OPTICAL INFERENCE MACHINES

INTERIM REPORT, AFOSR-86-0301

June 27, 1988

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

8

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 88-0767	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Optical Inference Machines		5. TYPE OF REPORT & PERIOD COVERED Interim Report Aug. 15, 1986-Aug. 14, 1987
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Cardinal Warde		8. CONTRACT OR GRANT NUMBER(s) AFOSR-86-0301
9. PERFORMING ORGANIZATION NAME AND ADDRESS Room 13-3065, Dept of Electrical Engineering Massachusetts Institute of Technology 77 Massachusetts Avenue, Cambridge, MA 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Darpa 61102F 5780/00
11. CONTROLLING OFFICE NAME AND ADDRESS Defense Advanced Research Projects Agency 1400 Wilson Blvd. Arlington, VA 22209 ATTN: Dr. John Neff		12. REPORT DATE June 24, 1988
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Air Force Office of Scientific Research Bolling Air Force Base, D.C. 20332 Dr. Lee Giles		15. SECURITY CLASS. (of this report) unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Related information published in Applied Optics, 25, 940, 1986

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Optical Artificial Intellegence; Optical inference engines; Optical logic; Optical information processing.

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

We have been investigating an optical inference machine based on the mapped-template architecture (C. Warde and J. A. Kottas, Appl. Opt. 25, 940, 1986). During this first year, two subsystems for use in our study of this machine were developed: (1) a portable electron-beam-addressed microchannel spatial light modulator (e-beam MSLM) and (2) a computerized control system for the architecture. This report describes the mapped-template inference machine and the development of its components in the context of the overall system architecture.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

TABLE OF CONTENTS

OPTICAL INFERENCE MACHINES

ABSTRACT

We have been investigating an optical inference machine based on the mapped-template architecture (C. Warde and J. A. Kottas, Appl. Opt. 25, 940, 1986). During this first year, two subsystems for use in our study of this machine were developed: (1) a portable electron-beam-addressed microchannel spatial light modulator (e-beam MSLM) and (2) a computerized control system for the architecture. This report describes the mapped-template inference machine and the development of its components in the context of the overall system architecture.

INTRODUCTION

Symbolic logic problems involve, in an abstract sense, a set of data objects 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.

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.

Symbolic logic problems are relatively common. They arise in areas such as expert systems and other artificial intelligence systems. In recent years, the computer science language PROLOG has become one of the tools used for solving these types of problems on electronic computers.[1] For example, two goals of fifth-generation computers are (1) to develop a machine capable of logical inference and database operations, and (2) to design a language based on PROLOG that would be suitable for inferencing and representing knowledge.[2]

To solve a query, electronic PROLOG sequentially searches the knowledge base for the appropriate rules and facts. This search process uses a flexible pattern-matching technique called unification which involves searching, matching, and backtracking through the knowledge base.[3,4] The performance of electronic PROLOG is limited by its use of serial searching and backtracking techniques. PARALOG, an implementation of PROLOG which uses

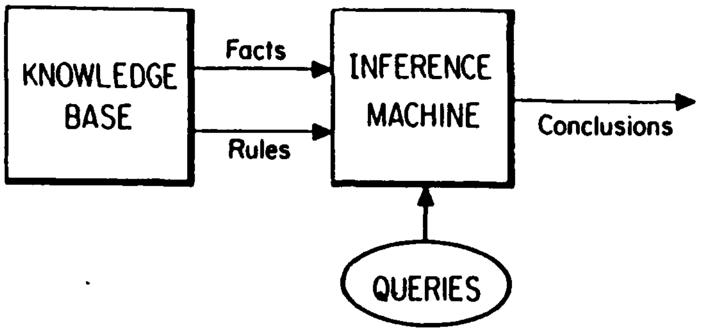


Fig. 1- General structure of an inference machine.

parallel unification, addresses this issue and is currently under investigation.[2]

It is well known that two-dimensional parallel optical processors inherently perform high-speed pattern matching. Such systems should therefore be more efficient at searching than their serial electronic counterparts because the parallelism eliminates the need for backtracking through the knowledge base. Furthermore, since searching and pattern matching processors do not require high accuracy or large dynamic range, it would appear that optical processors should also be well suited for symbolic logic processing

Optical inference machines, however, should be designed to be compatible with electronic computers. The goal should be to exploit the strengths of both systems so as to realize hybrid inference machines that are more efficient and versatile than either purely electronic or optical computers. For example, an optical inference machine could potentially be integrated into an electronic fifth-generation computer so that a hybrid machine capable of operating at speeds in excess of 10^9 logical inferences per second could be produced.

APPROACH

Hybrid Optical Realizations

The research program is focused on hybrid optical inference machines that would complement the electronic computer. More specifically, on the development of the query-driven system based on mapped-template logic that is described in the APPENDIX. In this system, the parallelism and speed of optics are exploited to perform the functions of searching, matching, and logic. The role of the electronics is to perform information storage and to retrieve and transfer data, rules, and operator queries to the optical processor. Thus in Fig. 2, the inference filter is the optical processor while the controller and knowledge base form the electronic support system.

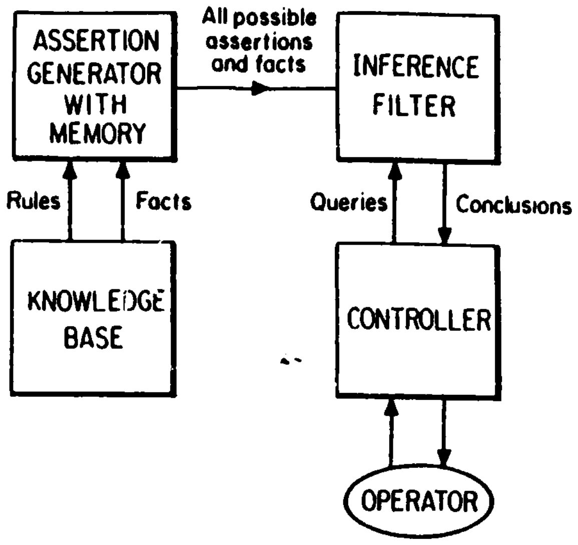


Fig. 2- **Block diagram of a deductive query-driven inference machine.**

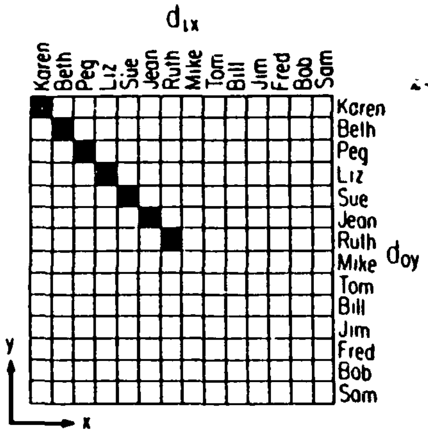
To implement these optical inference machines, three types of optical devices are required: (1) an input interfacing device which converts electrical signals to two-dimensional optical signals, (2) an optical logic device, and (3) an output interfacing device for transforming optical signals into electrical signals. The input interfacing device and the optical logic device should be capable of short term storage.

In the mapped-template discussed below, the electrical-to-optical input device could be any two-dimensional electrically-addressed spatial light modulator (E-SLM) which has short term storage, such as the e-beam MSLM.[6] An example of an optical logic device which can perform two-dimensional logic with memory is the photo MSLM[7,8] which is an optically-addressed spatial light modulator (O-SLM). The logic operations that can be performed internally by the photo MSLM include AND, OR, NAND, NOR, XOR, and NOT. The optical-to-electrical output device is a two-dimensional photodetector array. To obtain good noise rejection and low error rates, digital optical signals (binary intensity levels) are assumed for all input and output signals in the optical processor.

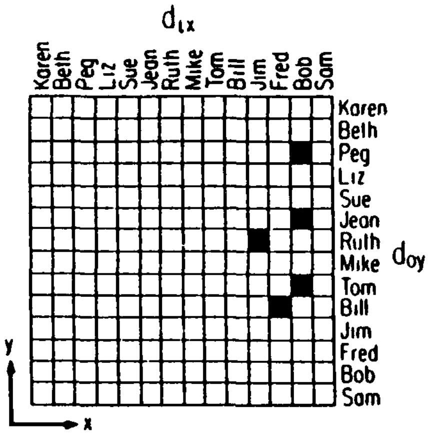
Mapped-Template Optical Inference Machine

In the mapped-template optical inference machine, mapping templates are used to optically store the relationships between the data objects and are thus defined by the facts. Conclusions are inferred to queries by applying these mapping templates to the data objects in the order prescribed by the rules. Historically, the operation of the mapping templates is similar to the associative nets described by Willshaw and Longuet-Higgins.[9]

Mapping templates are binary masks consisting of transparent squares (logical 1, and shown as black squares in Fig. 3) on an opaque background (logical 0, and shown as white in Fig. 3). As an example, a mapping template for the relations **is-male**, **is-female**, **married-to**, and **father-of** would map an input set from **D**, the set of data objects to an output set, also from **D**. Let $\mathbf{D}_i$ and $\mathbf{D}_o$ represent the input and output sets of data objects. Furthermore, let the data objects in the mth position of $\mathbf{D}_i$ and $\mathbf{D}_o$ be denoted by d_{im} and d_{om}. Mapping templates corresponding to the **is-female** and **father-of** facts as defined are shown in Fig. 3 with the elements of the input set $\mathbf{D}_i$ (d_{ix} along the columns (x axis) and the elements of the output set $D_o(d_{oy})$ along the rows (y axis) of the templates.



(a) d_{ix} **is-female** d_{oy}



(b) d_{ix} **father-of** d_{oy}

Fig. 3 - **Mapping Templates for (a) the is-female facts and (b) the father-of facts for the entire set of data objects in Eq. (1).**

The interpretation of these templates is as follows: A transparent square in the (x,y) position of, say **father-of** indicates the fact

$$d_{ix} \textbf{ father-of } d_{oy} \tag{1}$$

Given these two templates, the mapping templates for **is-male** and **married-to** are straightforward to generate.

Note that the mapping between $\mathbf{D}_i$ and $\mathbf{D}_o$ is not necessarily one-to-one. However, a mapping template is reciprocal in that if the right-hand side of Eq. (1) is specified instead of the left, the relationships for the left-hand side may be inferred from the template.

To perform logical inferring, the template-mapping concept is implemented as illustrated in Fig. 4. Given an input vector $\mathbf{D}_i$, the associated output vector $\mathbf{D}_o$ for a particular mapping template is found by first vertically expanding $\mathbf{D}_i$ along the y-axis so it forms an array, each row of which equals $\mathbf{D}_i$, as shown in Fig. 4. This expanded form of $\mathbf{D}_i$ is then optically overlaid with the mapping template using imaging optics as shown, before the two-dimensional logical AND operation is performed. The resulting output, when viewed along the rows, corresponds to the output vector $\mathbf{D}_o$.

To perform the reciprocal operation of the mapping template, the input vector would be expanded horizontally and logically ANDed with the mapping

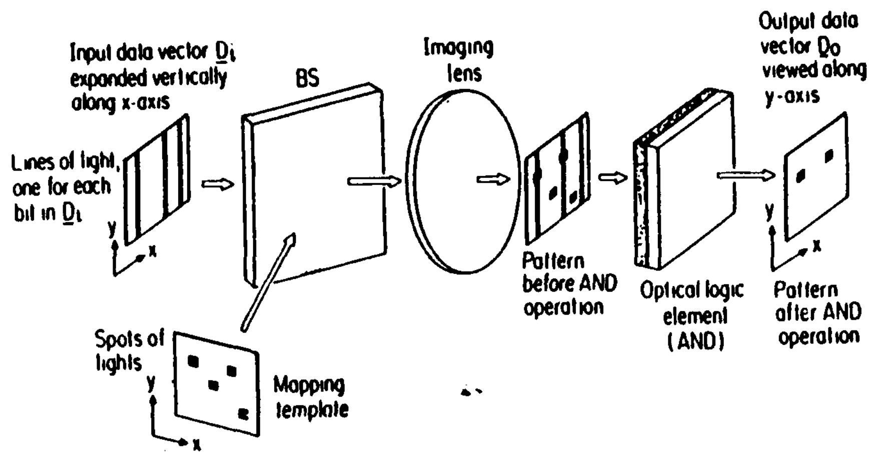

Input data vector Di expanded vertically along x-axis
Lines of light, one for each bit in Di
y
x
BS
Imaging lens
Spots of lights
y
x
Mapping template
Pattern before AND operation
Optical logic element (AND)
Output data vector Do viewed along y-axis
y
x
Pattern after AND operation

Fig. 4 - Conceptual implementation of mapped-template logic.

template. The output vector would then be taken looking down the columns. Depending upon the mapping template, it is possible for multiple inputs in D_i to produce the same output element in D_O. For this reason, a two-dimensional output photodetector array is used for establishing the exact input-to-output correspondence, should this be needed in solving the query.

A hybrid optical inference machine which implements mapped-template logic is shown in Fig. 5. It consists of an electronic controller, two E-SLMs, two O-SLMs, and a two-dimensional photodetector array. The controller in this system electronically stores the knowledge base and controls the SLMs and the shutter. The modulator O-SLM 1 is operated in the logic mode and usually performs the AND operation, while O-SLM 2 is used as a two-dimensional memory unit to allow further processing of the outputs, and is optional.

When the controller is given a query by the operator, a vertical line is written on E-SLM 1 at the location of the known data objects in D_i. Then, the controller writes the mapping template corresponding to the rule (or first condition) associated with the query onto E-SLM 2. The outputs of both E-SLMs are imaged onto O-SLM 1 with lens L_1. The logical AND of the two inputs is formed in O-SLM 1 and imaged onto the photodetector array by lenses L_2 and L_3. If desired, the output could also be imaged onto O-SLM 2 by lens L_2 and latched. The stored output in O-SLM 2 could then be imaged via lens L_4 back into O-SLM 1 by opening shutter S should further processing be necessary.

The output of the photodetector array is fed back to the controller where the inferred data objects in D_O which satisfy the current mapping rule are determined. Further mapping templates are then applied by the controller as determined by the query and the rules.

Since multiple outputs for the same data object could be generated, the viewing of the rows or columns of the output array could lead to an integral multiple of a single light beam intensity. In this case, the photodetector output is electronically clipped to the single light beam level if the

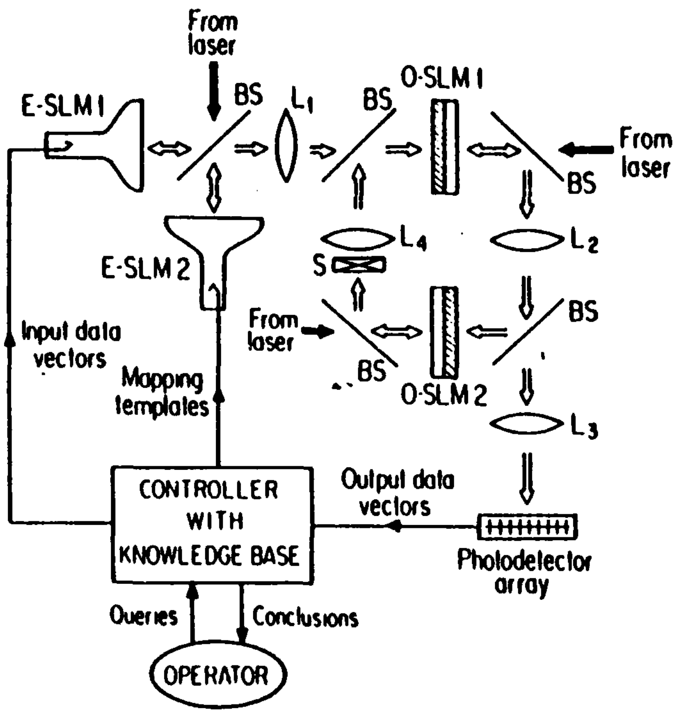


Fig. 5 - Mapped-template optical inference machine.

photodetector output is to be fed back to E-SLM 1 as input via the controller.

If the optional optical feedback loop is not used, there is a possible modification to the system in Fig. 5 which will simplify the device requirements. Instead of performing the logical AND operation in O-SLM 1, the output of E-SLM 1 (the expanded input data vector) could be used to read out the mapping template in E-SLM 2, thus eliminating the need for a two-dimensional optical logic device. However, the advantage of having O-SLM 1 is that (1) it can conveniently perform the logical NOT operation on a condition, and (2) the processed patterns are automatically latched into O-SLM 1. This allows the controller to begin setting up the next mapping template while it simultaneously reads the photodetector array, thus providing some degree of concurrent operation.

Further possibilities for increasing processing speed are to place multiple mapping templates which are spatially separated from each other on E-SLM 2. The input data vectors on E-SLM 1 would have to be repositioned accordingly. However, multiple inferences could then be made in parallel.

FIRST-YEAR GOALS

The goals of the first year of the program were to develop the subsystems necessary to implement the mapped-template inference machine described in the Appendix. These components included a portable electron-beam-addressed microchannel spatial light modulator (e-beam MSLM) and a computerized control system (CCS) for the inference machine. The e-beam MSLM is to be used as an electrical-to-optical input device for presenting input queries to the machine along with the applicable rules from a knowledge base. The CCS will perform several tasks:

1. Store the knowledge base (the mapping templates).
2. Allow the user to query the system.
3. Return conclusions to the user.
4. Govern the low-level operation of the inference machine by passing the applicable rules and facts to the inference engine (through two e-beam MSLM's) and then sequencing all MSLM's in the proper order to generate the conclusions.
5. Monitor the state of the inference machine and its components.
6. Provide the necessary power supplies for all MSLM's (both optical and electronic).

During this year, these two subsystems were constructed and tested successfully. Their development is described in the following section.

SYSTEM DEVELOPMENT

Portable E-Beam MSLM

A general e-beam MSLM is shown schematically in Fig. 6. The function of the device is to convert a serial electronic signal into a two-dimensional optical signal. To achieve this, the electronic signal is used to control a focused electron beam (produced by the electron gun in Fig. 6) so as to deposit a charge distribution onto the back of an electrooptic crystal. A microchannel plate (MCP) is included to amplify the electron beam intensity. The effects of the charge distribution can be encoded onto an optical beam by reading out the device with polarized coherent light. After the light passes through a crossed polarizer, the optical intensity distribution is representative of the charge pattern. Thus, a two-dimensional optical signal is generated from a serial electronic signal. For more information, a detailed description of the operational principles of the e-beam MSLM (and the related device, the optically-addressed MSLM) can be found in Refs. 6-8.

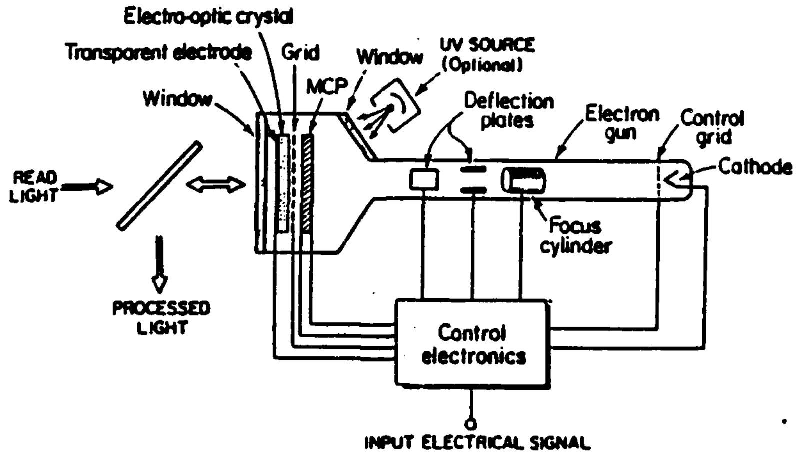


Fig. 6 - Electron-beam-addressed microchannel spatial light modulator (e-beam MSLM).

The mapped-template architecture uses two e-beam MSLM's and we already have one such device. However, it is heavy and bulky and cannot be repositioned easily due to its size and the placement of its external vacuum system. Consequently, in the first year of the program, we constructed a second device is smaller, easier to align, and portable.

The design for the portable e-beam MSLM is shown in Figure 7. It consists of three main sections supported by an adjustable base. The modulator section (on the left side in Fig. 7) contains the electrooptic crystal, the MCP, and an acceleration grid. The electron gun is housed in the section directly behind the modulator and a small vacuum pump is positioned in the section below the electron gun. To minimize the effects of the pump's magnet, the pump is mounted slightly behind the electron gun. Furthermore, the horizontal tube in which the electron beam propagates is wrapped with highly magnetically permeable metal (not shown in Fig. 7).

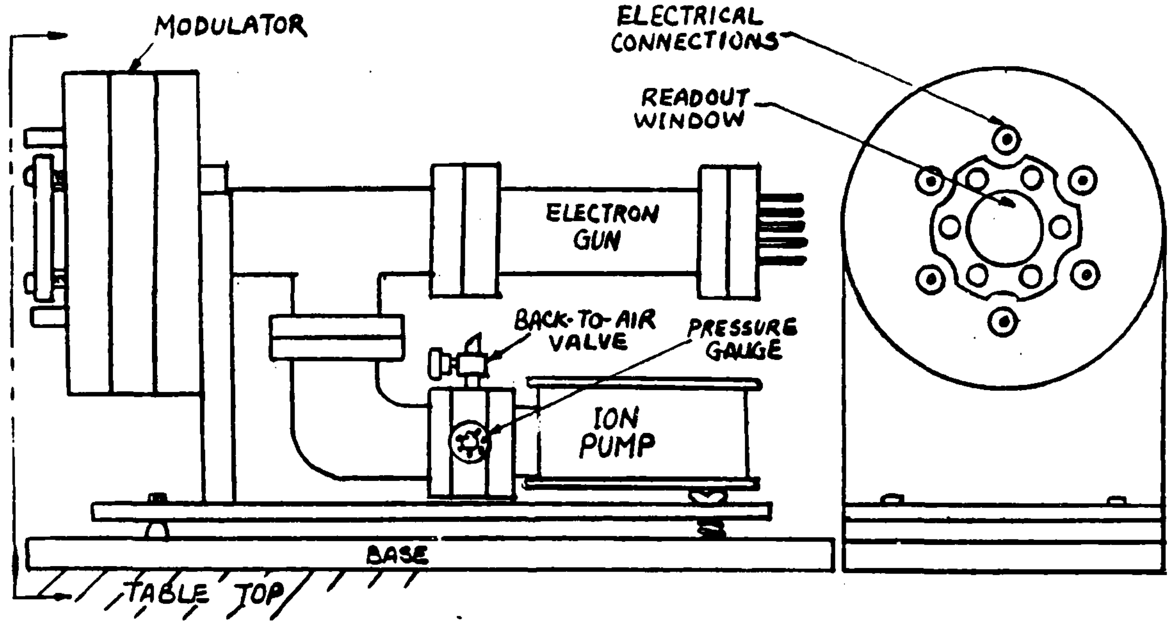


Fig. 7 - Design for the portable e-beam MSLM.

A particularly useful feature of this design is the adjustable base. It allows the position of the entire modulator to be adjusted conveniently while the device is in place on an optical table. The degrees of freedom are shown in Fig. 8. As can be seen, θ_1 and θ_2 allow the crystal to be aligned easily and properly to the readout laser beam, while θ_3 permits the axes of the crystal to be oriented easily at 45 degrees to the polarization of the incoming readout light.

In the modulator section, the crystal, grid, and MCP are mounted in a sandwich structure as illustrated in Figure 9. The stainless steel flanges provide structural support and act as electrodes for the crystal, the grid, and the input face of the MCP. For the output face, a thin nickel flange is used to reduce the gap between the grid and the MCP output face. The sandwich arrangement is held together with three sets of screws which ease the assemblying procedure and minimize the physical interaction with these delicate optical components.

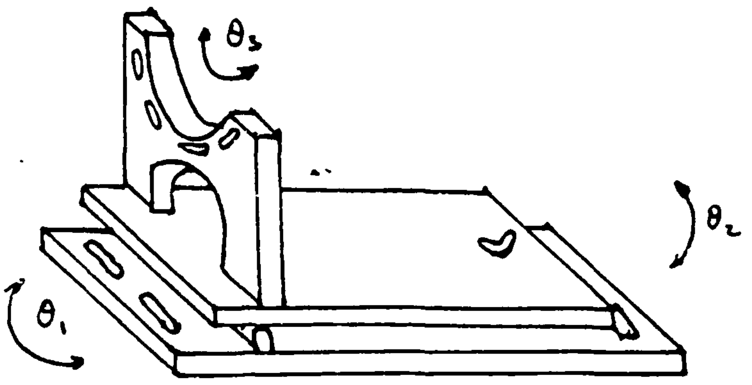


Fig. 8 - Adjustable base for the portable e-beam MSLM.

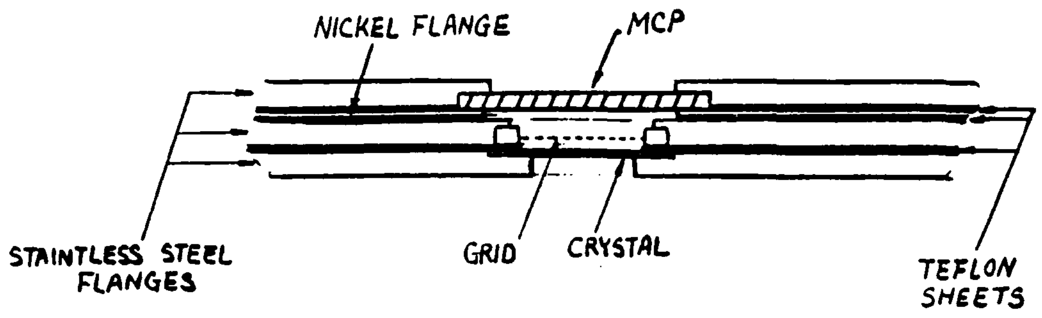


Fig. 9 - Cross-section of the internal support structure for the crystal, the grid, and microchannel plate (MCP).

The completed e-beam MSLM is shown in Fig. 10. To test the device, the setup shown in Fig. 11 was used in conjunction with the computerized control system discussed in the next section. A sample output image from the device is given in Fig. 12. This image consists of a 16-by-16 pixel array with the appropriate rows and columns turned on to form a 3-by-3 array of large dark areas. The device worked fairly well but it only could create low resolution images. This was due to a faulty electron gun that could not produce a uniform spot over the extent of the crystal. Furthermore, the beam exhibited significant halo (a distribution of charge away from the main beam). Both of these effects reduced the quality of the output image, as evidenced by the nonuniformity of the image in Fig. 12. However, with a new electron gun, the portable e-beam MSLM will be a very useful device for the mapped-template inference machine.

Fig. 10 - Photograph of the completed portable e-beam MSLM.

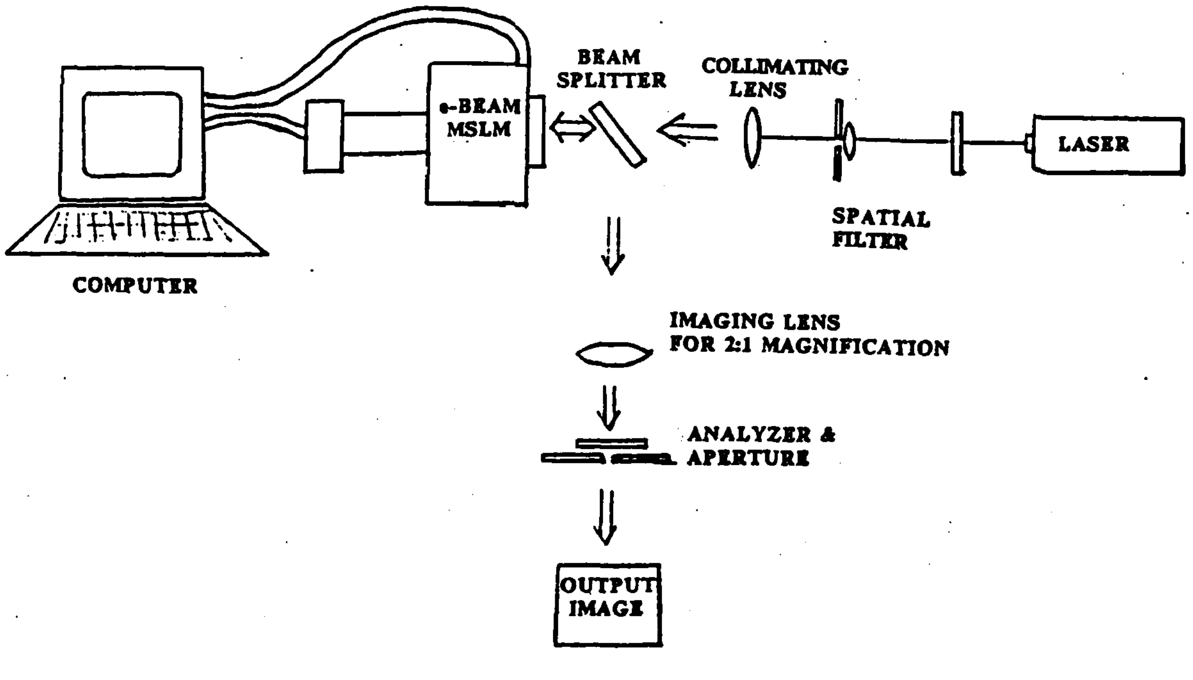


Fig. 11 - Optical readout setup for testing the e-beam MSLM.

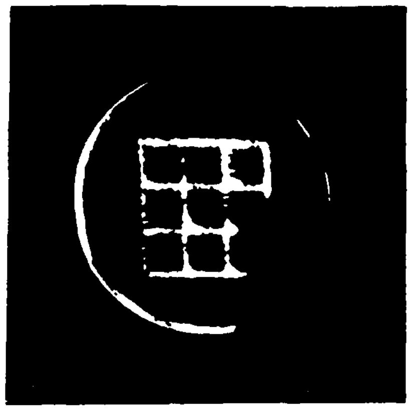


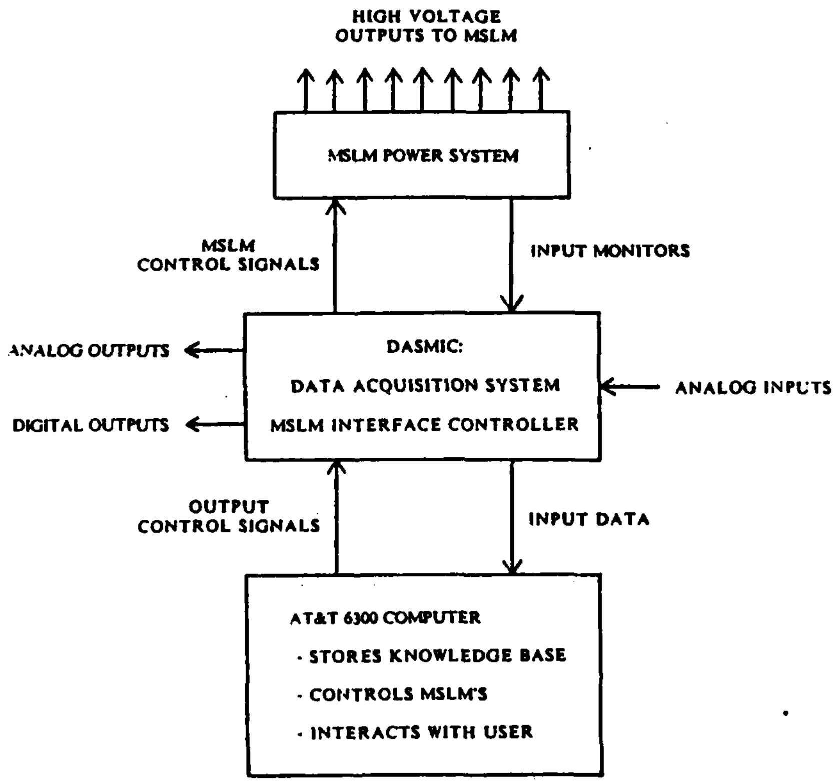
Fig. 12 - Sample output image from the e-beam MSLM.

Computerized Control System (CCS)

Because the CCS must perform several tasks, it is designed as three modules, each of which focuses on a particular function. The modules are:

1. Computer controller
2. MSLM power supply
3. Data Acquisition System and MSLM Interface Controller (DASMIC)

The relationship between the three modules is illustrated in Fig. 13.



F'g. 13 The three modules comprising the computerized control system (CCS).

In the mapped-template architecture, the computer controller is a small, dedicated computer (an AT&T PC 6300) which stores the knowledge base, controls the MSLM's, and interacts with the user. The second task here is done using the second and third modules in the above list. The MSLM power supply module contains several individual high-voltage power supplies for biasing both an e-beam MSLM and an optically-addressed MSLM (herein abbreviated as an optical MSLM). The data acquisition module, nicknamed DASMIC (for Data Acquisition System and MSLM Inferface Controller), provides several digital-to-analog (D/A), analog-to-digital (A/D), and digital-to-digital (D/D) ports under computer control. It also incorporates a customized photodetector input with a computer-controllable gain. In addition, DASMIC contains a special bus that allows a computer to access several different MSLM power supplies. With the appropriate software, the computer controller can access all of the capabilities of either type of MSLM through DASMIC and an MSLM power supply.

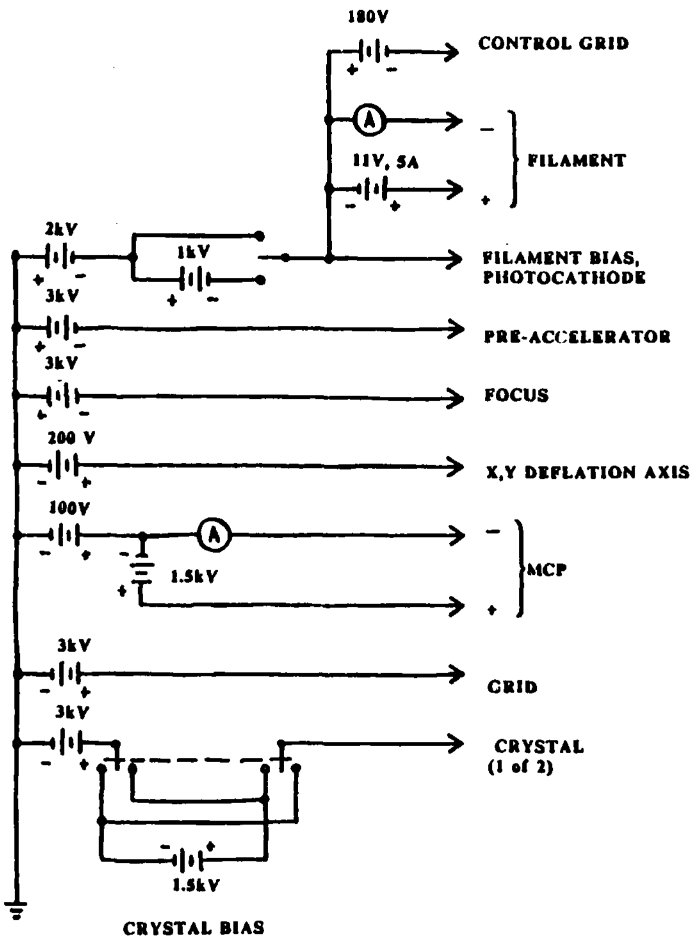


Fig. 14 - Equivalent circuit for the MSLM power supply.

For the mapped-template inference machine, only one MSLM power supply was needed. An equivalent circuit of the MSLM power supply constructed during the past year is shown in Fig. 14. Table I lists individual power supplies (corresponding to the various components within an MSLM) that are contained in this unit.

TABLE I

	MSLM		
Supply Name	Type	Control	Range
Filament	E	M	11 V at 5 A Cathode
- Filament Bias	E	M	-2 kV
- Photocathode	O	M	-3 kV
Control Grid	E		-170 V
- On/Off Mode		C	
- Pulse Mode		M, C	
- Pulse Duration		M	1 to 70 msec
Preaccelerator	E	M	-2 kV
Focus	E,O	M,C.	-2 kV
X,Y Deflection Axes	E	M,C.	-100 to +80 V w.r.t. MCP Input
- Range		M	180 V
- Maximum Offset		M	105 V
MCP	E,O	M,C	+1.5 kV
- Current Limiter	E,O	M	0.5 mA
- Input Bias	E	M	100 V
Grid	E,O	M,C	3 kV
Crystal (2)	E,O	M,C	3 kV
- Bias	E,O	M,C	1.5 kV
- Bias Polarity	E,O	M	+ or -
- Ramp to New Level	E,O	C	up or down to full range
- Variable Ramp Rate	E,O	C	0.25 to 64 sec for 3 kV ramp
Supply Status Indicators			
- Filament current	E	C	1 = ok, 0 = below 3.8 A
- MCP current	E,O	M,C	0 = ok, 1 = at limit

Notes:

O =	Applicable for optically-addressed MSLM
E =	Applicable for e-beam MSLM
C =	Can be controlled via computer
M =	Can be controlled manually

The table indicates which individual supplies and their associated functions are applicable to e-beam and optical MSLMs and also if they can be controlled via a computer. For the mapped-template machine, two identical crystal supply circuits were included in the box for controlling the two e-beam devices in the architecture. (The third device, an optical MSLM, can be controlled using DASMIC and a small high-voltage supply which we have already procured.)

The MSLM power supply has been tested on numerous occasions and has been quite useful and flexible for controlling the operation of both e-beam and optical MSLMs in optical processing systems.

The features of DASMIC, the data acquisition module, are shown in Figure 15 and are summarized in Table II:

TABLE II

Data Acquisition Function	Qty.	Analog Range	Digital Range
Analog Inputs	16	0 to 10 V	0 to 255 Analog Outputs
(bipolar)	4	0 to 5 V -5 to 0 V	0 to 255
Digital Outputs	4	0, 5 V	0, 1
	4	0, 15 V	0, 1
Photodetector Amplifier Input	1	0 to 10 V	0 to 255
- Gain	1	1 to 40	0 to 255 (nonlinear)

The array of inputs and outputs represent standard capabilities required by the mapped-template architecture. However, although it is not needed in this architecture, the photodetector amplifier with a computer-controllable gain was included in the design for future optical processing experiments. It will be useful in systems which will need an optical measurement with a variable gain to establish an operating point or stabilize the system.

In addition to these data acquisition functions, DASMIC allows the computer controller to access up to four MSLM power supply boxes for future growth.
As with the MSLM power supply, DASMIC has been used many times in optical processing systems to monitor current and voltage levels and control shutters and separate power supplies as well as the MSLM power supply.

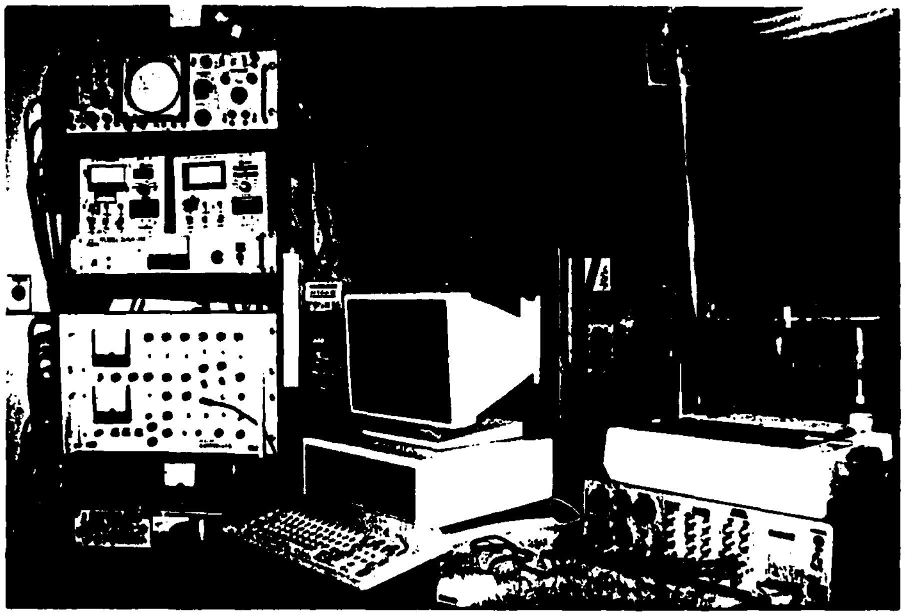


Fig. 15 The complete CCS. The computer controller is the AT&T PC 6300 in the center. The MSLM power supply is the large box to the left of the 6300, and DASMIC is in the lower right hand corner.

A picture of the complete CCS is shown in Figure 15. Many different programs have been written for the AT&T PC 6300 computer controller to exercise the e-beam MSLM using DASMIC and the MSLM power supply. While not optimized for speed and throughput, the CCS is a highly versatile system for controlling optical processing systems made up of MSLMs.

FUTURE WORK

In the near term, there are primarily three tasks which must be completed before a working mapped-template inference machine can be demonstrated:

1. Replace the electron gun in the portable e-beam MSLM with one that has a fine spot size. This will eliminate the current problems with this device. We currently are searching for acceptable electron guns and their distributors.
2. Construct a two-dimensional photodetection system. In the mapped-template architecture, the CCS must be able to access the full two-dimensional light distribution in the optical output. The general photodetector amplifier in DASMIC only can examine the intensity at a single pixel. We will examine the use of a video camera and a frame grabber for the computer controller module of the CCS to solve this problem.
3. Write the necessary software for the CCS for controlling the MSLM's in the mapped-template architecture in conjunction with the electronically stored knowledge base.

At the theoretical level, we will be exploring ways of making the mapped-template inference machine more robust, more fault tolerant, and less memory intensive. As currently designed, the architecture uses one beam (pixel) of light per data object in the knowledge base. This makes the output conclusions more susceptible to errors if a problem develops with a pixel in the optical system. The inference machine also relies heavily on electronic storage and retrieval. The electronic-to-optical conversions may pose significant bottlenecks in the processing speed. As a result, we will be studying these problems with the mapped-template architecture to try to solve them in the coming year.

REFERENCES

1. D. S. Nau, "Expert Computer Systems," IEEE Comput. 63 (Feb. 1983).
2. T. Moto-oka and H. S. Stone, "Fifth-Generation Computer Systems: A Japanese Project," IEEE Computer, March 1984, pp. 6-13.
3. K. L. Clark and F. G. McCabe, *Micro-Prolog: Programming in Logic*, Englewood Cliffs, New Jersey: Prentice-Hall, 1984.
4. W. F. Clocksin and C. S. Mellish, *Programming in Prolog*, New York: Springer-Verlag, 1981.
5. C. Warde and J. A. Kottas, "Hybrid Optical Inference Machines: Architectural Considerations," Appl. Opt. 25, 940 (1986).
6. A. Schwartz, X. Y. Wang, and C. Warde, "Electron-Beam-Addressed Microchannel Spatial Light Modulator," Opt. Eng. 24, 119 (1985).
7. 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).

8. C. Warde and J. I. Thackara, "Oblique-Cut $LiNbO_3$ Microchannel Spatial Light Modulator," Opt. Lett. 7, 344 (1982).

9. D. J. Willshaw and H. C. Longuet-Higgins, "Associative Memory Models," Machine Intelligence 5, 351 (1970).

LIST OF PERSONNEL

Faculty and Staff

Professor Cardinal Warde

Visiting Scientists

Kuang Yi Huang
Horatio Lamela Rivera

Graduate Students

Ala Alryyes
Jenq Yang Chang
Scott Hathcock
James Kottas
Suzanne Lau
Doyle Temple

Undergraduate Students

Phillip Mak

APPENDIX

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

Hybrid optical inference machines: architectural considerations

Cardinal Warde and James Kottas

A class of optical computing systems is introduced for solving symbolic logic problems that are characterized by a set of data objects and a set of relationships describing the data objects. The data objects and relationships are arranged into sets of facts and rules to form a knowledge base. The solutions to symbolic logic problems involve inferring conclusions to queries by applying logical inference to the facts and rules. The general structure of an inference machine is discussed in terms of rule-driven and query-driven control flows. As examples of a query-driven inference machine, two hybrid optical system architectures are presented which use matched-filter and mapped-template logic, respectively.

I. Introduction

A. Definitions

Symbolic logic problems involve, in an abstract sense, a set of data objects 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 so that the statement is always interpreted as 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 relationships and 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.

B. PROLOG

Symbolic logic problems are relatively common. They arise in areas such as expert systems and other artificial intelligence applications. In recent years, the computer science language PROLOG has become a tool for solving these types of problem on electronic computers.[1] For example, two goals of fifth-generation computers are (1) to develop a machine capable of logical inference and data base operations and (2) to design a language based on PROLOG that would be suitable for inferring and representing knowledge.[2]

To solve a query, electronic PROLOG sequentially searches for the knowledge base for the appropriate rules and facts. This search process uses a flexible pattern-matching technique called unification which involves searching, matching, and backtracking through the knowledge base.[3,4] The performance of electronic PROLOG is limited by its use of serial searching and backtracking. PARALOG, an implementation of PROLOG which uses parallel unification, addresses this issue and is currently under investigation.[2]

C. Role of Optics

It is well known that 2-D parallel optical processors inherently perform high-speed pattern matching. Such systems should, therefore, be more efficient at searching than their serial electronic counterparts because the parallelism eliminates the need for backtracking through the knowledge base. Furthermore, since searching and pattern matching processors do not require high accuracy or large dynamic range, optical processors should in principle be well suited for symbolic logic processing.

We believe, however, that optical inference machines should be designed to be compatible with electronic computers. The goal should be to exploit the strengths of both systems so as to realize hybrid inference machines that are more efficient and versatile than either purely electronic or optical computers. For example, an optical inference machine could potentially be integrated into an electronic fifth-generation computer so that a hybrid machine capable of operating at speeds in excess of 10^9 logical inferences per second (LIPS) could be produced.

The authors are with MIT Department of Electrical Engineering & Computer Science, Cambridge, Massachusetts 02139.
Received 19 June 1985.
0003-6935/86/060940-08$02.00/0
© 1986 Optical Society of America.

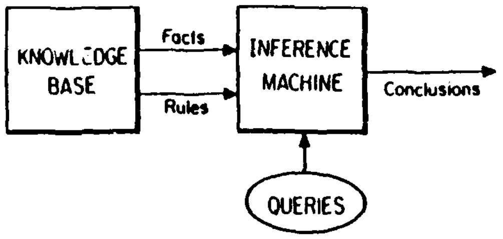


Fig. 1. General structure of an inference machine.

D. History

Previous work in optical symbolic processing was performed by several researchers in the late 1960s and early 1970s. Gabor,[5] Akahori and Sakurai,[6] Nakajima *et. al.*,[7] and Lohmann and Werlich[8] used holography as the basis for their processing techniques. Willshaw *et. al.*,[9] Willshaw and Longuet-Higgins,[10] and Gabor[5] approached the problem using associative network concepts. However, during the 1970s and early 1980s, the emphasis of research on optical computing systems shifted to numerical problems such as matrix–matrix multiplication,[11–13] array processing,[14] and solving sets of linear equations.[15]

More recently, there has been a resurgence of interest in the area of optical symbolic processing. Huang[16,17] has addressed the symbolic problem in a general sense, investigating algorithms and architectures for performing symbolic substitution optically in classical finite-state machines. Furthermore, Huang[18] and Fisher *et al.*[19] have recognized that there may be a possible role for optics in symbolic processors, particularly in solving certain classes of artificial intelligence problems. However, specific applications of optical computers to symbolic logic processing appear, until now, to have been unaddressed.

In this paper, the concepts associated with symbolic logic processors are introduced, and the general architecture of an optical machine capable of inferring logical conclusions from a set of facts and rules is discussed. The general system is approached from two different information flow patterns: rule-driven and query-driven flow. Two hybrid optical realizations for a query-driven inference machine are presented which use classical matched-filter logic and mapped-template logic, respectively. The intent here is to describe these systems from a conceptual point of view. Therefore, no attempt is made to address all the issues involved in realizing a practical system.

II. General Inference Machine Architecture

The general structure of an inference machine is shown in Fig. 1. It accepts as input 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.

For example, a set of data objects could be a set of names of people. For illustrative purposes, let this set be denoted as

$$\mathbf{D} = \{\text{Karen, Beth, Peg, Liz, Sue, Jean, Ruth, Mike, Tom, Bill, Jim, Fred, Bob, Sam}\}. \quad (1)$$

A set of relationships for **D** might be the possible relationships between the people, such as marriage, mother, father, male, and female. Let this set of relationships be denoted by

$$\mathbf{R} = \{\textbf{married-to, mother-of, father-of, son-of, daughter-of, child-of, is-male, is-female}\}. \quad (2)$$

The data objects and relationships are linked as a collection of facts and rules which relate the elements of **D** and **R**. In this example, the facts could be defined as

Mike **is-male.**	Karen **is-female.**
Tom **is-male.**	Beth **is-female.**
Bill **is-male.**	Peg **is-female.**
Jim **is-male.**	Liz **is-female.**
Fred **is-male.**	Sue **is-female.**
Bob **is-male.**	Jean **is-female.**
Sam **is-male.**	Ruth **is-female.** (3)
Mike **married-to** Karen.	Bob **father-of** Peg.
Bob **married-to** Beth.	Bob **father-of** Tom.
Jim **married-to** Liz.	Bob **father-of** Jean.
	Jim **father-of** Ruth.
	Fred **father-of** Bill.

Using these facts, the remaining relationships in **R** may be defined as rules. For example,

$$\begin{aligned} &X\ \textbf{mother-of}\ Y \ \text{ IF } && Z\ \textbf{married-to}\ X \text{ AND } Z\ \textbf{father-of}\ Y,\\ &X\ \textbf{child-of}\ Y \ \text{ IF } && Y\ \textbf{mother-of}\ X \text{ OR } Y\ \textbf{father-of}\ X\\ &X\ \textbf{son-of}\ Y \ \text{ IF } && X\ \textbf{child-of}\ Y \text{ AND } X\ \textbf{is-male},\\ &X\ \textbf{daughter-of}\ Y \ \text{ IF } && X\ \textbf{child-of}\ Y \text{ AND } X\ \textbf{is-female}, \end{aligned} \quad (4)$$

where *X*, *Y*, and *Z* are variables. The bodies of these rules (i.e., the part to the right of IF) consist of two conditions, each of which could be a fact or another rule. These conditions are then connected by the logical operators AND or OR. In general, a rule could have any number of conditions, and a condition could have a logical NOT operation performed on it. For example, the **daughter-of** rule could be modified to use the **son-of** rule by defining it with

$$X\ \textbf{daughter-of}\ Y \text{ IF } X\ \textbf{child-of}\ Y \text{ AND NOT } X\ \textbf{son-of}\ Y. \quad (5)$$

To satisfy a rule, there must be at least one data value for all variables for which all conditions are simultaneously true. In the **mother-of** rule, there must be at least one value each for *X*, *Y*, and *Z* so that *Z* is both married to *X* and the father of *Y*. Using the format of Eq. (4), additional relationships such as **sister-of** and **brother-of** are straightforward to define. Together, the facts in Eq. (3) and the rules in Eq. (4) form the knowledge base.

In general, a query into a knowledge base consists of a rule with at least one variable. For example, a possible query of this knowledge base could be "Who is the mother of Jean?", which can be expressed as

$$? \text{ mother-of Jean,} \quad (6)$$

where ? represents the desired unknown data object. From the knowledge base, only the assertion Beth **mother-of** Jean is true. Hence the conclusion of Eq. (6) is that the query is true when ? is the data object Beth.

Given a query and knowledge base, conclusions can be inferred using either inductive or deductive reasoning. In the inductive case, conclusions of a general nature are inferred by the application of specific queries to the knowledge base. The cardinality of the set of induced conclusions could in general be quite large, and, in principle, conclusions not representative of the knowledge base would be possible.

On the other hand, deductive reasoning produces specific conclusions from a set of general rules and facts, and the conclusions are always a subset of the knowledge base. For simplicity and practicality, we shall limit the allowed conclusions to the data objects within the knowledge base. Therefore, in this paper, we will consider only machines based on deductive reasoning.

Block diagrams for two general architectures of a deductive inference machine are shown in Figs. 2 and 3. Both systems have in common a knowledge base, controller, and inference filter. The functions of the controller are to (1) control the flow of information through the inference machine, (2) accept queries as input from the operator, and (3) transmit conclusions to the operator as output. The knowledge base stores all the data objects and relationships in the form of facts and rules. The role of the inference filter is to generate a set of all conclusions possible given a set of rules and facts from the knowledge base.

The system in Fig. 2 corresponds to a rule-driven inference machine, whereas that in Fig. 3 represents a query-driven inference machine. The systems are distinguished from each other by the methods they employ to infer the conclusions. In the rule-driven system, all possible assertions and facts from the knowledge base are generated *ab initio*, and thereafter the conclusions are derived from these inferences by application of the query. In contrast, the query-driven system first uses the query to select appropriate subsets of the rules and facts and then infers specific conclusions from these rules and facts.

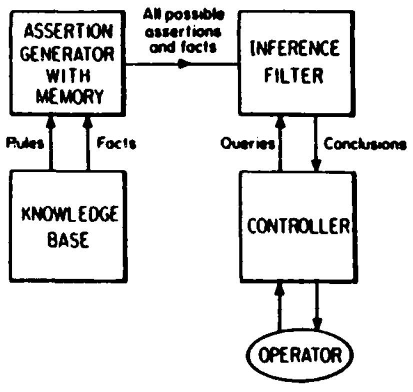


Fig. 2. Block diagram of a deductive rule-driven inference machine.

The rule-driven system of Fig. 2 approaches the ideal parallel system in that the assertion generator produces the facts and all possible assertions from the entire knowledge base by replacing all the rules with appropriate assertions. In the previous example, the **mother-of**, **child-of**, **son-of**, and **daughter-of** rules would lead to the assertions

Beth **mother-of** Peg,	Tom **son-of** Bob.	
Beth **mother-of** Tom,	Tom **son-of** Beth.	
Beth **mother-of** Jean,	Bill **son-of** Fred.	
Liz **mother-of** Ruth,		(7)
Peg **child-of** Bob,	Peg **daughter-of** Bob.	
Peg **child-of** Beth,	Peg **daughter-of** Beth.	
Tom **child-of** Bob,	Jean **daughter-of** Bob.	
Tom **child-of** Beth,	Jean **daughter-of** Beth.	
Jean **child-of** Bob,	Ruth **daughter-of** Jim.	
Jean **child-of** Beth,	Ruth **daughter-of** Liz.	
Ruth **child-of** Jim,		
Ruth **child-of** Liz,		
Bill **child-of** Fred,		

Thus the output of the assertion generator would be the set of facts and assertions defined by Eqs. (3) and (7). Note that the knowledge base is not updated by the assertion generator and that the output produced by the assertion generator is computed only once.

As shown in Fig. 2, the assertion generator of the rule-driven machine transfers the entire set of facts and assertions to an inference filter whose function is to match the queries from the controller with the facts and assertions to determine the data objects which satisfy the queries. After it has determined the conclusions for the query, the inference filter transfers the conclusions to the controller for output to the operator.

In the example ? **mother-of** Jean, the inference filter would compare the facts and assertions defined by Eqs. (3) and (7) with the query given in Eq. (6). Realizing that ? is the desired variable, the filter would find a match between the query with the third assertion given in Eq. (7) to obtain the answer Beth. In this example, there was only one possible conclusion, but, in general, several data objects may satisfy a query.

In contrast, the query-driven system of Fig. 3 is a more sequential machine than the rule-driven system of Fig. 2. Given a query from the operator, the controller uses the rules associated with the query to select subsets of rules and facts from the knowledge base that are relevant to the query. In the example of Eq. (6), the **mother-of** rule is associated with the query. The controller would examine the **mother-of** rule as defined in the knowledge base and extract its condition relationships **married-to** and **father-of**.

Once it has obtained the necessary subsets of rules and facts, the controller transfers these subsets to the inference filter along with the known data objects from the query [Jean in Eq. (6)]. The inference filter then matches the rules with the known query data to infer the set of data objects which make the query true [Beth for Eq. (6)]. Finally, the inference filter sends the

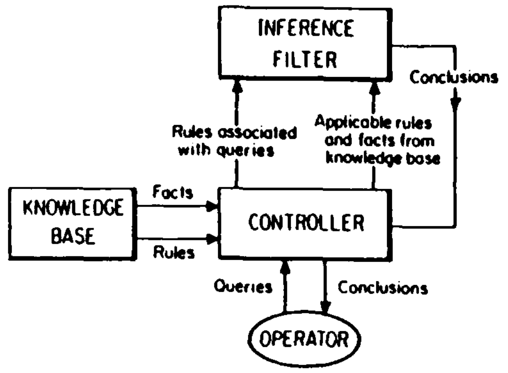


Fig. 3. Block diagram of a deductive query-driven inference machine.

conclusions back to the controller for output to the operator.

When consideration is given to implementation of an inference machine, the query-driven system may appear more attractive than the rule-driven system. This is because inferring the possible assertions and storing all the possible assertions and facts in the rule-driven system could be inefficient, expensive, and difficult to realize, particularly for rules which are recursively defined (i.e., when the rule has itself as a condition). Consequently, only query-driven systems are considered in the remaining sections.

III. Hybrid Optical Realizations

We shall confine our discussion to optical inference machines that complement the electronic computer. A complete system, therefore, will be hybrid in nature. This places design constraints on the input and output interfacing devices of the optical system. The optimum designs, therefore, are those that most effectively combine the individual strengths of optics and electronics. Two query-driven designs are described below, the first of which uses matched-filter logic in the inference filter, whereas the second is based on mapped-template logic.

In these systems, the parallelism and speed of optics are exploited to perform the functions of searching, matching, and logic. The role of the electronics is to perform information storage and retrieve and transfer data, rules, and operator queries to the optical processor. Thus in Fig. 3 the inference filter is the optical processor, while the controller and knowledge base constitute the electronic support system.

To implement these optical inference machines, three types of optical devices are required: (1) an input interfacing device which converts electrical signals to 2-D optical signals; (2) an optical logic device; and (3) an output interfacing device for transforming optical signals into electrical signals. The input interfacing device and optical logic device should exhibit at least short-term storage.

In the specific systems discussed below, the electrical-to-optical input device could be any 2-D electrically addressed spatial light modulator (E-SLM) which has short-term storage, such as the e-beam MSLM.[20] An example of an optical logic device which can perform 2-D logic with memory is the photo MSLM,[21,22] which is an optically addressed spatial light modulator (O-SLM). The logic operations that can be performed internally by the photo MSLM include AND, OR, NAND, NOR, XOR, and NOT. The optical-to-electrical output device is a 2-D photodetector array. To obtain good noise rejection and low error rates, digital optical signals (binary intensity levels) are assumed for all input and output signals in the optical processor.

A. Matched-Filter Optical Inference Machine

The general matched-filter optical inference machine employs analog pattern recognition techniques and parallel optical logic to apply a set of given rules to a set of facts to infer a set of logical conclusions to the queries. This method is similar to the optical correlograph system described by Willshaw and Longuet-Higgins.[10]

Figure 4 shows a specific implementation of a query-driven matched-filter hybrid optical inference machine. This machine consists of an electronic controller, two E-SLMs, two O-SLMs, and a photodetector array which is operated in a thresholding mode. In this and subsequent figures, it should be noted that (1) the input light to the O-SLM is absorbed within the device and is not transmitted, and (2) the readout light is reflected out of the device by an internal mirror.

In the matched-filter system of Fig. 4, the facts and rules are grouped in block form (subsets) and stored electronically in the controller for rapid retrieval and transfer to the optical system. The two E-SLMs, O-SLM 1, the lenses L_1, L_2, and L_3, and the photodetector array are arranged to form a classical VanderLugt matched-filter system.[23] Thus lens L_1 is one focal length away from the planes P_1 and P_2, lens L_2 is one focal length away from planes P_3 and P_4, and lens L_3 is one focal length away from planes P_3, P_5, and P_6. The multiplication of the Fourier transforms of the signals

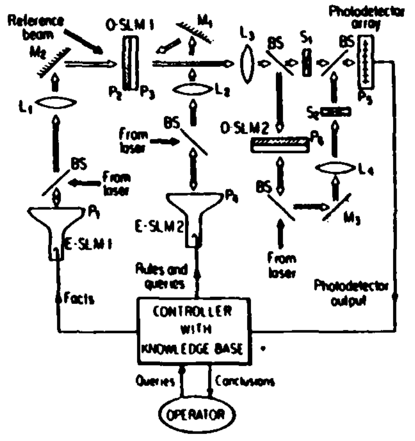


Fig. 4. Matched-filter optical inference machine.

to be matched is performed in O-SLM 1, and the matched-filter output is recorded on the photodetector array (shutter S_1 open, S_2 closed). The photodetector then transfers its output to the controller.

If the query dictates that several rules must be applied to the facts in succession, the resulting matched-filter outputs can be combined by using the optical logic capabilities of O-SLM 2. With S_1 closed and S_2 open, the logic output of O-SLM 2 can be imaged onto the photodetector array using lens L_4 and the photodetector output fed back to the controller. This ability permits rules to be applied as many times as necessary to various subsets of facts to generate the logical conclusions.

When operating the matched-filter optical inference machine, the operator queries the system through the electronic controller. In response, the controller writes the applicable subsets of facts onto E-SLM 1 and the applicable subset of rules onto E-SLM 2. This information is coded as a set of predetermined 2-D binary-level patterns. In the query example of Eq. (6), the **mother-of** rule and the complete set of facts in Eq. (3) would be the applicable sets.

The controller then activates O-SLM 1 which holographically records the Fourier transform of the facts as formed by lens L_1. The rules are similarly transformed by lens L_2, and this transform is used to read out O-SLM 1 via mirror M_1 as shown in Fig. 4. The output of O-SLM 1 is transformed by lens L_3 to form the matched-filter output on the photodetector array. This output consists of a set of focused spots of light which indicates the positions of the matches. These signals are then stored in O-SLM 2 and/or fed back to the controller, which then uses this input to select the possible conclusions from the set of facts.

Several options exist at this point, depending on the nature of the query being solved. For example, the controller could now load another part of the query into E-SLM 2, perform a second matched-filtering operation, and with S, closed and S_2 open perform a logical AND (with O-SLM 2) of the second correlation and the first which is already stored in O-SLM 2. The output of O-SLM 2 would then be read out onto the photodetector array. Thus the matched-filter inference machine is capable of sequentially performing all combinations of 2-D optical pattern correlations and binary level logic operations on patterns representing the data objects, rules, facts, and queries.

To solve the ? **mother-of** Jean query in Eq. (6), this system would first examine the query for the specified data objects (Jean in this case) and would then treat the **mother-of** rule as if its variables were replaced by the appropriate data objects. In this case, the effective **mother-of** rule would become

$$\begin{aligned} &?\ \textbf{mother-of Jean IF}\quad Z\ \textbf{married to ? AND}\\ &\qquad\qquad\qquad\qquad Z\ \textbf{father-of Jean.} \end{aligned} \tag{8}$$

Comparing Eq. (8) with the original **mother-of** rule as defined in Eq. (4), the variables X and Y have been replaced with the desired unknown symbol ? and the data object Jean, respectively. Since the **mother-of** rule has two conditions, the controller has to invoke two matched-filtering operations.

The order in which the conditions are satisfied does not matter since all of them must be true for the query to be satisfied. Since the second condition has the data object Jean as a constraint, the first matched-filtering operation matches the **father-of** facts (placed on E-SLM 1) with the data object Jean (placed on E-SLM 2). The output of the matched-filter is then a representation of all facts associated with the condition **father-of** Jean. In this case, there is only one fact associated with this condition, Bob **father-of** Jean. The controller then retrieves the father's name Bob and matches the condition Bob **married-to** with the set of facts. The second matched-filter output points to the fact Bob **married-to** Beth. Finally, the controller simply associates the conclusion Beth with ? and returns the conclusion to the operator.

In the case where there are several matches, it is possible for the controller to match all the resulting conclusions with the next condition for full parallelism. Furthermore, if no match is made (i.e., no spots of light above threshold on the photodetector array), the condition cannot be satisfied, making the query false.

The block electronic storage scheme suggested here is not the most efficient means of storing the rules and the facts because a single data object may be associated with several different facts. However, because electronic storage is relatively inexpensive, block-form storage does not appear to be inappropriate for the initial investigations of these machines.

Since data objects are not expected to change often, partitioning the knowledge base into blocks will generally not have to be done frequently. The advantage of block electronic storage is that it not only reduces the data acquisition and retrieval time but also eliminates the need to transfer the entire knowledge base to the spatial light modulators which currently have only modest space–bandwidth products.

B. Mapped-Template Optical Inference Machine

In the mapped-template optical inference machine, mapping templates are used to store the relationships between the data objects and are thus defined by the facts. Conclusions are inferred to queries by applying these mapping templates to the data objects in the order prescribed by the rules. This usage of mapping templates is similar to the associative nets described by Willshaw and Longuet-Higgins.[10]

Using the example defined by Eqs. (1)–(6), the relations **is-male, is-female, married-to,** and **father-of** from the facts in Eq. (3) would map an input set from D the set of data objects defined in Eq. (1), to an output set, also from **D**. Let $\mathbf{D}_i$ and $\mathbf{D}_o$ represent the input and output sets of data objects. Furthermore, let the data objects in the mth position of $\mathbf{D}_i$ and $\mathbf{D}_o$ be denoted by d_{im} and d_{om}. Using the data set **D** for $\mathbf{D}_i$ and $\mathbf{D}_o$ as defined in Eq. (1), the mapping templates corresponding to the **is-female** and **father-of** facts as defined in Eq. (3) are shown in Fig. 5 with the elements

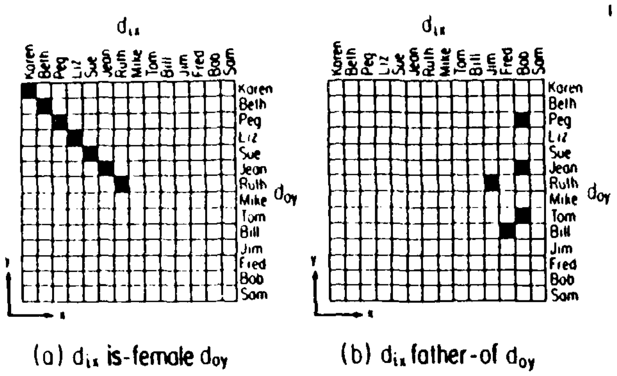

d_{ix}
Karen Beth Peg Liz Sue Jean Ruth Mike Tom Bill Jim Fred Bob Sam
d_{oy}
(a) d_{ix} is-female d_{oy}
(b) d_{ix} father-of d_{oy}

Fig. 5. Mapping templates for (a) the is-female facts and (b) the father-of facts for the entire set of data objects in Eq. (1).

of the input set $D_i(d_{ix})$ along the columns (x axis) and the elements of the output set $D_o(d_{oy})$ along the rows (y axis) of the templates.

The mapping templates are binary masks consisting of transparent squares (logical 1 and shown as black squares in Fig. 5) on an opaque background (logical 0 and shown as white in Fig. 5). The interpretation of these templates is as follows: A transparent square in the (x, y) position of, say, **father-of** indicates the fact

$$d_{ix}\ \textbf{father-of}\ d_{oy}. \tag{9}$$

Given these two templates, the mapping templates for **is-male** and **married-to** are straightforward to generate.

Note that the mapping between D_i and D_o is not necessarily one-to-one. However, a mapping template is reciprocal in that if the right-hand side of Eq. (9) is specified instead of the left, the relationships for the left-hand side may be inferred from the template.

Alternatively, to limit the size of the mapping template and conserve space, D could be subdivided into subsets whose data objects are related in some way. Considering the facts in Eq. (3), it is reasonable to split D into a set of males and a set of females denoted by

$$\begin{aligned} D_m &= \{\text{Mike, Tom, Bill, Jim, Fred, Bob, Sam}\} \\ D_f &= \{\text{Karen, Beth, Peg, Liz, Sue, Jean, Ruth}\}, \end{aligned} \tag{10}$$

where D_m and D_f represent the male and female sets, respectively. With these subsets, the relationships **is-male** and **is-female** would no longer be needed.

With the data set partitioned, the mapping templates for the factual relationships between the elements of D_m and D_f would simply be the corresponding regions in the original full-size mapping templates in Fig. 5. For the relation **is-female** and the data set D_m, the template would always be opaque.

To perform logical inferring, the mapping-template concept is implemented as illustrated in Fig. 6. Given an input vector D_i, the associated output vector D_o for a particular mapping template is found by first vertically expanding D_i along the y axis so it forms an array, each row of which equals D_i, as shown in Fig. 6. This expanded form of D_i is then optically overlaid with the mapping template using imaging optics and a 2-D logical AND operation is performed. The resulting output, when viewed along the rows, corresponds to the output vector D_o.

To perform the reciprocal operation of the mapping template, the input vector would be expanded horizontally and logically ANDed with the mapping template. The output vector would then be taken looking down the columns.

Depending on the mapping template, it is possible for multiple inputs in D_i to produce the same output element in D_o. For this reason, a 2-D output photodetector array is used for establishing the exact input-to-output correspondence, should this be needed in solving the query.

A hybrid optical inference machine which implements mapped-template logic is shown in Fig. 7. It consists of an electronic controller, two E-SLMs, two O-SLMs, and a 2-D photodetector array. Like the matched-filter optical inference machine, the controller in this system electronically stores the knowledge base and controls the SLMs and the shutter. The modulator O-SLM 1 is operated in the logic mode and usually performs the AND operation, while O-SLM 2 is used as a 2-D memory unit to allow further processing of the outputs, and is optional.

When the controller is given a query by the operator, a vertical line is written on E-SLM 1 at the location of

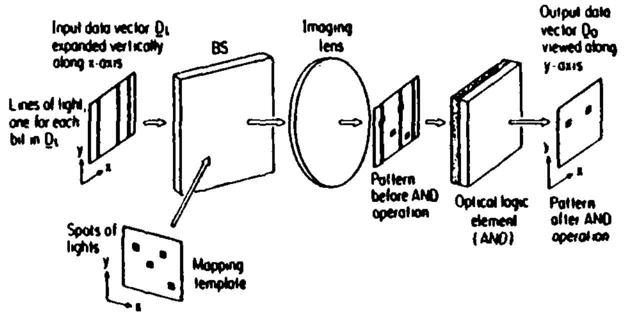

Input data vector D_i expanded vertically along x-axis
Lines of light, one for each bit in D_i
BS
Imaging lens
Output data vector D_o viewed along y-axis
Pattern before AND operation
Optical logic element (AND)
Pattern after AND operation
Spots of lights
Mapping template

Fig. 6. Conceptual implementation of mapped-template logic.

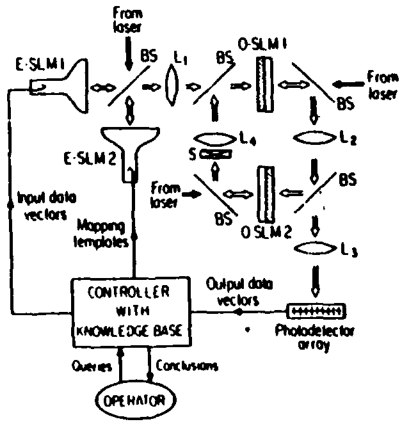


Fig. 7. Mapped-template optical inference machine.

the known data objects in D_i. Then the controller writes the mapping template corresponding to the rule (or first condition) associated with the query onto E-SLM 2. The outputs of both E-SLMs are imaged onto O-SLM 1 with lens L_1. The logical AND of the two inputs is formed in O-SLM 1 and imaged onto the photodetector array by lenses L_2 and L_3. If desired, the output could also be imaged onto O-SLM 2 by lens L_2 and latched. The stored output in O-SLM 2 could then be imaged via lens L_4 back into O-SLM 1 by opening shutter S should further processing be necessary.

The output of the photodetector array is fed back to the controller where the inferred data objects in D_o which satisfy the current mapping rule are determined. Further mapping templates are then applied by the controller as determined by the query and rules.

Operation of this optical inference machine can be demonstrated for the ? **mother-of** Jean query in Eq. (6). As with the matched-filter machine, the mapped-template system considers the effective form of the **mother-of** rule given the data object Jean as specified in Eq. (8). The controller first uses the mapping rule template for **father-of** as shown in Fig. 5 and the input vector corresponding to Jean, which is, from Eq. (1), [0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]. Since Jean is specified on the output side of **father-of**, the input vector is expanded horizontally rather than vertically on E-SLM 1. Scanning the rows of the output array produces the output vector [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0] which corresponds to Bob.

The controller then feeds this output vector back to E-SLM 1 as the input vector for the **married-to** mapping template. Since this input is on the right side of the **married-to** rule, the vector is expanded vertically on E-SLM 1. The inference operation is repeated with the **married-to** mapping template on E-SLM 2, producing the output vector (view along the columns) [0 1 0 0 0 0 0 0 0 0 0 0 0 0 0], which indicates the conclusion Beth.

Since multiple outputs for the same data object could be generated, viewing the rows or columns of the output array could lead to an integral multiple of a single light beam intensity. In this case, the photodetector output is electronically clipped to the single light beam level if the photodetector output is to be fed back to E-SLM 1 as input via the controller.

If the optional optical feedback loop is not used, there is a possible modification to the system in Fig. 6 which will simplify the device requirements. Instead of performing the logical AND operation in O-SLM 1, the output of E-SLM 1 (the expanded input data vector) could be used to read out the mapping template in E-SLM 2, thus eliminating the need for a 2-D optical logic device. However, the advantage of having O-SLM 1 is that (1) it can conveniently perform the logical NOT operation on a condition, and (2) the processed patterns are automatically latched into O-SLM 1. This allows the controller to begin setting up the next mapping template while it simultaneously reads the photodetector array, thus providing some degree of concurrent operation.

Further possibilities for increasing processing speed are to place multiple mapping templates which are spatially separated from each other on E-SLM 2. The input data vectors on E-SLM 1 would have to be repositioned accordingly. However, multiple inferences could then be made in parallel.

IV. Concluding Remarks

Basic architectures for a hybrid optical machine capable of solving symbolic logic problems have been discussed in general terms. This inference machine was considered from both a rule-driven and query-driven approach. Two hybrid optical designs of a query-driven inference machine were described which used matched-filter logic and mapped-template logic.

In comparing the two designs, the mapped-template system should be less demanding on the spatial resolution characteristics of the spatial light modulators and should be easier to implement than the matched-filter machine. Furthermore, the mapped-template system should have better noise performance since there is no analog processing in this system. That is, all optical signals remain encoded as binary intensity levels in the mapped-template system, whereas the matched-filter system must contend with the noise from the analog matched-filtering process, even though binary intensity input and output patterns are used.

Although two hybrid architectures have been presented, other equally effective system designs are possible. Given the growing interest in integrating symbolic logic processing into the computer of the future, the idea of downloading the inference operations of scanning, searching, and matching to a parallel optical processor merits continued investigation.

This work was supported in part by the Air Force Office of Scientific Research under grant AFOSR-84-0358.

References

1. D. S. Nau, "Expert Computer Systems," IEEE Comput. 63 (Feb. 1983).
2. T. Moto-oka and H. S. Stone, "Fifth-Generation Computer Systems: A Japanese Project," IEEE Comput. 6 (Mar. 1984).
3. K. L. Clark and F. G. McCabe, *Micro-Prolog: Programming in Logic* (Prentice-Hall, Englewood Cliffs, NJ, 1984).
4. W. F. Clocksin and C. S. Mellish, *Programming in Prolog* (Springer-Verlag, 1981) New York.
5. D. Gabor, "Associative Holographic Memories," IBM J. Res. Dev. 156 (1969).
6. H. Akahori and K. Sakurai, "Information Search Using Holography," Appl. Opt. 11, 413 (1972).
7. M. Nakajima, T. Morikawa, and K. Sakurai, "Automatic Character Reading Using a Holographic Data Processing Technique," Appl. Opt. 11, 362 (1972).
8. A. W. Lohmann and H. W. Werlich, "Holographic Production of Spatial Filters for Code Translation and Image Restoration," Phys. Lett. A 25, 570 (1967).
9. D. J. Willshaw, O. P. Buneman, and H. C. Longuet-Higgins, "Non-Holographic Associative Memory," Nature London 222, 960 (1969).

10. D. J. Willshaw and H. C. Longuet-Higgins, "Associative Memory Models," Machine Intell. 5, 351 (1970).
11. R. A. Athale, W. C. Collins, and P. D. Stilwell, "High Accuracy Matrix Multiplication With Outer Product Optical Processor," Appl. Opt. 22, 368 (1983).
12. R. P. Bocker, "Optical Digital RUBIC (Rapid Unbiased Bipolar Incoherent Calculator) Cube Processor," Opt. Eng. 23, 26 (1984).
13. Y. Z. Liang and H. K. Liu, "Optical Matrix–Matrix Multiplication Method Demonstrated by the Use of a Multifocus Hololens," Opt. Lett. 9, 322 (1984).
14. H. J. Caulfield, W. T. Rhodes, M. J. Foster, and S. Horvitz, "Optical Implementation of Systolic Array Processing," Opt. Commun. 40, 86 (1981).
15. M. Carlotto and D. Casasent, "Microprocessor-Based Fiber-Optic Iterative Optical Processor," Appl. Opt. 21, 147 (1982).
16. A. Huang, "Parallel Algorithms for Optical Digital Computers," Proc. Soc. Photo-Opt. Instrum. Eng. 422, 13 (1983).
17. K. Brenner and A. Huang, "An Optical Processor Based on Symbolic Substitution," in *Technical Topical Meeting on Optical Digest, Computing* (Optical Society of America, Washington, D.C., 1985), paper WA4.
18. A. Huang, "Why Use the Parallelism of Optics?," in *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, D.C., 1985), paper WA2.
19. A. D. Fisher, C. L. Giles, and J. N. Lee, "An Adaptive, Associative Optical Computing Element," in *Technical Digest, Topical Meeting on Optical Computing* (Optical Society of America, Washington, D.C., 1985), paper WB4.
20. A. Schwartz, X. Y. Wang, and C. Warde, "Electron-Beam-Addressed Microchannel Spatial Light Modulator," Opt. Eng. 24, 119 (1985).
21. 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).
22. C. Warde and J. I. Thackara, "Oblique-Cut $LiNbO_3$ Microchannel Spatial Light Modulator," Opt. Lett. 7, 344 (1982).
23. A. B. VanderLugt, "Signal Detection by Complex Spatial Filtring," IEEE Trans. Inf. Theory IT-10, 2 (1964).

END
DATE
10-88
DTIC