AD-A204 924

UMENTATION PAGE

Form Approved
OMB No. 0704-0188

1b. RESTRICTIVE MARKINGS
None

2a. SECURITY CLASSIFICATION AUTHORITY

3. DISTRIBUTION/AVAILABILITY OF REPORT
Unlimited

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE

4. PERFORMING ORGANIZATION REPORT NUMBER(S)

5. MONITORING ORGANIZATION REPORT NUMBER(S)
AFOSR-TR- 89-0112

6a. NAME OF PERFORMING ORGANIZATION
Massachusetts Inst. of Tech.

6b. OFFICE SYMBOL (If applicable)

7a. NAME OF MONITORING ORGANIZATION
Air Force Office of Scientific Research

6c. ADDRESS (City, State, and ZIP Code)
77 Massachusetts Avenue
Cambridge, MA 02139

7b. ADDRESS (City, State, and ZIP Code)
Bolling AFB, BLDG 410
DC 20332

DTIC ELECTE 1989

8a. NAME OF FUNDING/SPONSORING ORGANIZATION
~~DARPA~~ Same as 7a

8b. OFFICE SYMBOL (If applicable)
NE

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
AFOSR-86-0301

8c. ADDRESS (City, State, and ZIP Code)
1400 Wilson Blvd.
Arlington, VA 22209 Same as 7b

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
DARPA	5780	00	

11. TITLE (Include Security Classification)
Optical Inference Machines

12. PERSONAL AUTHOR(S)
James Kottas, Cardinal Warde

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
INTERIM Annual	FROM 15/8/87 TO 14/8/88	December 30, 1988	38

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Symbolic inference optical inference engines, optical information processing, spatial light modulators, neural networks, interconnect weights, chaos.

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

We have been investigating an optical inference machine based on neural network principles. This approach represents a diversion from the mapped-template architecture (C. Warde and J. A. Kottas, Appl. Opt. 25, 940, 1986) to overcome the latter system's limitations. During the second year of this grant, we designed the Infernet, a neural network architecture which we believe can solve symbolic inference problems, and a corresponding optical implementation. This report describes our new system from an architectural viewpoint. We also describe an automated two-dimensional photocalibration system, developed for our computerized control system, that will provide fast analog intensity measurements on images as well as calibration of image registration from plane-to-plane in the optical processor.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL

22b. TELEPHONE (Include Area Code)
(202) 767-4933

22c. OFFICE SYMBOL
NP

DD Form 1473, JUN 86 Previous editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

OPTICAL INFERENCE MACHINES

Interim Report

AFOSR-86-0301

December 19, 1988

James Kottas
Cardinal Warde

Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
Bolling Air Force Base, D. C. 20332

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

Symbolic logic problems involve, in an abstract sense, a set of data objects (also known as symbols) and a set of relationships describing the data objects. The data objects and relationships constitute a knowledge base which is generally arranged as sets of facts and rules. A fact is a statement connecting a relationship with one or more data objects such that the statement is always interpreted as being true. On the other hand, a rule is a statement which defines a relationship using other relationships, data objects, and/or facts.

A symbolic logic problem is usually stated in the form of one or more queries which are questions concerning the relationships and the data objects. The queries are answered by applying logical inference to the knowledge base of rules and facts. This inference process generates a set of assertions (inferred facts) from the knowledge base. The solution to the queries therefore becomes a set of conclusions in the form of data objects which is inferred from the set of assertions so as to satisfy the queries. Symbolic logic problems are relatively common. They arise in areas such as pattern recognition, expert systems, and other artificial intelligence systems.

The general structure of an inference machine is shown in Fig. 1. It accepts as inputs a set of facts and a set of rules from the knowledge base, and one or more queries. The output of the inference machine is a set of specific conclusions which are logically inferred from the facts and rules in response to the queries.

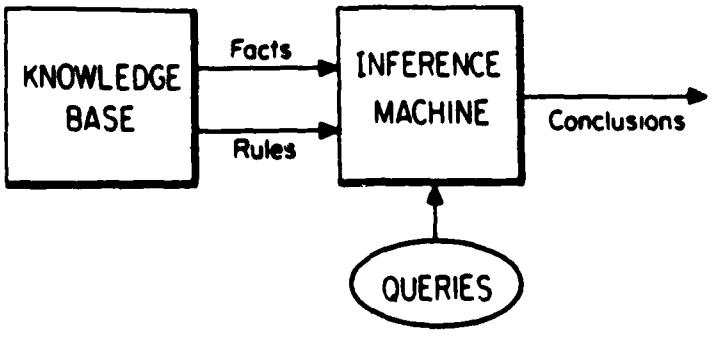


Fig. 1 General structure of an inference machine.

During the first year of the program, we designed and started to investigate our mapped-template architecture for solving symbolic inference problems [1,2]. This approach focused on the use of optical information processing techniques to quickly search the knowledge base. However, we found that some of the limitations of the mapped-template system which could be overcome by modifying the architecture. First, the mapped-template architecture is inherently not fault tolerant. That is, if any optical obstructions (such as large dust particles) became lodged on the lenses, mirrors, or spatial light modulators, they could block the light beam corresponding to a particular data object and thus cause large errors. Second, the number of usable data objects was limited by the size of the spatial light modulators. Since current research devices have only modest space-bandwidth products, the number of data objects that could be represented was quite limited. Third, the mapped-template concept implements deductive logic, making it necessary to program the knowledge base into the system. This makes the system inflexible to assimilating new information (a problem shared by conventional expert systems). Finally, since the knowledge base is stored electronically, the architecture is very dependent upon a digital computer for manipulating the knowledge base. This necessarily involves some searching, and so although the mapped-template approach reduces the amount of electronic searching it does not completely eliminate it.

Because of these limitations, we decided to consider the use of neural network models to realize an optical inference machine architecture. Neural networks offer possible solutions to the above limitations. They offer a significant amount of fault tolerance. While the size of any neural network in an optical implementation is constrained by the spatial light modulators, the size of the knowledge base is ultimately limited by the number of interconnects. If holographic techniques are used to implement the interconnects, then a large number can be achieved in principle. With the knowledge base stored optically, the electronic computer is only responsible for controlling the optical system and interacting with the user (and so the systems we developed during the first year of the program are still necessary). Thus, the computer does not have to directly manipulate the knowledge base. Furthermore, neural network models offer the flexibility of learning new information via inductive reasoning, a feature not shared by the deductive mapped-template architecture.

We retain the belief that optical inference machines should be a hybrid composition of optical and electronic systems. Although electronics is a well developed technology, it is expensive to perform parallel processing, a natural ability of optics. During the first year of the program, we developed a computerized control system (CCS) for controlling the optical devices and acquiring electronic measurements. However, this system lacked an efficient method for electronically measuring a two-dimensional light distribution. Consequently, during this second year, we developed an automated two-dimensional photocalibration system for our CCS. It offers flexibility of alignment and fast, convenient operation.

APPROACH

In general, an inference machine can be thought of as a fancy signal processing box with inputs and outputs. Everything outside of the box is considered the "real world" or its environment. The queries to the system are signals that originate in the real world and enter the inference machine through its inputs. In the same way, the conclusions are signals that enter the real world via the system's output. This structure allows feedback in the real world so past conclusions may affect future queries.

When the machine is constructed using neural network concepts, the internal region of the box can be considered as the "neural world." For a neural-network inference machine to be practical, it should have the following functional characteristics:

1. The operational goal of the system is to infer approximate conclusions to its inputs and generate outputs quickly and efficiently. "Approximate" conclusions are appropriate here because neural networks are known to have good generalization properties but are weaker at detailed analysis. The tradeoff is the gain of fault tolerance and speed at the expense of accuracy. If a more exact analysis of the situation is desired, a digital computer could be employed to further any analyses.

2. The inputs can convey complete or incomplete analog information in that there may not be a unique mathematical representation for a particular input that is feasible to generate or even consider. An example problem for which this type of information is typical is pattern recognition. A mathematical description of the desired pattern cannot be known conveniently, but a child can easily recognize the pattern without this information.

3. The system should be an asynchronous continuously-driven system as opposed to being solely an input-driven or "query-driven" system in the terminology of Ref. 1. That is, its inference processing should be able to proceed in the absence of real-world inputs. Similarly, all inputs should be treated as asynchronous functions of time.

4. The system should be capable of anticipating or expecting future inputs given the past history of the inputs and the information it has accumulated.

Finally, since several inference processes involve causality, it would be useful to endow these systems with this time-domain feature. Current neural network models do not use the temporal dimension effectively for processing.[1]

[1] This is in contrast to using neural networks to process time signals, in which case, the network is trained on sample time signals to implement a learned one-pass transformation. As a result, one of the challenges to using neural network models in creating an inference machine is how to use their temporal dynamics for processing.

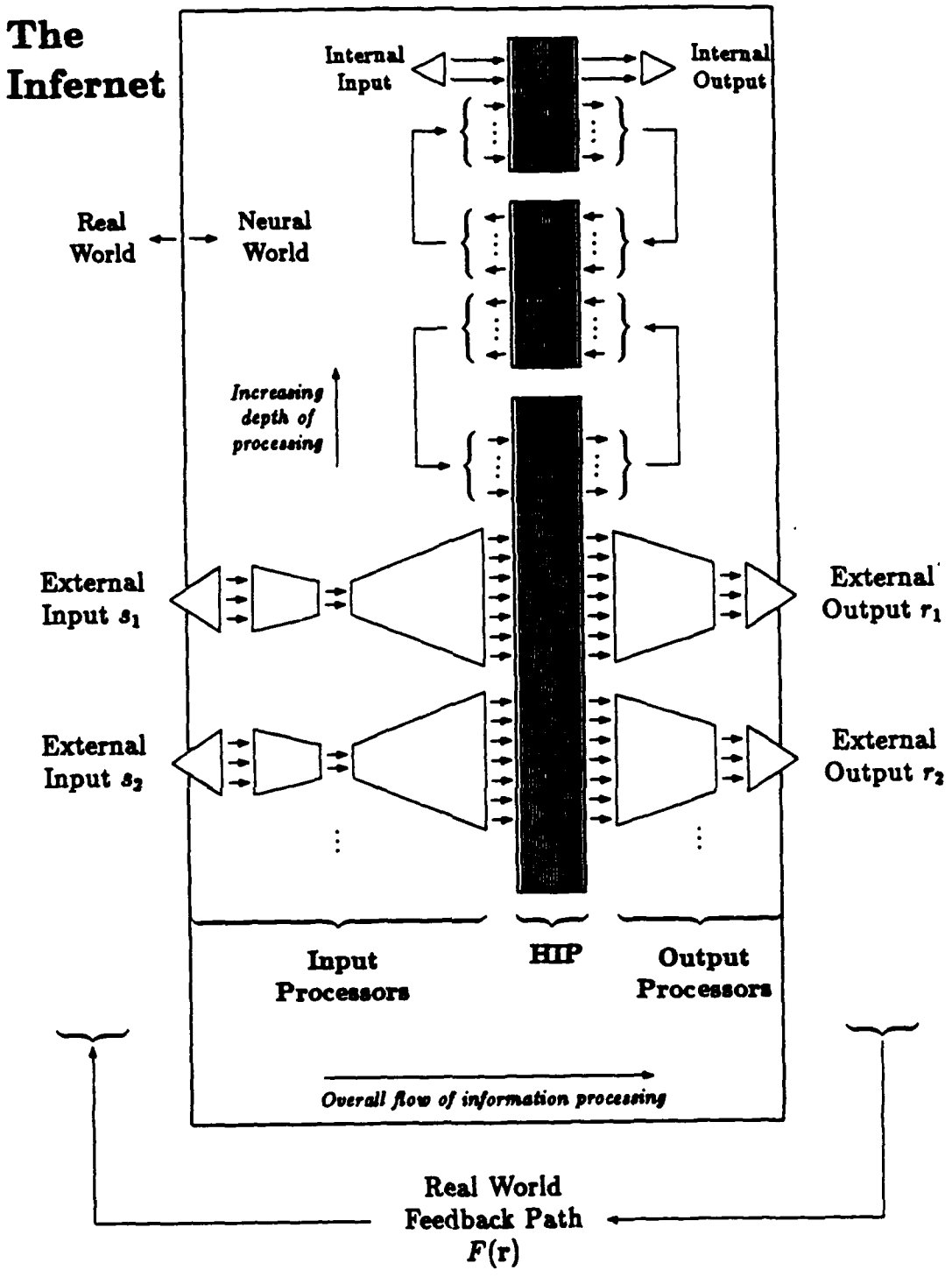


Fig. 2 Block diagram of the Infernet. Triangles represent input/output neural layers, trapezoids represent input/output processing layers, and rectangles represent inference processing layers. The number of arrows between two layers gives a relative indication of the number of connections.

THE INFERNET SYSTEM

With the above considerations in mind, we have conceived the Infernet (for INFERence NETwork), an architecture designed using neural layers as the fundamental building block. A block diagram of the Infernet is shown in Fig. 2. The architecture can be grouped into three sections: the input processors, the hierarchical inference processor (HIP), and the output processors. The main flow of information processing occurs from left to right in this figure. Each processor with each section is composed of several neural network layers (represented by trapezoids in Fig. 2, with the lengths of the left and right sides of a trapezoid indicating its relative number of inputs and outputs, respectively). To understand the processing concept of the Infernet, it is more convenient to begin with the input/output structures and then proceed to the HIP.

The Infernet interacts with the real world via N_{in} external inputs and N_{out} external outputs. The input processors are responsible for sensing and analyzing the external input, s_i, $i=1,2,...,N_{in}$, to produce a vector of neural signals x_i for the HIP. For external inputs, there must be corresponding input processors.

In general, each input processor consists of three layers, as shown in Fig. 3. The first layer (shown as a left-pointing triangle on the left side in Figs. 2 and 3) senses the external stimuli relevant for the particular external input and produces a representative set of neural signals. In effect, this first layer performs an input format conversion between the real world and the neural world of the Infernet. The second layer decomposes the neural signals into a basis set of information building blocks on which all analyses can be performed.[2] As this step usually can involve a reduction in the volume of input data, the output side of the second layer is shown in Fig. 3 as being smaller than the input side. In an actual implementation, the first two layers may be combined because data reduction may not be necessary for all input stimuli of interest.

The third layer processes the basis set by analyzing various collective features to expand the information available. Since several different analyses may be performed by this third layer, the output side is shown larger than the input side. The output of the third layer, x_i, is the output of the i^{th} input processor, which corresponds to the external input s_i. The combined outputs x_i, $i=1,2,...,N_{in}$ comprise all of the inputs to the HIP that originate in the real world.

For all three layers, the actual processing operations are determined by the topology of the interconnections and their associated weights. When the weights are allowed to vary, the input processors can adapt to new reductions/analyses with time.

---

[2] An example of the basis set of information building blocks is the on-center/off-surround (and off-center/on-surround) receptive fields in the human visual system.

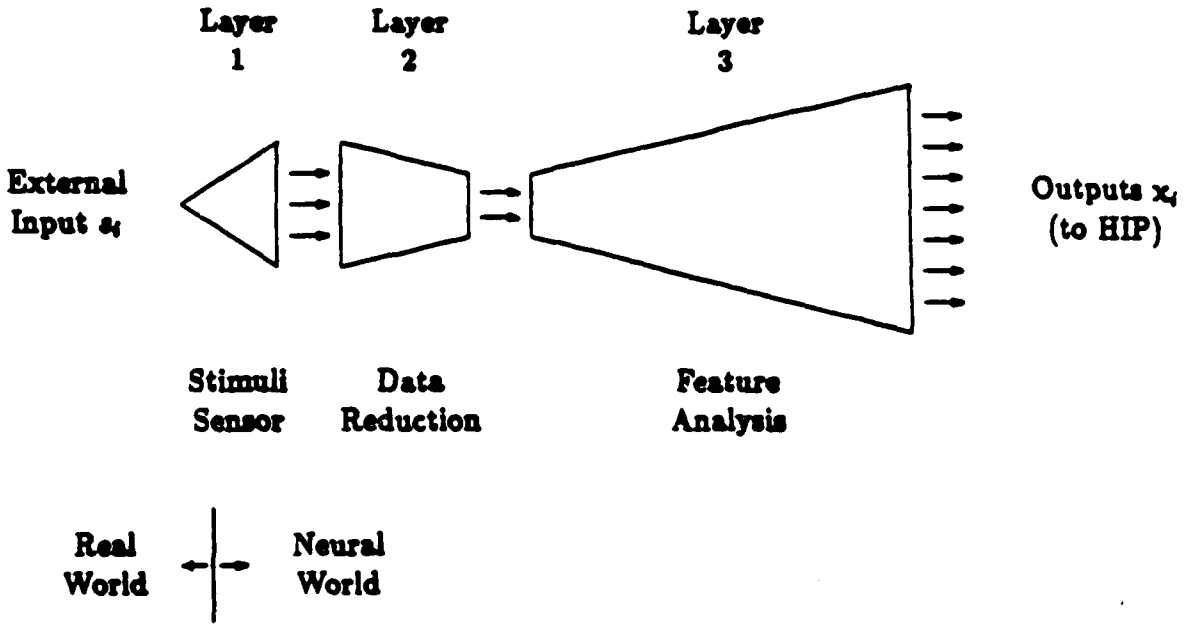


Fig. 3 An input processor for the Infernet

The output processors perform the complementary operation of the input processors. They convert a portion of the neural output signals y_i,i= 1,2,...,N_{out}, from the HIP into action responses r_i for the real world. For N_{out} external outputs, there must be the same number of output processors. As shown in Fig. 4, each output processor consists of two layers.

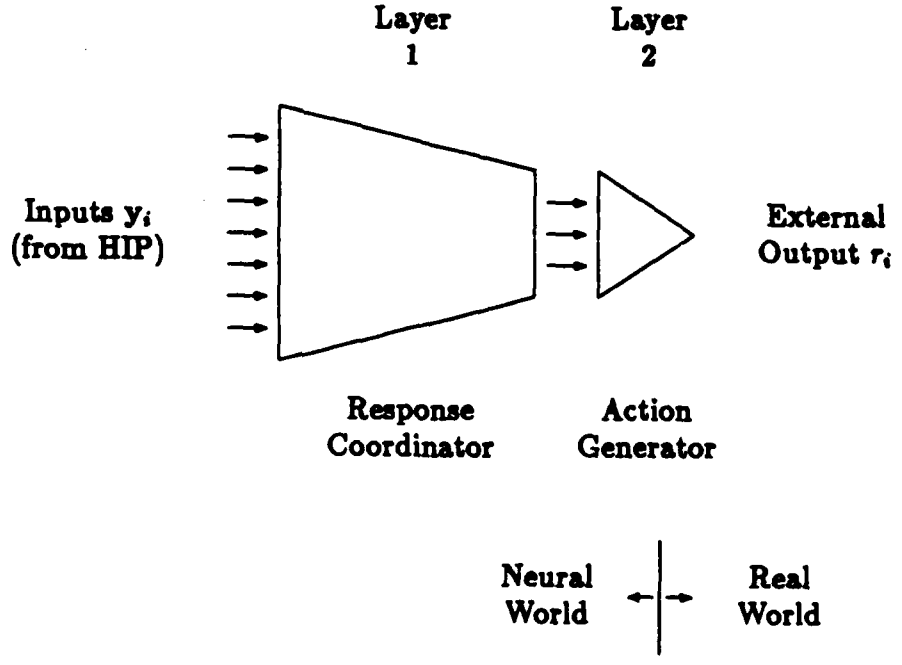


Fig. 4. An output processor in the Infernet

The first layer transforms the HIP signals y_i into the basis set of coordinated response signals for the real world. The second layer actually generates the real-world actions r_i corresponding to these response signals. As with an input processor, the actual reduction operations and output transformations are determined by the interconnection topology and strengths within the two layers. When the strengths are allowed to vary, an output processor can adapt to produce new or improved external output actions with time.

As indicated in Fig. 2, the external outputs possibly can influence the external inputs via feedback in the real world (represented by the feedback function F(r) where r is the vector of external outputs r_i). It is believed that this feedback path is very important in the learning process, particularly for inductive learning. One of the tasks of this program is to investigate the effects of real-world feedback on the ability of the Infernet to assimilate new knowledge.

Functionally, both the input and output processors are input-driven structures in that they produce no output signals in the absence of the appropriate inputs (or stimuli). In this sense, they represent static processing sections.
The HIP, on the other hand, is a dynamic processing section. Connecting the output processors with the input processors, the HIP is the inference engine of the Infernet and thus contains the knowledge base. Because of its dynamical nature, the HIP operates continuously to allow for further *inferring and/or processing on* the knowledge it has stored. This ability permits the Infernet to continue processing inputs even after the driving stimuli have been removed and the input processors have become silent.

An expanded block diagram of the HIP is shown in Fig. 5. Since the HIP is composed of neural layers, the information contained in the knowledge base is stored in a distributed manner according to the topology of the interconnections between the neurons and their associated synaptic strengths. Consequently, Fig. 5 does not show any explicit memory structures (i.e. layers whose sole function is information storage) within the HIP.

The HIP consists of several levels of neural layers arranged in a linear fashion such that any particular level only can affect its two neighboring levels. The configuration shown in Fig. 5 depicts three levels in a vertical format. The lower level of this chain, labeled "Level 1", accepts the processed input signals x_i, $i=1,2,...,N_{in}$, from the input processors and also receives the signal u_{21} from the next higher level, labeled "Level 2". The first level processes these inputs by mixing them using several neural layers. The resulting outputs form the second level signal, u_{12}, and the response signals y_i, $i=1,2,...,N_{out}$, which are sent to the output processors.

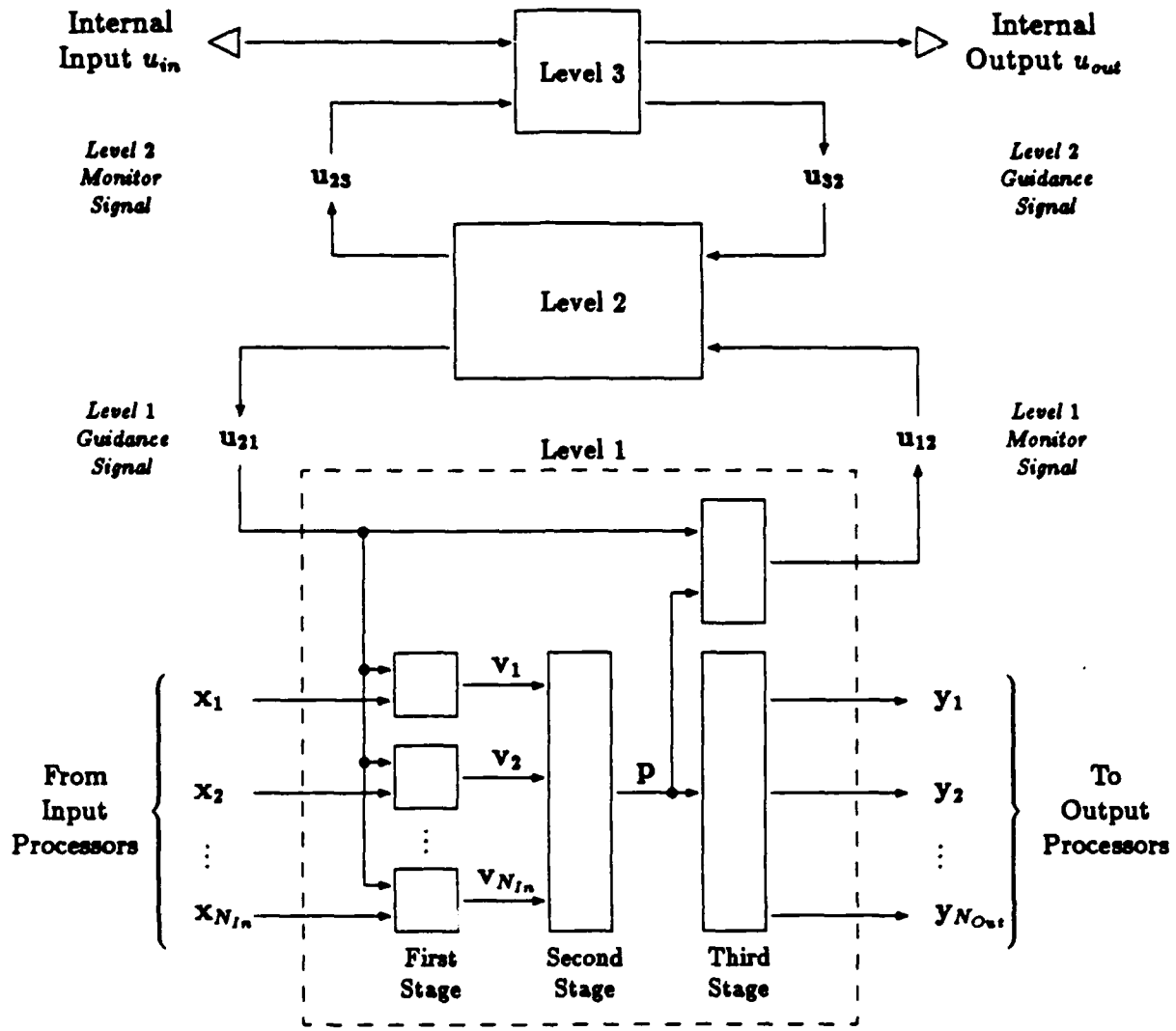


Fig. 5 **An expanded block diagram of the hierarchical inference processor (HIP) of the Infernet. In this figure, each connection arrow represents a vector of neural signals and each rectangle corresponds to a neural layer. Thus, Level 1 is composed of several layers while Levels 2 and 3 consist of single layers. The triangles for the internal input and output signals indicate that they can undergo a transformation to/from the neural world of the HIP, depending upon their desired form. Note that the structure has no explicit memory blocks or components; knowledge is stored distributively throughout the HIP levels.**

As envisioned, the first level in this chain serves as the main decision-maker of the Infernet, and for this role it receives a guidance signal u_{21} from the second level. In this context, u_{21} can be viewed as an expectation signal which can be used to set the state of the first level to anticipate certain external inputs. The feedback signal u_{12} informs the second level as to the decisions made by the first level. In this way, the second level acts like an advisor for the first level and, in an hierarchical sense, can be considered above the first level. In terms of

expectation, u_{12} can be used to determine what the second level should anticipate from the third level. Another benefit of the $u_{12} \rightarrow u_{21}$ feedback path is that it provides a mechanism for the HIP to perform further processing on the stored information.

The idea of hierarchical advising now can be applied to the second level. In addition to the first level, this level interacts with a third level via the monitoring signal u_{23} and the guidance signal u_{32}. However, signals between Levels 1 and 3 are not permitted because they would eliminate the hierarchical structure of the HIP. This arrangement would allow a "fully interconnected" topology between the levels and, thus, would undermine the interaction between the various levels. Since this interaction is believed to be important in creating a neural network inference engine, such interlayer connections, between nonadjacent levels in the HIP, are not considered here. For a general HIP, the advising process can be extended to include many more levels. However, the Infernet system shown here consists of only three levels in the HIP to study the general inference-making properties of such an architecture.[3]

Without a monitoring level with which to interact, the third level has the potential to be affected by some other source. As shown in Figs. 2 and 5, this guidance signal is called the internal input u_{in}; and correspondingly, the third-level monitor signal is the internal output u_{out}. The "internal" aspect of the guidance signal refers to the fact that its influence originates at the upper level of the HIP and filters down to the lowest level. One possible use for u_{in} is to provide a form of external supervision when training the Infernet. In the same way, the internal monitor signal u_{out} provides an indication of the activity of the highest level in the HIP. Depending upon the their usage, the internal input and output signals may be desired in a real-world format, which is illustrated in Figs. 2 and 5 by the small left- and right-pointing triangles, respectively.

Although the first level remains the primary decision-making layer, it is believed that the entire hierarchical structure is necessary to make the HIP an inference engine. The first level receives guidance inputs directly from the second level and indirectly from the third level. Thus, to a certain extent, all levels are involved in the decision-making process. It is this interlevel advising in conjunction with the anticipatory guidance signals that is believed to give the HIP architecture its inference and expectation capabilities, properties which must be validated during the remainder of this research program.

As shown in Fig. 5, the second and third levels each are composed of a single neural layer whose sole function is to combine the monitor signal from the lower level and the guidance signal from the upper level to form a guidance signal for the lower level and a monitor signal for the upper level. The exact method by which these two layers perform this task depends

---

[3] In addition, a "fully interconnected" network places large demands upon the implementation technology.

upon their intralayer connection schemes.

In contrast, the first level is composed of several smaller layers (or sublayers) for combining the outputs of the input processors with the second level guidance signal. One possible arrangement for these sublayers is shown in Fig. 5. The concept of this arrangement is to compare and correlate the external inputs with the expectations of the upper levels of the HIP and then decide on an action based on this comparison. Lastly, the upper levels can be updated as to any decisions made, while the output processors generate the corresponding real-world responses.

Based on this concept, the topology of the first level sublayers consists of three stages. The first stage compares the external inputs with the upper level HIP signals by individually mixing the input signals x_i with the guidance signal u_{21} using N_{in} sublayers to produce N_{in} influence signals v_i (see Fig. 5). The second stage consists of a single sublayer which generates a decision signal **P** based on the influence signals. Thus, the second stage has most of the decision-making power of the first level. Finally, the decision signal **P** is used to create a new monitor signal u_{12} using the old guidance signal u_{21} in one of two sublayers comprising the third stage. The other sublayer decodes the decision signal **P** into the corresponding action signals y_i so the output processor can produce the appropriate real-world responses.

DYNAMIC KNOWLEDGE BASE ENCODING

As previously stated, the knowledge base is stored in the HIP; and since the HIP is a neural network, the actual storage format is a spatial distribution of synaptic weights. As with the input and output processing stages, the interconnection strengths within the HIP layers can vary with time, thus allowing the knowledge base to be altered and expanded with time.

However, there is a significant difference between a knowledge base that a human can understand and a spatial distribution of numbers. As a result, a formidable problem exists: how to determine the spatial distribution of interconnection weights given a human-understandable knowledge base. To address this problem, a solution which relies on the dynamic nature of the HIP is proposed and shall be referred to as dynamic knowledge base encoding. This idea represents an attempt to use the temporal dimension of a neural network for processing.

In this scheme, a knowledge base consisting of data objects, relationships, facts, and/or rules is recognizable as temporal "waves" of neuron activity cycling through the levels of the HIP. Each piece of knowledge (data objects, relationships, etc.) is stored distributively over localized regions of the HIP. Many waves can exist simultaneously. Associations within the knowledge base are made "visible" by monitoring the trajectory of a wave within the HIP.

The role of the interconnection coefficients becomes one of indirectly storing the knowledge base. Rather, they represent the degree to which a wave along one trajectory can be redirected into another path, indicating that different information is being addressed.

The inference process becomes the sequence of transitions from one wave trajectory to another, and so on until a conclusion is generated. In the case of *deductive inference*, a wave is simply redirected into other regions of the HIP whereby it either can stimulate new waves or pass through a region which is the correct response to the inputs. On the other hand, *inductive inference* becomes one of creating access to previously unaccessible regions of the HIP and associating particular pieces of knowledge with those regions. The interaction of various waves within the HIP is illustrated in Fig. 6.

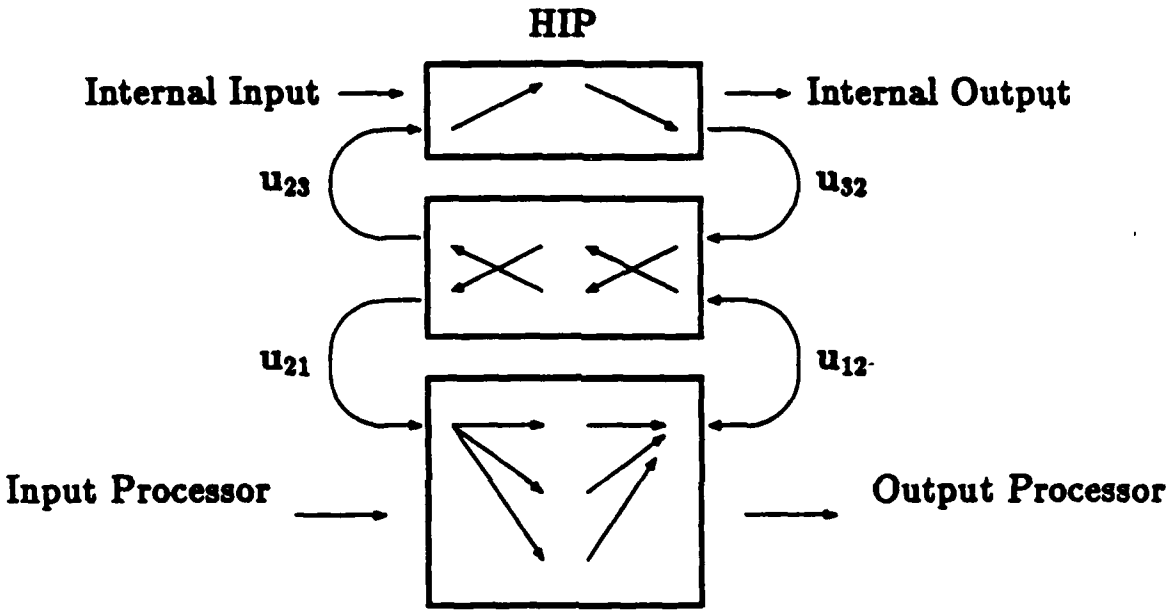


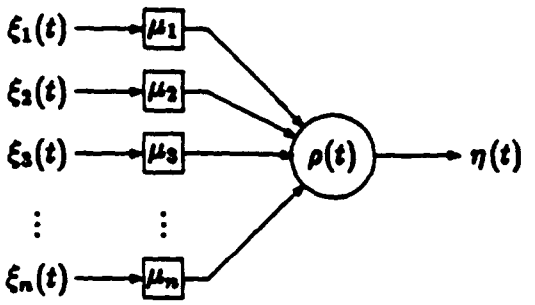
Fig. 6 Neural wave interaction within the HIP.

Note that if time could be stopped or frozen, the knowledge base would be "concealed" in that the past trajectory of a wave would not be visible nor could its future path be determined easily. Thus, current and future pieces of knowledge are "hidden" from view, although they do not go away because they are maintained by the interconnection distribution.

The idea of dynamic knowledge base encoding and neural waves raises several important issues, such as the interconnection topology and initial weight distribution, the wave time scale, and the relationship, if any, between neural waves.

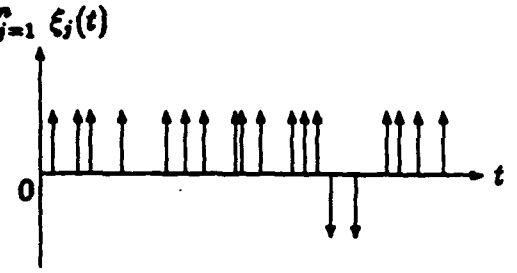
NEURAL CELLS

Although the Infernet is designed using neural layers as the basic building blocks, the neuron or cell is the fundamental processing unit in a neural network, the general structure of which is shown in Fig. 7(a).



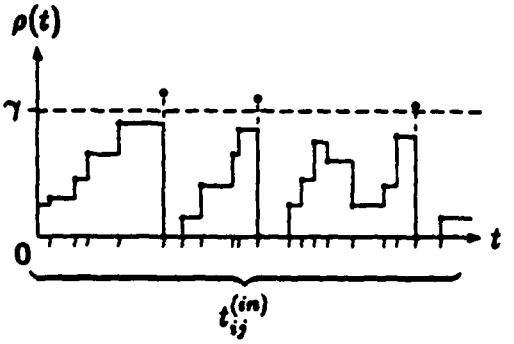
(a)

$$\xi_j(t) = \pm \sum_{i=1}^{n_j} \alpha \delta\left(t - t_{ij}^{(in)}\right)$$



(b)

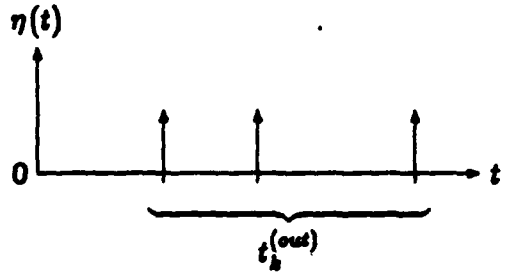
$$\rho(t) = \int_{\text{last pulse}}^{t} \sum_{j=1}^{n} \mu_j(t') \xi_j(t')\, dt'$$



(c)

$$\eta(t) = \pm \sum_{k=1}^{n} \alpha \delta\left(t - t_k^{(out)}\right)$$

$$= \pm \alpha \frac{d}{dt} \operatorname{int}\left[\frac{1}{\gamma}\rho(t)\right]$$



(d)

Pulse condition: $t_k^{(out)} = t_{ij}^{(in)}$ when $\rho\left(t_{ij}^{(in)}\right) \geq \gamma$

Fig. 7 The relationship between the neuron inputs $\xi_j(t)$, the neuron state $\rho(t)$, and the neuron output $\eta(t)$ in the pulse neuron model. The structure of the cell is shown in (a), the set of inputs overlaid on a common time axis in (b), the state in (c), and the resulting output sequence in (d). The state is shown for the case when $\rho(t)$ is reset to zero when an output impulse is generated. Note that the jumps in the state correspond to the input impulses and are scaled by the interconnection coefficients $\mu_j(t)$.

A cell has n inputs $\zeta_j(t)$, n interconnection weights μ_j, a state variable $\rho(t)$, and an output $\eta(t)$ which is derived from $\rho(t)$. To capture the dynamic nature of the HIP, we decided to first investigate pulse neurons rather than continuous-value cells. In a pulse neuron, the output is a sequence of identical pulses. The information being transmitted is contained in the timing relationships between the pulses. In contrast, continuous-value neurons have continuous outputs that can have any analog value within a given range (usually from either -1 or 0 to 1).

Pulse neurons have several advantages over continuous-value neurons. Analogous to frequency-modulation communication, pulse neurons are more tolerant of noise inherent in the network. Similarly, small variations in the actual pulse shape (amplitude and duration) do not affect the transmission of information. Neural waves would be observed as regions of increased firing frequency which propagate through the network.

The output of an ideal pulse neuron can be described by an impulse train in the form

$$\eta^{\pm}(t) = \pm \sum_{i=1}^{\kappa} \alpha\delta(t - t_i) \tag{1}$$

where t_i is the time at which the i^{th} impulse occurs, κ is the total number of impulses generated, and α is the magnitude of each pulse. The sign of the impulse train depends upon the type of connection that the neuron makes with other cells.

It is assumed that a cell can make only one type of connection with other neurons (either excitatory or inhibitory connections, but not both). As a result, the + sign is used in Eq. (1) for an excitatory neuron (i.e. one that makes only excitatory connections) and the - sign is for inhibitory neurons. For simplicity, the terms "excitatory neuron" and "inhibitory neuron" will be used to refer to neurons that make only excitatory or inhibitory connections, respectively. Note, however, that a neuron can have both excitatory and inhibitory inputs; it is only the output that is constrained to be a certain type.

The inputs to a cell can be described by

$$\xi_j^{\pm}(t) = \pm \sum_{i=1}^{\kappa_j} \alpha\delta\left(t - t_{ij}^{(in)}\right) \tag{2}$$

where $\zeta_j^{\pm}(t)$ represents the j^{th} input and the $\pm$ superscript indicates the signal is from an excitatory (+) or inhibitory (-) neuron, respectively. Also, in Eq. (2), the i^{th} impulse time for the j^{th} input is given by $t_{ij}^{(in)}$ and the total number of impulses by κ_j. As with continuous-value neurons, the state of a pulse neuron is simply the temporal summation of the inputs weighted by the interconnection coefficients μ_j and can be expressed by,

$$\rho(t) = \int_{-\infty}^{t} \sum_{j=1}^{n} \mu_j(t') \xi_j^{\pm}(t')\, dt' \tag{3}$$

where $\rho(t)$ represents the summed history up to time t of all of the impulses that were received since the neuron started operating. In this way, $\rho(t)$ is a memory or state variable for the neuron. From Eq. (1), the output of the neuron is given by

$$\eta^{\pm}(t) = \pm \sum_{k=1}^{\kappa} \alpha \delta\left(t - t_k^{(out)}\right). \tag{4}$$

where $t_k^{(out)}$ represents the impulse times for the output and the appropriate sign is taken for the corresponding type of output (excitatory or inhibitory). In a pulse neuron, an output impulse is generated whenever the value of the summed inputs crosses an integer multiple of a threshold γ. Alternatively, this statement is equivalent to saying that the inputs are summed until a threshold is reached, at which time an output impulse is generated and the "summer" is reset to zero. Thus, the k^{th} output impulse time is related to the input impulse times by

$$\rho\left(t_k^{(out)}\right) = k\gamma \tag{5}$$

if the state is not reset after an output impulse is generated and by

$$\rho\left(t_k^{(out)}\right) = \gamma \tag{6}$$

if it is reset. From this relationship, the activity of a neuron can be determined by knowing the state $\rho(t)$. In addition, the cell output can be calculated using

$$\eta^{\pm}(t) = \pm\alpha \frac{d}{dt} \mathrm{int}\left(\frac{1}{\gamma}\rho(t)\right) \tag{7}$$

instead of the time-based sum from Eq. (4). Here, int(x) is the function which returns the greatest integer less than or equal to x. This form does not depend on the output impulse counter k nor the input impulse times explicitly, and it is more useful when layers of neurons are considered and in the design of the optical implementation. The relationship between $\xi_j(t)$, $\rho(t)$, and $\eta(t)$ is illustrated in Fig. 7(b)-(d).

NEURAL INTERCONNECTIONS

Since the connection type (excitatory or inhibitory) is determined by the neuron output and not by the sign of the interconnection coefficients $\mu_j(t)$, only positive values for $\mu_j(t)$ need be considered, as evident by Eqs. (2) and (3). For simplicity, the range of possible values are limited to

$0 \le \mu_i(t) \le 1$, a range which is convenient to use in optical implementations, particularly when the interconnections are made with a holographic transmission mask.

For adaptability, the interconnection coefficients are allowed to vary with time. The key factors for determining an adaptability relationship for Infernet neurons are (1) being able to form long-term memory (LTM, i.e. storing a knowledge base and updating it) and (2) providing a method for global reinforcement (i.e. a way for a supervisor in the real world to tell the Infernet that an output response is either right or wrong). With LTM, the coefficients are adjusted in proportion to the correlation of changes in the activity of the connected neurons. This type of learning is a form of differential Hebbian learning and is used because it is believed to be more beneficial in a network in which predictive or inferential abilities are desired [3,4]. In the case of global reinforcement, all coefficients are increased (corresponding to positive reinforcement) or decreased (negative reinforcement) according to a reinforcement signal which indicates if the right conclusion was inferred to the given inputs. This signal is generated externally by a real-world supervisor and only indirectly affects the operation of the Infernet.

Since the coefficients are continuous-valued, the adaptability relationship meeting the criteria can be described by a differential equation. However, it is convenient to define certain functions first in order to easily define the Hebbian learning characteristics. Let

$$\Sigma(\eta) = \text{the number of impulses in } \eta(t) \text{ over the past } T_1 \text{ time units} \tag{8}$$

$$= \int_{t-T_1}^{t} \frac{1}{\alpha}\eta(t')\,dt' \tag{9}$$

and

$$\Delta(\eta) = \text{the change in } \Sigma(\eta) \text{ over the past } T_2 \text{ time units} \tag{10}$$

$$= \Sigma(\eta)\Big|_t - \Sigma(\eta)\Big|_{t-T_2}. \tag{11}$$

These functions are illustrated in Fig. 8.

With these functions, the differential Hebbian learning relationship is given by

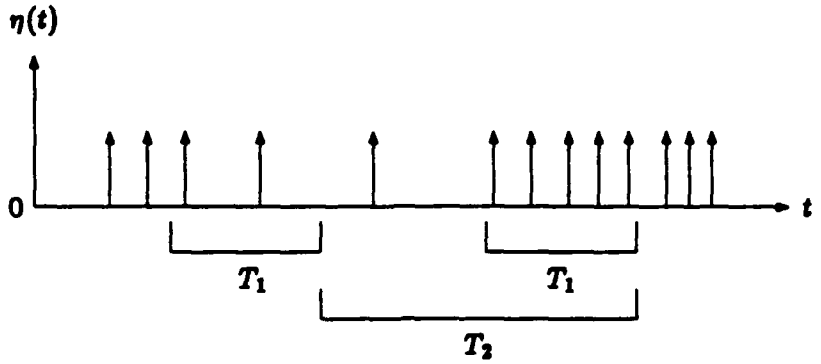
$$\frac{d}{dt}\mu_i = \text{dHebb}[\eta(t), \xi_i(t), \omega(t)] \tag{12}$$

$$= \left[\frac{\Delta(\eta)\cdot\Delta(\xi_i)}{\tau_1} + \frac{\omega(t)}{\tau_2}\right] W(\mu_i) \tag{13}$$

where $\xi_i(t)$ is the i^{th} input to the neuron, $\mu_i(t)$ is its associated synaptic coefficient, and $\omega(t)$ is the reinforcement function whose sign indicates

positive [$\omega(t) > 0$] or negative [$\omega(t) < 0$] reinforcement. Since a real-world supervisor determines $\omega(t)$, its form can be arbitrary. However, one possible function is

$$\omega(t) = \begin{cases} +1 & \text{if the Infernet is correct} \\ 0 & \text{if no correction is desired} \\ -1 & \text{if the Infernet is wrong} \end{cases} \tag{14}$$



$\Sigma(\eta)$ at time t = 5
$\Sigma(\eta)$ at time $t - T_2$ = 2
$\Delta(\eta)$ at time t = 3

Fig. 8 Sample plot of $\eta(t)$ illustrating the $\Sigma(\eta)$ and $\Delta(\eta)$ functions used in the differential Hebbian learning reltionship.

Also in Eq. (13), the time constants τ_l and τ_r are the LTM and the reinforcement time constants, respectively, with τ_r assumed to be much less than τ_l. Finally, $W(\mu_i)$ is a window function which restricts the range of μ_i to be $0 \leq \mu_i \leq 1$. Possible candidates for $W(\mu_i)$ include a normal rectangular window function in this range, $\sin^n(\pi\mu_i)$ where n is a positive integer, and $a\mu_i(1-\mu_i)$ where a is a positive constant. We currently are using the latter form.

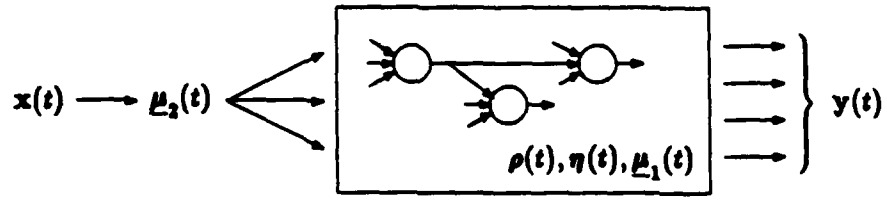
The interconnections described so far represent the normal method of interconnecting neurons. That is, a neuron receives many inputs from the outputs of other neurons and sends its output to still other neurons. To support dynamic neural waves in the HIP, a global signaling method will be necessary to provide constant stimulation for the pulse neurons. Without global signaling of some kind, the HIP would become a static structure that would cease being active once the external inputs were removed.

To implement global signaling in the HIP, simple cases are being considered whereby all cells in the HIP share a common input (with perhaps different

weighting coefficients). Rather than being a pulsed input, it is either a constant value or one that is allowed to vary slowly over time. Functionally, the global input acts like a "neural clock" by stimulating the neurons to fire on a regular basis in the absence of any external (real-world) stimuli. However, the firing frequency can vary between neurons and depends upon their respective global weighting coefficients. A global input of this type represents a distinct departure from conventional neural network theory.

NEURAL LAYERS

A neural layer is composed of many neurons and can have several inputs and outputs. The general form of a layer in the Infernet is shown in Fig. 9. The layer input, x(t) is a vector containing a total of M_{in} signals from other layers. In the same way, there are M_{out} signals in the layer output vector y(t). The layer consists of N neurons, and so the interconnections between them are represented by an N x N intralayer connection matrix $\underline{\mu}_1(t)$.



Number of neurons:	N	
Number of inputs:	M_{in}	
Number of outputs:	M_{out}	
Input vector:	$M_{in} \times 1$	$\mathbf{x}(t)$
Intralayer connection matrix:	$N \times N$	$\underline{\mu}_1(t)$
Interlayer connection matrix	$N \times M_{in}$	$\underline{\mu}_2(t)$
State vector:	$N \times 1$	$\rho(t) = \int_{\text{last pulse}}^{t} \left[\underline{\mu}_1(t')\eta(t') + \underline{\mu}_2(t')\mathbf{x}(t')\right] dt'$
Neuron output vector:	$N \times 1$	$\eta(t) = \pm\alpha \frac{d}{dt} \text{int}\left[\frac{1}{\gamma}\rho(t)\right]$
Output mapping matrix:	$N \times M_{out}$	$\mathbf{H}$
Output vector:	$M_{out} \times 1$	$\mathbf{y}(t) = \mathbf{H}^T\eta(t)$
Learning:	(Intralayer)	$\frac{d}{dt}\underline{\mu}_1 = \text{dHebb}[\eta(t), \eta(t), \omega(t)]$
	(Interlayer)	$\frac{d}{dt}\underline{\mu}_2 = \text{dHebb}[\eta(t), \mathbf{x}(t), \omega(t)]$

Fig. 9 **General form for a layer of pulse neurons in the Infernet.**

Likewise, the interconnections between the layer input x(t) and the N neurons are denoted by an N x M_{in} interlayer connection matrix $\underline{\mu}_2(t)$. The state of the layer is given by the vector $\rho(t)$ and is composed of the states of the N neurons. Correspondingly, $\eta(t)$ is the vector of all neuron outputs. The mapping between y(t) and $\eta(t)$ is given by $y(t) = H^T\eta(t)$ where H is an N x M_{out} binary matrix with one nonzero element per column.

The equations governing the dynamics of the layer are the vector extension of the individual neuron equations. Using Eq. (3), the layer state equation is

$$\rho(t) = \int_{-\infty}^{t} \left[\underline{\mu}_1(t')\eta(t') + \underline{\mu}_2(t')\mathbf{x}(t')\right] dt'. \tag{15}$$

Alternatively, the state-reset form could be used in which case the lower bound on the integral is changed from (infinity) to the time at which the last impulse was generated. From Eq. (7), the neuron outputs are given by

$$\eta^{\pm}(t) = \pm\alpha\frac{d}{dt}\,\mathrm{int}\left(\frac{1}{\gamma}\rho(t)\right) \tag{16}$$

where the sign is determined by the type of output and the int(x) function is applied to each vector element x_i. The intralayer and interlayer connection coefficients vary according to the differential equation given in (13). In matrix form,

$$\frac{d}{dt}\underline{\mu}_1 = \mathrm{dHebb}[\eta(t),\eta(t),\omega(t)] \tag{17}$$

$$= \left[\frac{\Delta(\eta)\otimes\Delta(\eta)}{\eta_l} + \frac{\omega(t)}{\tau_r}\mathbf{O}\right] W\left(\underline{\mu}_1\right) \tag{18}$$

and

$$\frac{d}{dt}\underline{\mu}_2 = \mathrm{dHebb}[\eta(t),\mathbf{x}(t),\omega(t)] \tag{19}$$

$$= \left[\frac{\Delta(\eta)\otimes\Delta(\mathbf{x})}{\eta_l} + \frac{\omega(t)}{\tau_r}\mathbf{O}\right] W\left(\underline{\mu}_2\right) \tag{20}$$

where $\Delta(\eta)$ is the vector whose elements are $\Delta(\eta_i)$ (i.e. the function applied to the elements of (η), **O** is an N x N matrix containing all 1's, and the symbol $\otimes$ denotes outer product.[4]

The most general topology of a layer is one that is "fully connected" so each neuron can communicate with every other cell. In this case, if there

---

[4] If $\mathbf{A} = \mathbf{b}\otimes\mathbf{c}$, then $a_{ij} = b_i c_j$.

are N neurons in a layer, there are a total of N^2 interconnections possible and the layer can have at most N inputs and N outputs. This configuration permits an output to be affected directly by an input via one neuron. Consequently, the inputs are not well "mixed" together before an output is generated. Also, the propagation of dynamic neural waves can be hampered by such high connectivity.

Therefore, layers in the Infernet are more restricted in their overall configuration to allow for more processing of the inputs before generating the outputs. Instead of making N connections, each neuron makes on average M_L connections with other cells *in its neighboring vicinity.* Here, M_{in}, M_{out}, and M_L are all on the order of M where $1 << M << N$. This arrangement creates a high local interconnection density, given by M_L, and a low global interconnection density, given by

$$M_G = \frac{\text{total average number of interconnections in a layer}}{\text{total number of possible interconnections in a layer}} \tag{21}$$

$$= \frac{NM_L}{N^2} = \frac{M_L}{N}. \tag{22}$$

With this type of topology, the intralayer connection matrix $\mathbf{M}_1$ has approximately NM_L nonzero elements somewhat clustered together (reflecting the neighborhood connection idea). Since $M_L << N$, $\mathbf{M}_1$ tends to be a sparse matrix.

As defined by Eq. (22), M_G also acts as a figure of merit for the amount of processing performed by the layer. It also is important in characterizing aspects of dynamic neural wave propagation through the layer. If $M_G << 1$, the layer has many neurons between the cells which receive the layer inputs and the ones which provide the layer outputs. Thus, the inputs undergo a relatively large amount of processing. However, if $M_G \to 1$, the inputs are processed relatively little compared to the former case.

THE INFERNET NEURAL NETWORK

Many of these layers are interconnected to form the Infernet architecture shown in Figs. 2-5. Since the neural signals are temporal impulses, the Infernet performs temporal processing on a spatial distribution of signals. If $\rho_{net}(t)$ represents the state of the network (as the vector of all neuron states), the overall transfer function G(.) of the Infernet determines the external outputs r in response to the external inputs given $\rho_{net}(t)$ and the knowledge base (the spatial distribution of interconnection coefficients throughout the Infernet). This function can be summarized by

$$\mathbf{r} = G(\rho_{net}; \mathbf{s}). \tag{23}$$

In conjunction with the real-world feedback function Fr to include the effects of r in s, Eqs. (15)-(23) form the basic mathematical model of the Infernet.

SUMMARY OF INFERNET FEATURES

The distinguishing characteristics of the Infernet can be seen best by comparing its various aspects with some of the previous work on artificial neural networks and systems. The following list summarizes the significant conceptual features of the Infernet:

1. The Infernet is a system that can exploit causality. That is, the information being processed is encoded as a spatial distribution of temporal signals. The Infernet cannot produce any outputs if time is "stopped," although all neuron states then can be observed. Furthermore, it is not necessarily simple to deduce what the Infernet will do next given a measurement of the neuron states. Thus, the Infernet is active only when its neurons are producing outputs pulses. In contrast, most other neural networks are designed to process a spatial distribution of signals (i.e. vectors) such that time can be stopped temporarily and the vectors still provide output information. Some of them only have valid outputs when the activity of the network ceases.

2. The Infernet architecture includes real-world inputs s and outputs r as opposed to assuming that information already is encoded into a neural format using some method. In addition, the effects of real-world feedback are available to the Infernet to increase learning efficiency.

3. The Infernet is a specific example of an architecture which uses layers as the fundamental building block for designing a system. Furthermore, the Infernet architecture is not just a serial arrangement of layers but a two-dimensional distribution of layers with many feedforward and feedback paths. Conventional neural networks tend to be a linear sequence of layers with little or no feedback outside the layers.

4. The inference engine of the Infernet, the HIP, is a collection of stacked layers that communicate via monitoring and guidance signals between neighboring layers. This interaction between these layers in the Infernet should give it inferential capabilities. In addition, access ports (the internal input and output) are available to a real-world supervisor so he/she can influence and observe the operation of the upper levels of the HIP and the inference process. Other neural networks that have been studied have not been based on this hierarchical structure.

5. The knowledge base is stored distributively throughout the HIP in the interconnection strengths and topology. As a result, the HIP does not have any explicitly-defined memory structures or components (like the content-addressable and associative memories being studied by other researchers).

6. Access to the knowledge base is via dynamic neural waves (temporal flow of increased neuron activity) which cycle through the layers of the HIP. Stopping time conceals the knowledge base in that it is not necessarily simple to determine what the Infernet was processing. To

the best of our knowledge, dynamic knowledge base encoding and access is unique to the Infernet.

7. The Infernet can anticipate inputs since previous input stimuli and/or current dynamic neural waves can prime the state of the network in specific regions of various layers so these regions are nearly ready to excite new waves.

8. The HIP operates even in the absence of any external inputs because of a global input that is common to all HIP neurons. Most other neural network models only focus on the select electrical signaling method believed to be used in the brain.

9. The learning of the Infernet can be unsupervised and/or supervised indirectly via the global reinforcement signal. Under indirect supervision, the interconnection coefficients are increased when the output conclusions are correct and decreased if they are wrong. The supervision is indirect in that only interconnection strengths are changed and not the physical processing actions that generated the outputs. In addition, when a knowledge base has been learned satisfactorily, the learning procedure can be halted. Other types of neural networks use unsupervised learning.

10. An Infernet layer is not "fully interconnected" intentionally for three reasons. First, it allows the processing characteristics of this type of layer to be studied which hopefully can aid in designing a network. Secondly, it is believed to be necessary for supporting dynamic neural waves. Finally, this type of interconnection arrangement reduces the technological demands on the implementation. Other research approaches permit a "fully interconnected" network as needed.

11. The interconnection strengths in an Infernet layer can have only positive values between 0 and 1. The type of connection (excitatory or inhibitory) is determined by the associated output neuron. In the other models, the strengths are bipolar real numbers (usually without a limit) and their sign determines the connection type.

12. For theoretical analyses, the output signals of Infernet neurons are idealized to be continuous-time δ functions which have the same amplitude. Conventional neurons usually have either continuous-valued outputs (analogous to analog communications) or binary-valued output states (which can convey only a limited amount of information).

Rather than being a new neural model, the Infernet represents an architecture which is intended to complement the work of other neural network researchers.

SECOND-YEAR GOALS

Given the system design of the Infernet, there were three goals for the second year of the project:

1. Design a suitable optical implementation for the Infernet.

2 Perform computer simulations on the Infernet to study the dynamic knowledge encoding scheme.

3. Develop a two-dimensional photometric system for use with our computerized control system (CCS).

Because these goals are relatively independent, they can be worked on in parallel. However, an actual implementation of the optical design from the first goal would have to wait until the second goal confirmed that the Infernet architecture acts as an inference machine.

SYSTEM DEVELOPMENT

Optical Implementations of the Infernet

The layer is the basic building block in the Infernet because it is the fundamental structure to which all neural equations apply. Thus, the focus of the optical implementation is to realize a cascadable layer. This approach is completely general since given a working layer, several layers can be arranged to form the Infernet network. Alternatively, the entire network can be considered one big layer.

Rather than simply showing the final design of our optically-implemented layer (herein referred to as an optical layer), it is insightful to start at the equation level and build up to the optical realization. Mathematically, the main variable to consider is the vector state variable $\rho(t)$. Its scalar elements $\rho_i(t)$, $i=1,2,...,N$, are functions of two independent variables, i and t. Since optical data paths are two-dimensional, the state elements can be represented as a scalar function of two *spatial* variables, t^S,β, as shown in Fig. 10.

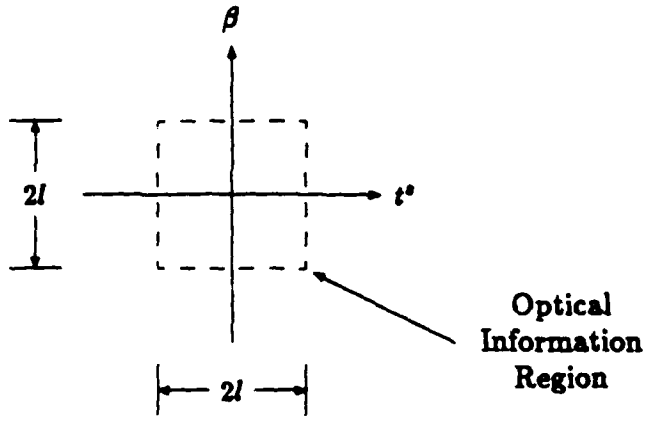


Fig. 10 The spatial coordinate system used in the optical data path. The region within the dashed box contains the light distribution representing a neural variable.

Here, t^S corresponds to a spatial axis representing the temporal variable t and β corresponds to i and is orthogonal to t^S. The layer state in the space domain can be written as the scalar function $\rho(t^S,\beta)$. Note, however, that in the vector state variable, i is a discrete index whereas β can be a continuous spatial variable. This flexibility allows an optical layer to implement continuum of neurons, limited only by the resolution of the optical components.

The same coordinate transformation can be applied to $\eta(t)$ and $x(t)$ to give $\eta(t^S,\beta)$ and $x(t^S,\beta)$, respectively. In the same manner, the interconnection matrices $\underline{\mu}_1(t)$ and $\underline{\mu}_2(t)$ become scalar functions of the three spatial variables (t^S,β_i,β_o). Thus, $\mu_{1ij}(t)$ and $\mu_{2ij}(t)$ can be rewritten as $\mu_1(t^S,\beta_i,\beta_o)$ and $\mu_2(t^S,\beta_i,\beta_o)$ where β_i and β_o correspond to i and j, respectively.[5] For reference, these transformations are summarized below:

$$\begin{aligned} \rho(t) &\longrightarrow \rho(t^s,\beta) & \underline{\mu}_1(t) &\longrightarrow \mu_1(t^s,\beta_i,\beta_o) \\ \eta(t) &\longrightarrow \eta(t^s,\beta) & \underline{\mu}_2(t) &\longrightarrow \mu_2(t^s,\beta_i,\beta_o) \end{aligned} \tag{24}$$

In terms of these new spatial coordinates, the state equation in (15) can be rewritten as

$$\rho(t_o^s,\beta_o) = \int_{-\infty}^{t_o^s}\int_{-\infty}^{+\infty} [\mu_1(t_i^s,\beta_i,\beta_o)\,\eta(t_i^s,\beta_i) + \mu_2(t_i^s,\beta_i,\beta_o)\,x(t_i^s,\beta_i)]\; d\beta_i\, dt_i^s \tag{25}$$

where the second integral over β_i has replaced the finite sum of the dot products in Eq. (15). Note also that the β_i integral extends over all space (from $-\infty$ to $+\infty$). In practice, this range is reduced to the finite length of one side of the two-dimensional optical data paths.

Equation (25) is a special case of the more general form given by

$$\rho(t_o^s,\beta_o) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mu(t_i^s,\beta_i;\, t_o^s,\beta_o)\,\eta(t_i^s,\beta_i)\; d\beta_i\, dt_i^s \tag{26}$$

which represents a space-variant system since the integrand kernel $\mu(t^S_i,\beta_i;t^S_o,\beta_o)$ depends upon the output coordinates (t^S_o,β_o) as well as the input coordinates (t^S_i,β_i). This type of equation can be implemented using the optical setup shown in Fig. 11.[6] In this system, the amplitude distribution of the light incident on the input plane is given by $\eta(t^S_i,\beta_i)$ and by $\rho(t^S_i,\beta_i)$ in the output plane. A lens with focal length f images the two planes with unity magnification. The space-variance is introduced by a computer-generated hologram (CGH) located directly in front of the lens. This hologram encodes the interconnection matrix $\mu(t^S_i,\beta_i;t^S_o,\beta_o)$. While

---

[5] The subscripts "i" and "o" will refer to certain input and output planes of the optical layer, respectively.

[6] Provide certain approximations are maintained.

current CGH technology does not permit the interconnection strengths to be varied in real time, this optical setup computes Eq. (26), making it useful for fixed (learned) knowledge bases.

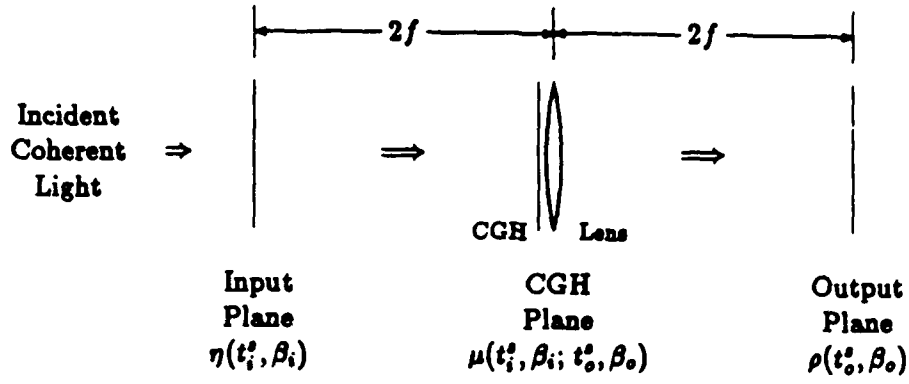


Fig. 11. A general space-variant optical imaging system which implements Eq. (26)

For computing Eq. (25), the design in Fig. 11 leads to the optical architecture shown in Fig. 12. This setup is composed of three space-variant imaging systems. In the first one, the neuron outputs $\eta(t^S_i,\beta_i)$ are present at input plane 1 and CGH 1 encodes the intralayer connection matrix $\mu_1(t^S_i,\beta_i,\beta_O)$. The second imaging path has $x(t^S_i,\beta_i)$ at input plane 2 and $\mu_2(t^S_i,\beta_i,\beta_O)$ stored in CGH 2. Both CGH 1 and 2 are designed so these paths sum only over β_i to produce

$$A = \int_{-l}^{+l} \mu_1(t_i^s, \beta_i, \beta_o)\, \eta(t_i^s, \beta_i)\, d\beta_i \tag{27}$$

and

$$B = \int_{-l}^{+l} \mu_2(t_i^s, \beta_i, \beta_o)\, x(t_i^s, \beta_i)\, d\beta_i \tag{28}$$

at the beam splitter (BS). Here, the optical data paths are assumed to be within $-l \le t^S_i,\ \beta_i \le +l$. These terms then are combined and processed by CGH 3 which simply sums over t^S_i up to $t^S_\cap$ to give

$$\rho(t_o^s, \beta_o) \approx \int_{-l}^{t_o^s} A + B\, dt_i^s \tag{29}$$

at the output plane. Note that $-l$ has been used as an approximation for -(infinity) in Eq. (29). This optical system thus implements an approximation of the layer state equation in (25).

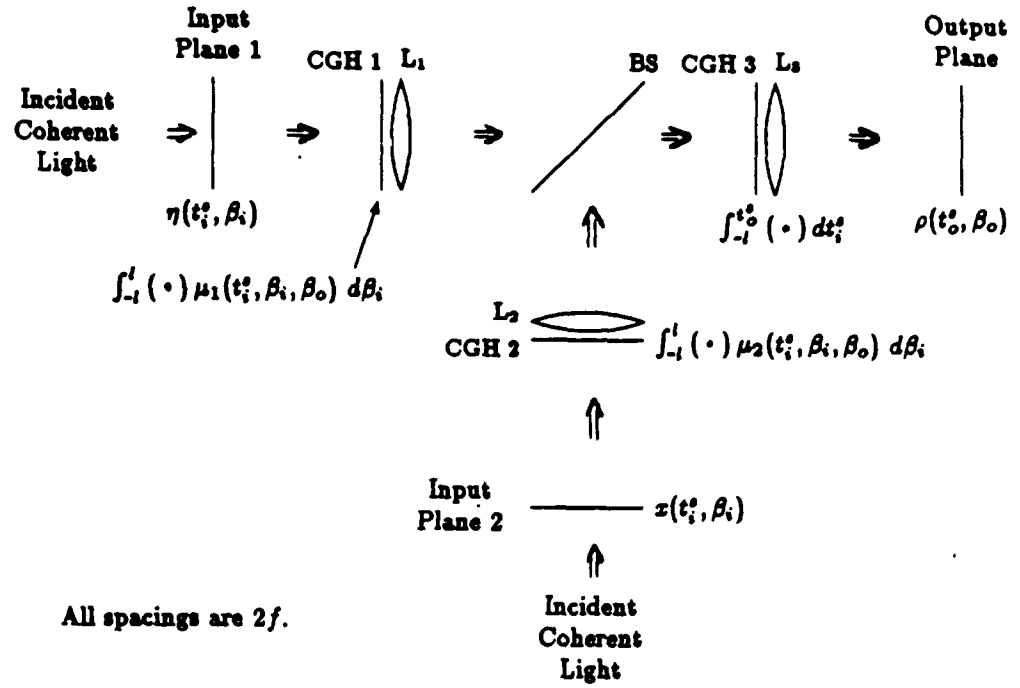


Fig. 12 **The space-variant optical imaging system which implements the layer state equation in Eq. (25).**

To complete the optical layer, the neuron outputs must be derived from the states as defined by Eq. (16). In terms of the spatial coordinates (t^s_o, β_o), the neuron outputs are given by

$$\eta^{\pm}(t_o^s, \beta_o) = \pm\alpha \frac{\partial}{\partial t_o^s} \text{int}\left[\frac{1}{\gamma}\rho(t_o^s, \beta_o)\right]. \tag{30}$$

As an approximation, the integer function can be ignored, resulting in the linear relationship.

$$\eta^{\pm}(t_o^s, \beta_o) \approx \pm\frac{\alpha}{\gamma} \frac{\partial}{\partial t_o^s}\rho(t_o^s, \beta_o) \tag{31}$$

which can be implemented using another space-variant imaging system as shown in Fig. 11 with an appropriately design CGH. Alternatively, $\eta(t^s_o, \beta_o)$ can be computed using a conventional optical processor based on Fourier optics [5]. In this setup, the Fourier plane filter can be either a CGH, a half-plane optical density wedge, or a schlieren knife-edge filter. These filters are oriented perpendicular to the t^s_o spatial axis so as to calculate the derivative in the t^s_o direction only. While all of these filters approximate a true differentiation filter, the schlieren technique

is the simplest but the most crude in that it more closely resembles a high-pass filter rather than a true differentiator.

A more exact way of calculating $\eta(t^s_0,\beta_0)$ is to threshold $\rho(t^s_0,\beta_0)$ using a two-dimensional spatial light modulator such as the microchannel spatial light modulator (MSLM) [6,7] and then compute the partial derivative along t^s_0 with a conventional optical processor. In addition to being more accurate, this method allows $\rho(t^s_0,\beta_0)$ to remain in the interval $[0,\gamma]$ thereby eliminating the need to evaluate the integral over t^s_i in Eq. (16) starting from $-\infty$.

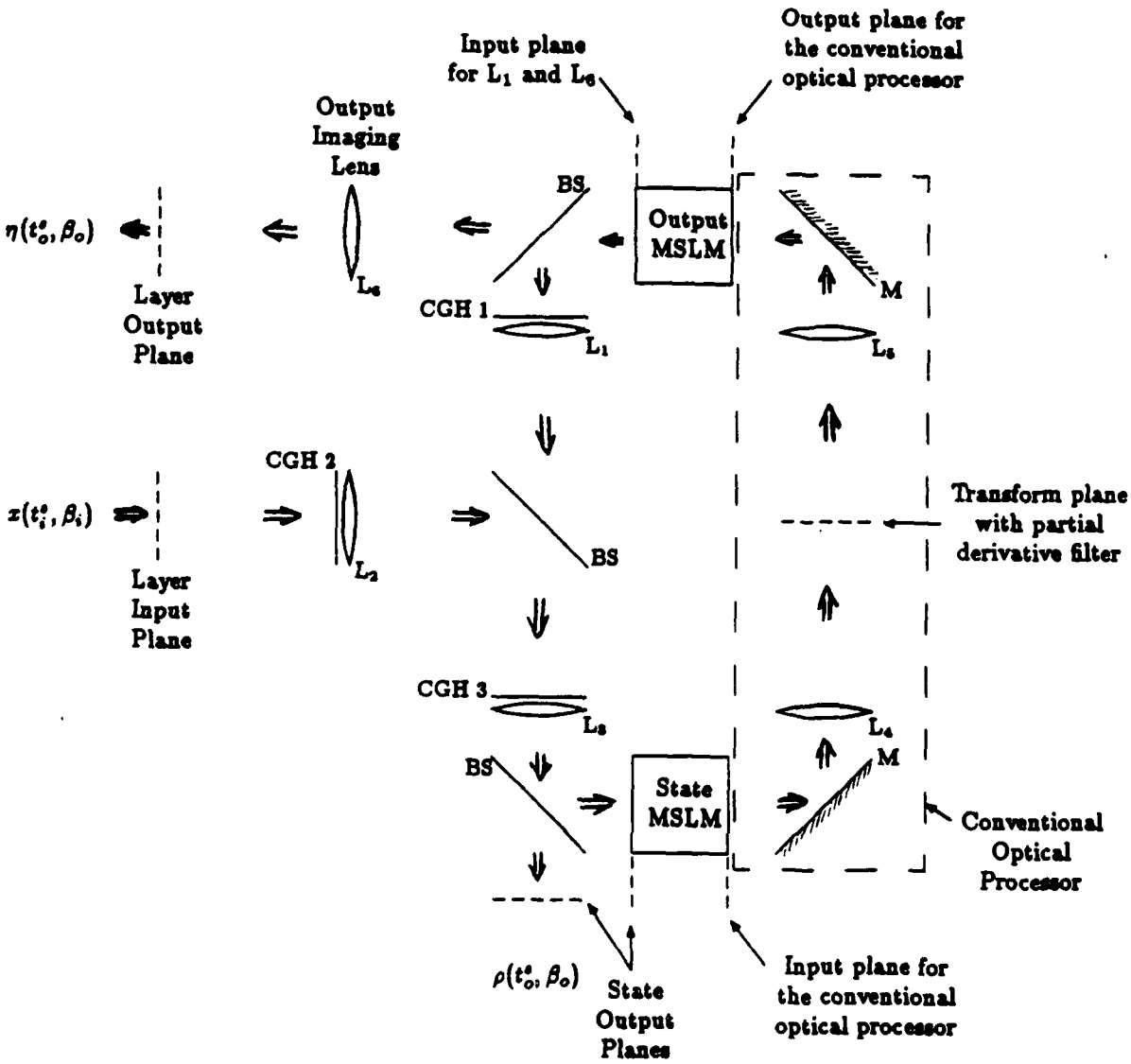


Fig. 13 The optical implementation of an Infernet layer. As in Fig. 12, the distances between the imaging lenses L_1, L_2, and L_3 and their respective input and output planes are 2f. In the conventional optical processor, the distances between the transform lenses L_4 and L_5 and their associated input, transform, and output planes are f.

Using the MSLM approach, the complete optical layer is shown in Fig. 13. The layer input $x(t_i^s,\beta_i)$ enters the system at the layer input plane. This signal can be the output from another optical layer or be generated electronically via an electrically-addressed two-dimensional spatial light modulator (such as the electron-beam-addressed MSLM [8]). Likewise, $\eta(t_o^s,\beta_o)$ exits at the layer output plane and can be directed into the input plane of another optical layer or measured with a two-dimensional photodetector array. In addition, the state output $\rho(t_o^s,\beta_o)$ is available at the state output plane. Note that the optical layer illustrated here employs two MSLM's, one to threshold the layer state and other to buffer the layer output, and uses the schlieren technique in a conventional optical processor to approximate the partial derivative given in Eq. (31).

When an optical layer is in operation, dynamic neural waves would appear as regions the light moving around the readout side of the output MSLM. Waves may be exited within the layer or enter via the layer input plane. They then may cycle within this layer, exciting more waves, or be sent to other optical layers. Since these waves are of a temporal nature, they do not appear in the static hardware diagram in Fig. 13.

An alternative to the general optical layer shown in Fig. 13 is the simplied design shown in Fig. 14. The key simplifications in this design are that the cells are continuous-valued rather than being pulse neurons. This eliminates the need for a spatial representation of the time axis (t^s) and thus allows a matrix of cells to be processed (instead of a vector of neurons as in Fig. 13). While this design is much more sensitive to noise, it is simpler to construct and operate in order to demonstrate the basic implementation concepts before proceeding to the more complex optical layer in Fig. 13.

With this optical implementation for a layer, the role of the electronic CCS is to control all layers via their MSLMs, monitor the operation of the network, and interact with the real world. In this latter case, the CCS would also function as the stimuli sensing and the data reducing layers in an input processor (as shown in Fig. 3) and the action generating layer in an output processor (shown in Fig. 4).

The goal of the simplified optical layer is to demonstrate the basic architectural components of the Infernet. Schematically, the relationship of the simplified optical layer to the Infernet block diagram in Fig. 5 is illustrated in Fig. 15. Thus, for demonstration purposes, it can act as the entire first level in the HIP. Then, the electrical-to-optical input spatial light modulator must emulate all of the outputs of the input processors as well as the guidance signal from the second HIP level. Similarly, the optical output sensed by the camera represents the input signals to the output processors and the monitor signal for the second HIP level. Because the focus of the simplified optical layer is more specialized, the role of the CCS is complicated since it must keep track of many different signals. However, the added complexity to the CCS is small compared to the benefit of being able to demonstrate more quickly an optical implementation of the key aspects of the Infernet.

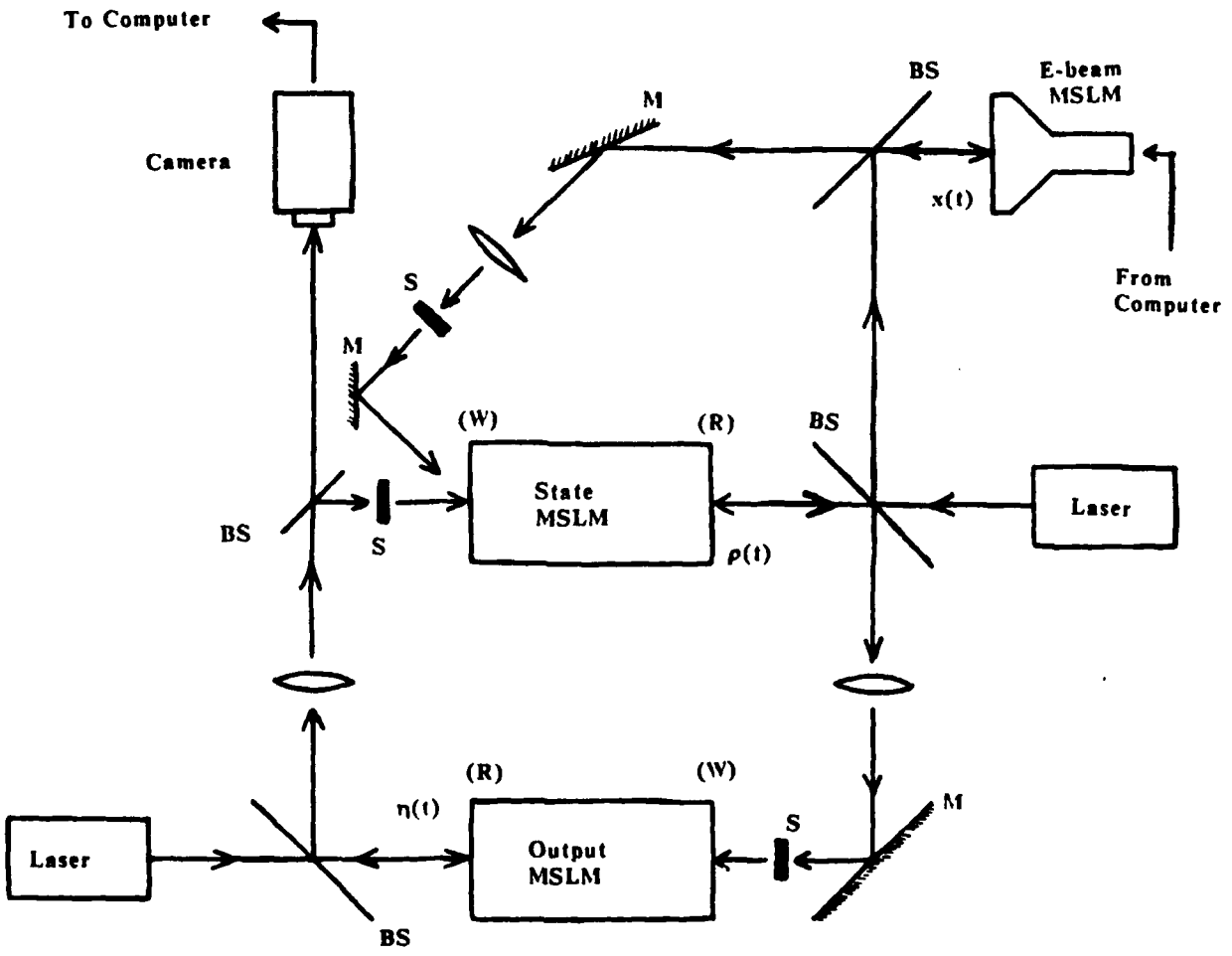


Fig. 14 Simplified version of the optical layer shown in Fig. 13. Interconnection CGH's are omitted for clarity. Each optical path represents the firing activity of a matrix of cells.

At the system level, there are primarily two limitations common to these implementation designs. First, the use of CGHs to realize the interconnections does not permit the strengths to vary with time and so they must be predetermined, resulting in a fixed, nonadaptable knowledge base. However, this limitation is not too restrictive in that an experimental prototype of the optical architecture still can be built and studied during the third year of the program. Secondly, the performance of a layer is limited by the resolution and speed of the spatial light modulators and the number of electronic-to-optical (and optical-to-electronic) data conversions in the system. In the general optical layer (Fig. 13), a third limitation exists in that the layer inputs must be shifted along the t^S_i axis for continuous time operation. Another spatial light modulator which has the needed shifting property would have to be inserted at the layer input plane to accomplish this task. Alternatively, if $x(t^S_i,\beta_i)$ is generated via an electrically-addressed spatial light modulator, the shifting can be performed electronically. However, the current optical device technology still can be used to demonstrate the implementation designs.

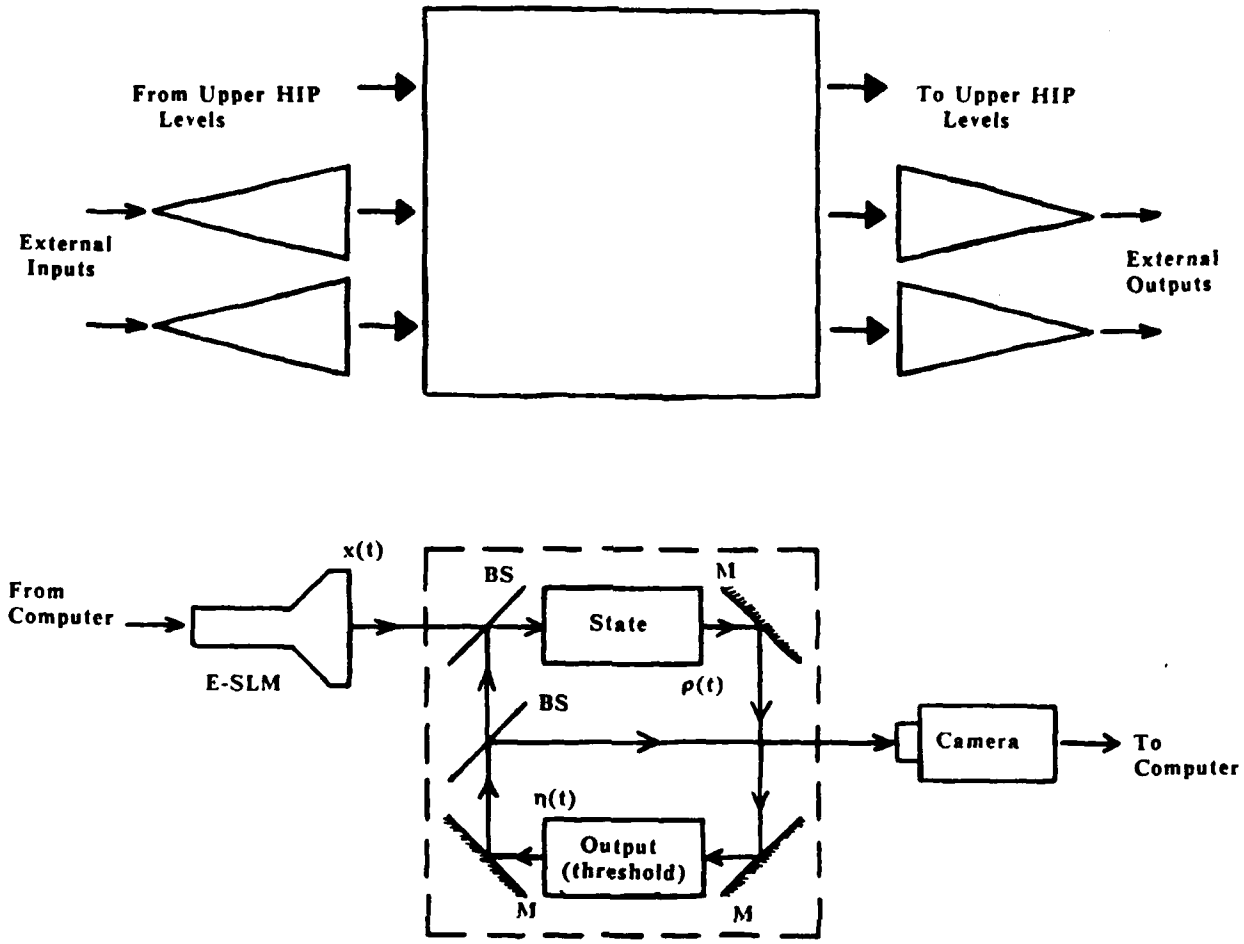


Fig. 15 **Relationship of the simplified optical layer (Fig. 14) to the block diagram of the Infernet (Fig. 5).**

COMPUTER SIMULATIONS

The goals of the computer simulations are threefold: (1) to understand the dynamics of the Infernet, (2) to explore various methods by which a knowledge base can be encoded into the Infernet, and (3) to determine the corresponding interconnection strengths and compute the required CGHs for the optical implementation demonstrations.

During the second year of the program, several simulations were carried out to study the various aspects of the Infernet model. The primary focus of these simulations is to gain insight into the dynamics of a general layer. Early simulations showed that significant computation power is needed to process a layer with only tens of cells. Consequently, smaller layers are being studied to limit the simulation time to a reasonable value, given our computing resources (AT&T 3B2 running Unix). A summary of the major results is given below.

With no global input, pulse neurons act as time-domain filters, reducing the number of pulses being produced by successive cells. Very few pulses are produced at the outputs of a layer even when there are many pulses at the layer inputs. This confirms the need for additional (global and/or continuous) excitation in some way. With a constant global input to all cells, a background firing rate is obtained with the intracellular connections simply varying the firing rate from this bias value. Unless stated to the contrary, all later simulations have the equivalent of a continuous global input that is spatially uniform and constant in magnitude to all cells.

For a dynamic neural wave to exist, a localized region of increased firing activity (a wave "packet") should be able to exist and propagate without spreading out over the entire layer. With a nearest-neighbor interconnect scheme (in which a cell makes connections with only its closest neighbors), any wave packet eventually disperses over time and ceases to be localized and propagating. With a localized-feedforward interconnect scheme (in which a cell connects to its neighboring cells only on one side), a wave packet disperses and forms linear wave "fronts" (a linear wave packet) that extends across the entire layer and propagates in one direction.

Mathematically, if a wave packet has a two-dimensional envelope (representing the firing rate of a spatial distribution of pulse neurons) denoted by $G(x,y)$ at time $t=0$, the propagating envelope at time t can be expressed as $G(x-v_x t, y-v_y t)$ where v_x and v_y are the velocities along the x and y directions and are functions of time. In the absence of dispersion, the envelope retains its shape. However, it can be shown that the envelope at time $t+1$ is related to the envelope at time t through a convolution with the weighting vectors of the cells.[7] For no dispersion to exist, this implies that the weights must be set such that $G(x,y)$ convolved with the weight vectors should reproduce $G(x,y)$. This implies the weight vectors are impulses, indicating that each cell makes connections with only onto itself. If both $G(x,y)$ and the weight vectors were Gaussian in shape, the convolution would result in another Gaussian with a larger variance, thus effectively producing dispersion. Hence, the computer simulations and mathematical analyses show that any interconnect scheme will produce dispersion, thus prohibiting the propagation of localized regions of firing activity.

As originally envisioned, a neural wave was a representation of neural activity within the layer. Given the dispersion problem, an alternative wave definition was investigated to see if it had the desired properties. Rather than looking for the firing activity to follow a wave envelope, we decided to look for possible wave-propagation behavior in the continuous global input. Previously, this input was constant throughout the layer. Now, it was changed to have a spatial Gaussian-like distribution whose position could be affected by the neural firing activity within the layer. The net effect was that the cell activities were the wave shape plus a

---

[7] The weighting vector for a cell is simply the vector of weights for the inputs to that cell.

perturbation term due to the interconnection structure. The position of this "global input wave" was monitored by following the central peak.

Primarily three different schemes for the interaction between the cell activities and the global input wave were examined: (1) net wave gradient, (2) potential field, and (3) variable deflection. These methods are classified by the way in which the cells could alter the wave's trajectory. In the first case, the wave acceleration in a particular direction was proportional to the sum of the cell activities weighted by the gradient of the wave envelope in that direction.

In the second method, the cell activities acted as the drive to create a potential field which could create forces (and thus accelerations) on the wave. This approach is analogous to treating the wave as an electron in a quasi-static electric potential distribution and analyzing the system using Newtonian mechanics. Both of these methods produced relatively inflexible layers in that it was difficult to control the resulting wave trajectories and, more importantly, the coupling between trajectories.

In the third simulation, the cell activities followed the global input wave envelope exactly and the speed of the wave was held constant. The interconnect structure served to create a spatial distribution of angular deflections which could change the direction of the wave trajectory. The actual deflection at time t was the accumulated deflection at the center peak of the wave (variable deflection throughout the wave was not permitted as this would deform the its shape). Unlike the previous two trials, this formlation resulted in a layer that offered significant interaction between wave trajectories and allowed localized limit cycles.

Unfortunately, the layer was also a highly chaotic system -- any small perturbations in the initial conditions produced very different trajectories. This is most undesirable for a system which is supposed to be fault tolerant! While it was expected that a wave in the Infernet might take slightly different paths every time the system responsed to the same input, it is necessary that the system should consistently produce the correct response (after learning).

Once again, we rethought our definition of a wave, keeping in mind our past simulation experiences. From the beginning, we viewed a general Infernet layer as a medium capable of supporting and propagating localized regions of increased neural activity. However, the simulations and associated mathematical analyses did not support this behavior. An alternative to this view is to consider the layer as a resonator. Different wave trajectories become different resonator modes and the transition from one trajectory to another is the same as a transition from one mode to another mode. Similarly, a limit-cycle trajectory maps to a single resonator mode.

In order to see how a layer can act as a resonator, consider HIP Level 1 to be a single layer, HIP Level 2 to provide unity feedback, and HIP Level 3 to be nonexistent. Then, the outputs of Level 1 are effectively fed back into the inputs of Level 1 (ignoring the input processors for now). A mode becomes any input distribution to Level 1 which produces the same distribution at the output of Level 1.[8] To determine how the layer forming

the first level depends upon the neural network model, there are several different models are available (for example, back-prop [9,10] and adaptive resonance theory [11-14]). Because of their connectionistic origins, a mode is relatively fault tolerant and stable. Therefore, by using these ideas, a resonator theory of inference may be investigated in the context of the Infernet. We are currently examining this approach.

TWO-DIMENSIONAL PHOTOCALIBRATION SYSTEM

While working on the various computer simulations of the Infernet, we also developed a two-dimensional photocalibration system [15] for our computerized control system (CCS). The objectives of this subsystem were twofold:

1. Provide fast, analog measurements of a general two-dimensional grid of light beams in the presence of background light (noise) and with random-access selection (as opposed to raster scanning).

2. Allow the grid parameters to be easily changed and have automatic recalibration.

The second objective gives our photocalibration system considerable flexibility. It permits the size of the grid to change as well as the orientation while keeping the necessary overhead parameters transparent to the measurement process. It also allows any inversions of the axes due to lenses and mirrors to be accounted for regardless of the actual optical setup.

The only hardware required is a vidicon camera and a frame grabber board for our AT&T PC 6300 (the central component of the CCS). Consequently, the above objectives are realized in software. For our system, we used a CCD-array camera and a PCVISIONplus Frame Grabber board with the ITEX PCplus library of image processing routines. Our software was written in Microsoft C version 5.0. There are two sets of routines, an initialization module for the second objective and a runtime module for the first one. Given the size of the desired grid (say, N x M), the initialization module determines the position and angle of rotation of the grid in the camera's field of view. Effectively, the module produces a mapping between the pixel coordinates of the optical system output and the pixel coordinates of the camera output. This task is accomplished by operating the optical system and processing four camera pictures. First, the optical output is erased and the first picture of the background light distribution is taken. Next, the optical system is operated such that it produces a beam of light at the (1,1) grid location in the output and a picture (#2) of the resulting image is taken. By comparing pictures #1 and #2, the initialization module can determine the location of (1,1) for the grid.

Then, without clearing the optical output, the optical setup is operated such that pixel (1,M) is illuminated while leaving the light at (1,1)

---

[8] A mode is essentially an eigenvector of the layer with a corresponding eigenvalue of 1.

intact. This image is captured in picture #3 and then compared to picture #2 to determine the (1,M) grid location in the camera's field of view. This also allows the angular rotation of the grid rows to be calculated. Finally, the light beam at (N,1) is produced via the optical system and a picture (#4) taken. As with pictures #2 and #3, pictures #3 and #4 are compared to determine the (N,1) pixel location and the angular rotation of the grid columns. An interesting aspect of this algorithm is that the optical grid pattern does not have to be rectangular; the grid only needs to form a parallelogram. Also, there can any number of row or column inversions.

With these measurements, the initialization module computes all of the necessary values needed by the runtime module to translate a grid coordinate (n,m) into a position in the camera field of view. Then, the maximum pixel radius for a grid elements (assuming a circular shape) is calculated from the size of the grid and the resulting row and column spacings between the grid elements.

The routines contained within the initialization module are:

Routine	Description
set_up_slm_mapping	Turns on and initializes the frame grabber, and captures the reference picture (#1).
find_slm_zero_zero	Takes picture #2 and locates the (1,1) grid element.
find_slm_maxx_zero	Takes picture #3 and locates the (1,M) grid element.
find_slm_zero_maxy	Takes picture #4 and locates the (N,1) grid element.
calculate_partials	Converts the information obtained from the above four routines into the mapping transformation values.
save_slm_map_data	Saves the mapping transformation parameters to a disk file so that the optical output does not have to be remapped every time the CCS is turned on.
load_slm_map_data	Reads the mapping transformation parameters from a disk file.
fg_x	Returns the (x,y) coordinates in the frame grabber fg coordinate system of a particular optical output grid element.
on_off_and_size	Allows a pixel representation to be changed between digital (on or off) and analog values. Also allows the radius of pixel to be set manually.

max_pixel_radius	Determines the size of the area to be searched when calculating the average intensity for an optical output pixel.
draw_mapping	Produces a pixel pattern showing the locations of alloptical output pixels enclosed by two concentric circles, one at the expected radius and the other at the maximum radius. This image can be superimposed onto an optical output image to see if the optical system has been moved since the mapping was last calculated. Useful for diagnostic purposes.

Routines for capturing pictures with the camera and any associated image processing routines are contained within the ITEX PCplus library.

Using the parameters found by the initialization module, the runtime module simply determines the average light intensity (encoded from 0 to 127) within the maximum radius for any grid element upon request. To use the runtime module, the optical system is cleared so the optical output is erased and a background picture is taken for reference (to reduce noise significantly). When the intensity at a particular pixel is desired, another picture is taken and the appropriate routine called which returns the value of the grid element. Other locations within the grid may be accessed as required.

There is only one major routine in the runtime module and it is listed below:

Routine	Description
get_avg_value	Returns the average value of intensity inside the max_pixel_radius area for a given grid element.

Together, the initialization and runtime modules satisfy the desired objectives outlines above and realize a translation, rotation, and scale-invariant photodetection system for measuring two-dimensional grid-like distributions of light. This system completes our design of the CCS for our optical systems based on spatial light modulators.

FUTURE WORK

With the CCS complete, the bulk of the work lies in theoretical analyses and computer simulations of the Infernet architecture. We plan to continue researching our resonator viewpoint of the HIP levels and of encoding a knowledge base into such a system using this mode theory. Once the simulations render a working inference machine, a set of CGH's will be made and the simplified optical layer will be demonstrated. Any demonstrations of the general Infernet optical layer (based on pulse neurons) will be foregone until the simulations demonstrate the advantages of this implementation over the simplified version.

REFERENCES

1. C. Warde, "Optical Inference Machines," Interim Report, AFOSR-86-0301, June, 1988.

2. C. Warde and J. A. Kottas, "Hybrid Optical Inference Machines: Architectural Considerations," Appl. Opt. 25, 940 (1986).

3. A. H. Klopf, "A Drive-Reinforcement Model of Single Neuron Function: An Alternative to the Hebbian Neuronal Model," in *Neural Networks for Computing*, AIP Conference Proceedings No. 151, J. S. Denker, ed., Snowbird, Utah, 1986, p. 265.

4. B. Kosko, "Differential Hebbian Learning," in *Neural Networks for Computing*, AIP Conference Proceedings No. 151, J. S. Denker, ed., Snowbird, Utah, 1986, p. 277.

5. J. W. Goodman, *An Introduction to Fourier Optics* (McGraw-Hill, New York, 1969).

6. C. Warde, A. M. Weiss, A. D. Fisher, and J. I. Thackara, "Optical Information Processing Characteristics of the Microchannel Spatial Light Modulator," Appl. Opt. 20, 2066 (1981).

7. C. Warde and J. I. Thackara, "Operating Modes of the Microchannel Spatial Light Modulator," Opt. Eng. 22, 695 (1983).

8. A. Schwartz, X. Y. Wang, and C. Warde, "Electron-Beam-Addressed Microchannel Spatial Light Modulator," Opt. Eng. 24, 119 (1985).

9. D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing, Vol. 2* (MIT Press, Cambridge, MA, 1986).

10. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," Nature 323, 533 (October 9, 1986).

11. S. Grossberg, *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*, (Reidel, Norwell, MA, 1982).

12. M. A. Cohen and S. Grossberg, "Masking Fields: A Massively Parallel Neural Architecture For Learing, Recognizing, and Predicting Multiple Groupings of Patterned Data," Appl. Opt. {(bold) 26}, 1866 (1987).

13. G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," Comput. Vision, Graphics, and Image Process. 37, 54 (1987).

14. G. A. Carpenter and S. Grossberg, "Absolutely Stable Learning of Recognition Codes by a Self-Organizing Neural Network," in *Neural Networks for Computing*, AIP Conference Proceedings No. 151, J. S. Denker, ed., Snowbird, Utah, 1986, p. 77.

15. M. G. Frey, "A Translation, Rotation, and Scale Invariant Photodetection System," MIT Bachelor's Thesis, Cambridge, MA, (May 1988).

LIST OF PERSONNEL

Faculty and Staff
Cardinal Warde
Margaret Eminian

Visiting Scientist
Kuang Yi Huang

Graduate Students
Jenq Yang Chang
Scott Hathcock
James Kottas
Doyle Temple

Undergraduate Student
Michael Frey