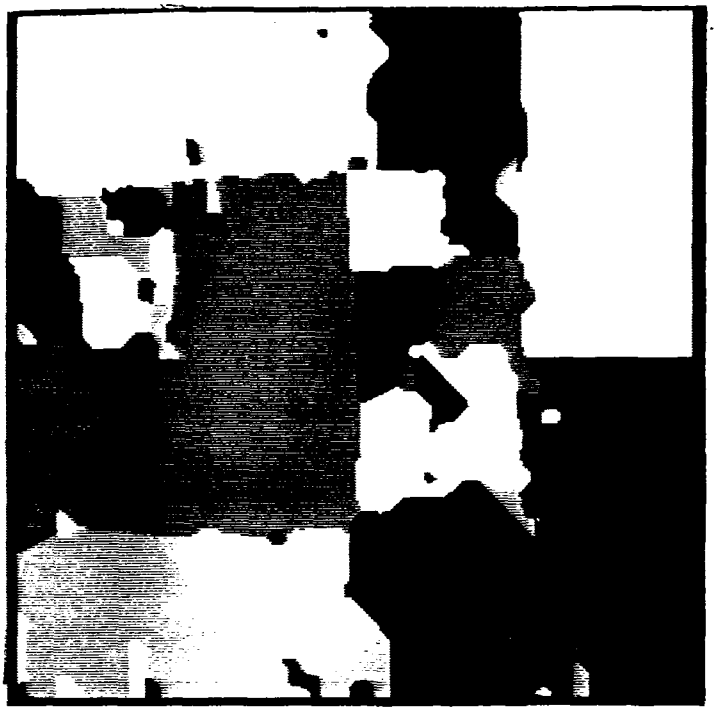
4. Hopfield,J.J., and Tank,D.W.," Neural Computation of Decisions in optimization Problems",*Biological Cybernetics*,vol. 52,pp. 114-122, 1985.

5. Mesrobian,E and Skrzypek,J. ," Discrimination of Natural Textures: A Neural Network Architecture ",in *Int. Conf. on neural networks* , vol.4, pp. 247-258, San Diego ,July 1987.

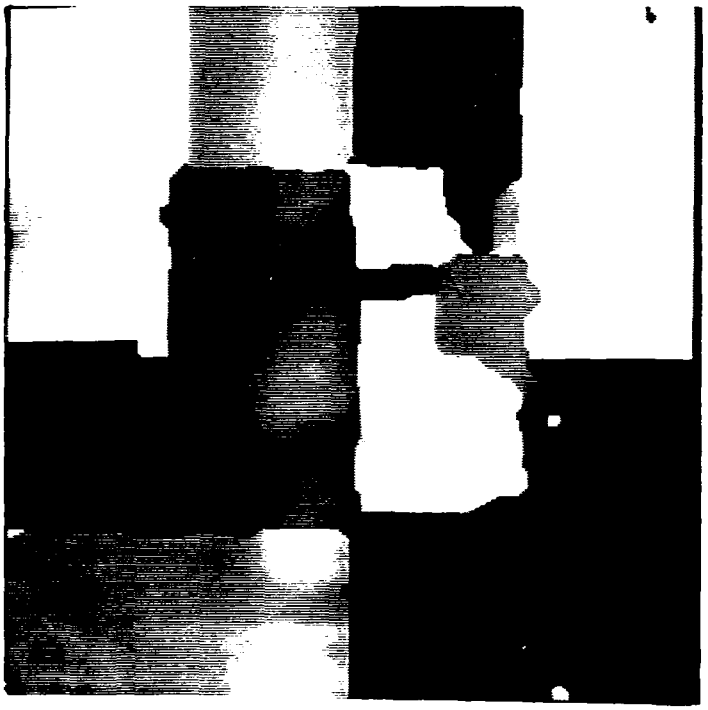
6. Narendra,K.S. and Thathachar,M.L.A.," Learning automata - A survey," *IEEE Trans., Syst.,Man, Cybern.*, vol. SMC-4,pp. 323 -334, 1974.

7. T.Simchony and R.Chellappa, " Stochastic and Deterministic Algorithms for Texture Segmentation " *Proc. of International Conf. Accous., Speech and Signal Process* ,pp 1120-1128 , NewYork, April 1988.

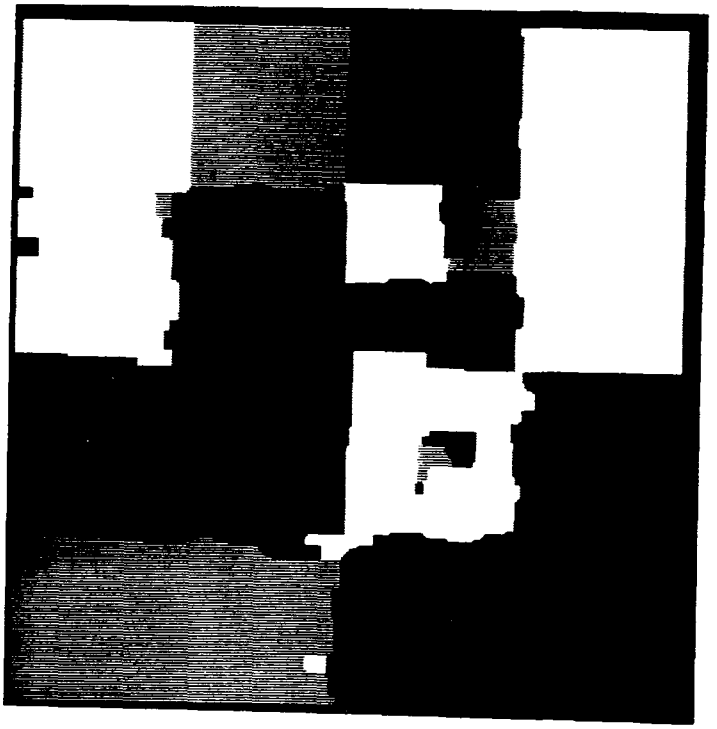
(A): Original Image



(B):Hopfield Network



(C):Stochastic Learning



(D):Simulated Annealing

Figure 1: Experimental results for a six class segmentation problem