CRC

COMPUTER ANALYSIS *and* PERCEPTION

Volume I

Visual Signals

Ching Y. Suen

Renato De Mori

CRC PRESS

Computer Analysis and Perception

Volume I
Visual Signals

Editors

Ching Y. Suen, Ph.D.
Professor and Chairman
Department of Computer Science
Concordia University
Montreal, Canada

Renato De Mori, D. Eng.
Professor of Computer Science
University of Torino
Torino, Italy

CRC

CRC Press, Inc.
Boca Raton, Fla.

Library of Congress Cataloging in Publication Data
Main entry under title:

Computer analysis and perception.

Bibliography: p.
Includes index.
Contents: v.1. Visual Signals— v. 2. Auditory Signals
1. Optical pattern recognition. 2. Speech processing systems. I. Suen, Ching Y. II. De Mori, R. (Renato) III. Title: Computer analysis and perception of auditory signals.
TA1650.C65 621.3819'598 81-15494

ISBN 0-8493-6305-5 (v. 1) AACR2
ISBN 0-8493-6306-3 (v. 2)

International Standard Book Number 0-8493-6305-5 (vol. I)
International Standard Book Number 0-8493-6306-3 (vol. II)

Library of Congress Card Number 81-15494
Printed in the United States

PREFACE

Recent progress in technology has significantly broadened the capability and applications of computers. At present, computers not only can see, but also speak and listen to human beings. Research and development in making computers *intelligent* have indeed reached a very advanced state. Consequently, much published literature has been devoted to system designs and recognition logics. However, there is a lack of comprehensive reference material spanning both spectra of auditory and visual signals, particularly with respect to their perception and analysis. In order to bridge this gap, we have asked the authors of this book to gather and present the lastest developments in this field. This volume is divided into two parts. The first part consists of six chapters concerned with computer recognition of visual signals of the environment and symbols drawn by human beings. The second part consists of five chapters related to auditory signals which include studies on speech perception, modeling, and recognition systems.

In Volume I, Chapter 1 introduces the problem of building computers that perceive real-world scenes and objects.

Chapters 2 and 3 discuss the process of recognizing symbols and handprinted characters by computer. The architecture of recognition units, methods used to achieve high recognition rates and the importance of standardization for handprint models are discussed.

Chapter 4 introduces real time techniques in computer analysis of cursive writing and lists primitives used to identify cursive scripts. Typical systems are also presented.

Chapter 5 reviews the use of signature dynamics as a means of personal identification. It illustrates an automatic signature verification system with field-test results.

Chapter 6 presents the principle and techniques of computer processing of hand-drawn sketches and figures.

In Volume II, Chapter 1 discusses auditory perception and its application to computer analysis of speech signals.

Chapter 2 presents numerical methods for making models of speech production and the possible application of these models for recognition.

Chapter 3 describes a model for segmenting speech signals into syllabic segments and for performing a precategorical classification of speech segments before generating phonemic hypotheses.

Chapter 4 deals with problems and methodologies for recognizing connected speech.

Finally, Chapter 5 discusses the impact of integrated circuit technology on speech processing methodologies.

We hope the techniques described in this book will enable scientists to design and implement algorithms and systems to recognize visual patterns and symbols. The ideas contained herein should stimulate the development of innovative approaches and models for computer analysis and perception of visual objects and patterns. Similarly, designers of speech recognition and understanding systems should benefit from the results reported here. This book can also be used as a reference for graduate courses in man-computer communications, pattern recognition, image processing, and artificial intelligence.

We would like to thank W. Gillespie, I. Greenshields, T. Radhakrishnan, and R. Shinghal for helpful suggestions and constructive comments on various chapters of this book.

R. De Mori would like to dedicate his contribution in preparing this book to the memory of Professor Rinaldo Sartori.

Ching Y. Suen
Renato De Mori
March 1981

THE EDITORS

Ching Y. Suen received an M. SC. (ENG.) degree from the University of Hong Kong and a Ph.D. degree from the University of British Columbia, both in Electrical Engineering. In 1972, he joined Concordia University where currently he is Professor and Chairman of the Department of Computer Science.

He conducted several research projects while he was a guest of the Research Laboratory of Electronics of MIT (U.S.A.) and the Institut de Recherche d'Informatique et d'Automatique (France) on several occasions between 1975 and 1979. In Spring 1979, he was an Invited Professor of the Département d'Electricité of Ecole Polytechnique Fédéral de Lausanne (Switzerland).

He participated in the development of several Canadian Standards on Optical Character Recognition and currently is Chairman of the Character and Mark Recognition Committee of the Canadian Standards Association.

He is the author of numerous papers dealing with transistor circuits and electronics, automatic analysis and synthesis of speech perception and psychophysics, electronic aids for the visually handicapped, handwriting education, computer vision, and character recognition. He is the author of a book entitled *Computational Analysis of Mandarin* (Birkhäuser Verlag, Basel-Stuttgart-Boston, 1979). His current interests include character analysis and recognition, language properties and text processing, speech analysis and synthesis, and mini- and micro-computer applications.

An active member of several professional societies, Professor Suen is an overseas Associate Editor of the journal *Signal Processing*.

Professor Renato De Mori received a doctorate in Electronic Engineering from Politecnico di Torino, Italy, in 1967. He was Assistant and Associate Professor at Politecnico di Torino from 1969 to 1976. In 1976 he became Chairman and Head of the Institute for Computer Science at the University of Torino. He acted as Chairman of sessions devoted to pattern recognition and artificial intelligence at World IFIP, and pattern recognition conferences, and is Associate Editor of the journals entitled *Signal Processing* and *Speech Communications,* both published by North Holland.

Professor De Mori is the author of about 70 papers, mostly published in international journals, and in the proceedings of international conferences devoted to computer systems, pattern recognition, and artificial intelligence.

He is author of the book *Computer Models of Speech Using Fuzzy Algorithms,* Plenum Press, 1982, and is co-author of *Syntactic Pattern Recognitions, Applications,* Springer-Verlag. He is a contributor to the book *Applications of Pattern Recognition,* CRC Press, Inc., 1982, and the book *Pattern Recognition and Signal Processing,* Sijthoff & Noordhoff International Publishers, 1978. His interests are speech signal processing, syntactic pattern recognition, knowledge representation, and organization of complex information systems.

He is also a member of the Scientific Council of the Centre National D'Etude Et de Telecommunications (CNET), Lannion, France.

Professor De Mori has been invited to give seminars in many European and American Universities and Research Centers, as well as at the University of Novosbirisk (U.S.S.R.) and in Japan.

CONTRIBUTORS

Marc Berthod, D. d'Etat
Institut National de Recherche
en Informatique et en Automatique
Valbonne, France

George L. Brantingham
European Speech Laboratory
Semiconductor Group
Texas Instruments,
Loubet (Nice), France

Piero Demichelis, D. Eng.
CENS—Istituto di Electtrotecnica
Generale
Politechnic of Torino
Torino, Italy

Renato De Mori, D. Eng.
Professor of Computer Science
Istituto di Scienze
dell'Informazione
University of Torino
Torino, Italy

Giovanna Giordano, D.Sc.
Istituto di Scienze dell'Informazione
University of Torino
Torino, Italy

Noel M. Herbst, Ph.D.
Research Staff Member
IBM T. J. Watson Research Center
Yorktown Heights, New York

Pietro Laface, D. Eng.
Cens—Istituto di Electtrotecnica
Generale
Politechnic of Torino
Torino, Italy

Wen C. Lin, Ph.D.
Department of Electrical and Computer
Engineering
University of California at Davis
Davis, California

C. N. Liu, Ph.D.
Research Staff Member
IBM T. J. Watson Research Center
Yorktown Heights, New York

Paul Mermelstein, D.Sc.
Manager, Speech Communications Research
Bell Northern Research
Visiting Professor, INRS—
Telecommunications
University of Quebec
Auxiliary Professor, McGill University
Montreal, Quebec, Canada

Shunji Mori, Ph.D.
Head of Pattern Processing Section
Electrotechnical Laboratory
Ibaraki, Japan

Günter Ruske, Dr.-Ing.
Department of Electrical
Engineering
Division of Data Processing
Technical University of Munich
Munich, Germany

Katsuhiko Shirai, Ph.D.
Professor, Department of Electrical Engineering
Waseda University
Tokyo, Japan

Ching Y. Suen, Ph.D.
Professor and Chairman
Department of Computer Science
Concordia University
Montreal, Canada

Leonard Uhr, Ph.D.
Professor
Department of Computer Sciences
University of Wisconsin
Madison, Wisconsin

COMPUTER ANALYSIS AND PERCEPTION

Editors

Ching Y. Suen, Ph.D. and Renato De Mori, D. Eng.

Volume I
VISUAL SIGNALS
Computer Perception and Scene Analysis
Automatic Recognition of Symbols and Architecture of the Recognition Unit
Standardization and Automatic Recognition of Handprinted Characters
On-Line Analysis of Cursive Writing
Automatic Signature Verification
Computer Processing of Hand-Drawn Sketches and Diagrams

Volume II
AUDITORY SIGNALS
Auditory Perception and its Application to Computer Analysis of Speech
On the Use of Syllables in Speech Understanding Models
Computer Recognition of Continuous Speech
Computer Models for Speech Production
Impact of Integrated Circuit Technology on Speech Processing Methodologies

TABLE OF CONTENTS

Volume I

Chapter 1

COMPUTER PERCEPTION AND SCENE ANALYSIS

Professor Leonard Uhr

TABLE OF CONTENTS

I. PATTERN RECOGNITION AND SCENE DESCRIPTION

This chapter attempts to introduce the problem of achieving computers that perceive real-world scenes of objects. It begins by exhibiting how enormous, and how difficult, this problem is, and by pointing out how it must be, and has been, cut down to size, and simplified. It then surveys key systems that have attempted to approach recognition and description in a relatively general manner.

The problem of perception of moving objects is emphasized, because it is central both to an understanding of human perception in the dynamic, constantly changing real world, and also to the development of robot systems that can interact effectively with real-world objects.

This chapter focuses on programs that have a parallel-serial layered converging structure, ones that successively transform and abstract raw sensed images as they search to recognize objects and build up a description of the scene. This kind of structure is suggested by the layered neuronal structure of the visual system of living organisms, and it is one that will, potentially, allow for the enormous speed-ups in processing time that will be needed before real-time recognition of moving objects is finally achieved.

The explosion in VLSI (Very Large Scale Integration) technologies that today make possible the manufacture of a single small ''chip'' on which is printed 100,000 transistor-equivalent devices means it is now possible to build very large parallel-serial arrays and networks containing thousands of computers all working together on the same problem. The first such systems are today just beginning to be completed. These will be the computers that will, for the first time, begin to execute the very large perceptual programs in true parallel-serial fashion (rather than simulating them in very slow serial iterations using traditional single-CPU serial machines). These arrays and networks are introduced, and the problems of mapping perception programs onto their parallel-serial structures are explored.

II. THE ENORMOUS COMPLEXITY OF PERCEPTION

A great multitude of different kinds of patterns exist in the real world. Each and every object has an enormous variety of ''signatures'', or ways it might appear—all the different images it might make on some sensing medium. Nature has evolved eyes, ears, antennae, and a variety of other sensing/perceiving organs, because it is vital that any living organism recognize and respond appropriately to the different aspects of its environment that are necessary, desirable, discomfitting, or lethal.

The spectrum of visible light (that is, energy that the visual systems of human beings and most other higher animals can sense through their eyes) is for human beings by far the richest source of information about the external environment. This chapter will concentrate on the kinds of visually sensed objects that impinge upon a two-dimensional sensor like the human eye's retinal array of rods and cones, and are perceived by an information-processing system like the human visual system.

Can we get computers to recognize and describe the objects and scenes that we human beings perceive? There are a great number of very important practical applications, including airplane and satellite pictures of cities, fields, etc.; microscope slides of neurons, blood cells, chromosomes, DNA, etc.; X-rays of bones, organs, etc.; city streets, houses, manmade objects; country landscapes, trees, plants and leaves; printed letters and words; human faces.

Each of these poses an extremely difficult problem, for it entails the recognition of enormously complex patterns of objects, where each object may present a potentially infinite number of different images. The human visual system uses over 100 million sensing elements (the rods and cones) in each eye, many millions of neurons transmitting and transforming the raw sensed image through the retina and lateral geniculate to the primary visual cortex in the brain, and many billions of neurons in the brain itself.[30,43,79] Necessarily, rough estimates suggest that half of the brain's 12 billion or so neurons are probably involved with perception, primarily with visual perception.

Even the largest of today's computers is still quite small in comparison.

We human beings are all experts at recognizing and perceiving the countless different objects in our familiar environment. We quickly learn to recognize and describe new objects, as when we are trained as X-ray technicians or airplane spotters. Because perception is right under our noses we find it hard to realize how great an achievement it really is.

Programs that recognize patterns and describe scenes, or even perform small parts of the perceptual task (e.g., extract only certain objects being looked for, find edges and contours, convert the scene into regions) may seem simpler and less ''intelligent'' than programs that play chess or GO, prove theorems, or write poetry. But there are good reasons to believe that perception is one of the most complex and most powerful of the intellect's tools. The first pattern recognition programs were written about 25 years ago, and hundreds or thousands of programs have been coded since then. Yet we are still a long way from having programs capable of recognizing the wide variety of objects that any ''ordinary'' human being can recognize.

It is necessary to drastically simplify the full-blown scene that a human perceiver can handle before today's computer programs can succeed. Today's simplifications are enormous. Virtually all of today's programs work on problem sets of pictures with all of the following simplifications made:

1. Instead of handling a continuing stream of inputs, virtually all programs work with only a single static image (roughly equivalent to a single still photo, in contrast to a movie).
2. Instead of resolving an image on each of the two human eyes' roughly 10 million cones (sensitive to color and shape) and 100 million rods (sensitive to motion, intensity gradients, and change), today's programs use a single ''retina'' with at most 250,000 primitive units (for a 500 by 500 television image), and more typically 40,000, 4000, or even 400 primitive units.
3. The very fine sensitivity of the eye for different intensities and for different colors is similarly coarsened: often only 16, 8, or even 2 levels of grey-scale are used; usually the problems of color are ignored.
4. In almost all cases the actual images used are greatly simplified. Clean pictures are chosen, rather than cluttered pictures. ''Good'', ''clear'' examples of objects are used (that is, ones that people, and computer programs, can easily recognize; which means that such choices frequently beg the question).
5. Only a very restricted set of objects is used. Here researchers fracture into many different groups, each working on the particular application problem (or, sometimes, artificial toy-like problem) of interest. For example, a specific system may be developed for landsat photos of North America, the Upper Great Lakes Region, cells, or white blood cells. Such programs will have built into them specific information about the particular objects encountered; they will not work on any other kinds of objects and their potential ability to do so almost always remains quite unknown.
6. Only a manageable number of objects from this already very restricted set is actually used. Rarely is a program written to handle more than 30 or 50 different objects.

Indeed, most programs are designed to recognize only 10 or 20 objects of interest; many concentrate on only 2 or 3, or even 1.

Such simplifications are absolutely necessary, to cut this very large and complex problem down to manageable size. There are many problems of great practical importance for which well-tailored specific programs can be designed.

But it is important to remember that a human being's visual perception is quite general, and can handle any and all of these special-purpose problems. Living perceptual systems serve as existence proofs that a general system can be built. And it can even be built in the small-step by small-step fashion used by natural evolution and learning.

III. PERCEPTION PROGRAMS AND PARALLEL COMPUTERS

A. Systems for General Pattern Recognition and Scene Description

The following is a brief, and necessarily sketchy, overview of research that attempts to attack the very difficult general problem of recognizing and describing objects and scenes of objects, using detailed television pictures of real-world scenes. I tend to concentrate on systems that try to attack the whole problem—which today means that they have been tested on (and at least to some extent, therefore, designed for) a very limited number of example scenes.

I also focus on parallel-serial layered converging systems, since they seem to me the most promising, especially looking to the future when they will be executed directly by parallel-serial array/network computers. These seem especially appropriate for the key problem of recognition and description of scenes of moving objects. And the new very large arrays of many thousands of simple computers and, possibly soon, networks of hundreds, or even thousands, of more powerful computers, offer real promise of achieving the great processing power needed to actually handle moving pictures in real time (that is, on the order of 20 to 100 ms per picture).

An enormous amount of research has been done on pattern recognition and scene analysis.[74,76,78]

Most recent research on perception systems has concentrated on one specific aspect of the total process, such as edge detection,[10,24,41,57] region growing,[90] or segmentation.[70] This reflects the enormous difficulty of the problem of perception, when it is no longer simplified to handle only one single input object, as in much of the early pattern recognition research, or simple idealized objects, such as the polyhedra used in robot research.

Relatively few systems have been implemented that actually describe the scene, and even fewer have reached the point where they can be tested. One of the most interesting is Kanade's,[59] which has been tested on only one input picture, but appears to use a tree of sub-subparts of a complex object in a clean and general way. Tenenbaum and Barrow[71] give partial descriptions of one indoor scene, using semantic relaxation labels to disambiguate regions. But it is not clear how ad hoc are the particular semantic constraints used (e.g., "picture" can be on a "wall" but never on a "door"—but in the real world, pictures sometimes are found on doors). Bacjsy,[3,4] Sloan,[63] and Shirai[61] have also developed very interesting systems.

Rubin and Reddy[58] reported results on one scene for a system that uses many large quasi-parallel knowledge sources, in the spirit of the Carnegie-Mellon systems for speech,[36,54] where several small PDP-11 computers work in parallel, interacting through a common working memory. But it is not clear to what extent Rubin and Reddy's program was designed to handle the particular scene processed.

Rosenfeld and Davis[56] have sketched out a multi-level relaxation labeling system, that seems rather close in spirit to the layered variable-resolution cones described below. But most uses of relaxation[91] have been on a single level, and with very simple inputs (e.g., a square, or a meandering river).

B. Layered Parallel-Serial Systems for Perception

A number of researchers have recently begun to explore layered parallel-serial converging "cone" or "pyramid" systems (e.g., Tanimoto,[67,68] Klinger and Dyer,[26] Kruse,[28,29] and see below). Tanimoto and Klinger[69] edited a book that examines such systems, and Hanson and Riseman's work[22] contains a number of papers devoted to them. Some researchers (e.g., Hanson and Riseman,[19–21]) use the cone only for the first layers of processing, and are building larger systems with a number of additional powerful processors (e.g., to grow regions, segment, compute depth). Others[31,32] use chiefly simpler averaging operations, and handle different operations separately, in separate pyramids.

In contrast, "recognition cones"[13,14,69,73,77–79,81] attempt to incorporate as much of the higher-level processing as possible into the cone itself.

C. Recognition and Description of Moving Objects

To perceive a moving object, a system must be able to handle still another dimension of data, since a continuing sequence of inputs must be processed in order to detect motion, and to more precisely determine its direction and speed. As a result, relatively little work had been done on this problem until quite recently, and it still presents enormous burdens to today's computers, if they are to work with real-world scenes. A single 500 by 500 pixel TV frame already contains enormous amounts of data. But motion means at least 3, or more reasonably 10 or 100, TV frames must be processed.

Among the earliest programs for motion were those of Uhr[75] (for simple one-dimensional strings) and of Potter[44–47] working with Uhr. (This ignores research on special-purpose problems like motion in cloud cover.[33] But that kind of work has not attempted to recognize objects; rather, it uses relatively simple techniques, e.g., autocorrelation of local pixels between two images of the same scene, to estimate motion.) Aggarwal and Duda,[1] Chow and Aggarwal,[9] Roach and Aggarwal,[55] Yalamanchili et al.,[89] Nagel,[38–40] Radig,[51] Radig et al.,[52] and Jain[25] have been among the most active researchers attacking this problem in recent years. In general, these systems use one (in the case of Potter and the earlier work of Nagel) or several simple templates or feature-detectors, try to find a standard position (e.g., from an unmoving part of the field), and then assess how some subpart of the field has changed with respect to the basic (set of) template(s). They therefore do relatively little in the way of recognition of objects in the scene—but with the attendant advantage that they are able to perceive "low-level" motion with a reasonably simple and fast set of processes.

Several additional pieces of research have been done using simpler, or abstracted, scenes. Chien and Jones[8] developed a very interesting robot system capable of handling certain types of moving objects. Uno et al.[84] and Holland et al.[23] report on systems that use television cameras placed in known positions vis-à-vis a conveyor belt carrying known objects at known orientations and moving at a known speed. Motion in simulated environments has been examined, at least to some extent, by Uhr and Kochen,[82] Korn,[27] Salveter,[60] Doran,[12] Badler,[2] O'Rourke and Badler,[42] and others. The related problem of detecting changes between two scenes (as due to, e.g., binocular disparity) has been attacked by Quam,[50] using correlational techniques, Price,[49] using symbolic representations, and others.

IV. A QUICK OVERVIEW OF PARALLEL COMPUTERS

A. Very Large Arrays of Thousands of Processors

Parallel array computers are just now being developed that are large enough to begin to handle real-world scenes.

Thus Duff[15,16] completed CLIP4® in 1980, a 96 by 96 array of programmable processors. Kruse[28,29] has built PICAP, which can zoom and pan around a large picture, to process a 64 by 64 array. Sternberg[64] has built a similar system, the Cytocomputer (see also Lougheed and McCubbrey,[35]) which has a very long pipeline of over 100 processors to speed up processing.

PICAP and the Cytocomputer are actually serial at the hardware level, but they are built to scan the image as fast as possible. PICAP takes about 1 μsec per pixel, or roughly 4 ms for a 64 by 64 array. The Cytocomputer takes about 1 μsec also; but, when the pipeline is full, which means that roughly 100 instructions are being executed at the same time, each on whichever pixel has reached it, overall processing is effected at a rate of about 10 nsec per instruction. The Cytocomputer processes very large 1024 by 1024 arrays.

Reddaway and his group (Flanders et al.,[17] Reddaway[53]) designed and built DAP, a 64 by 64 array of very fast processors. In these systems, all processors execute the same instruction, each on a different cell, or pixel, in the scene.

CLIP®, PICAP, or the Cytocomputer (which were designed for image processing and pattern recognition, and therefore have 3 by 3 window operations built into their hardware and parallel-logic circuits), can execute a single near-neighbor operation, like thinning, or local edge-detection, in a single machine instruction. DAP can execute the same process with a short sequence of instructions.

Such a 3 by 3 local operation, executed everywhere on the array, takes about 2 μsec for DAP, 11 μsec for CLIP4®, and 4 ms for PICAP. These are all general-purpose computers, which means that any complex function desired can be built from sequences of these basic window instructions (plus the other types of instructions in these computers' repertoires).

Goodyear® Aerospace[5] is now beginning to build a very large and very fast 128 by 128 array (the Massively Parallel Processor, or MPP®), with an expected completion date of Spring 1982 or 1983. It will use VLSI (Very Large Scale Integration) chips that have eight processors on each chip, and a very fast technology that should make it several times faster and more powerful still.

These arrays are interfaced to a general-purpose serial computer, along with additional memory and input-output devices (including TV cameras), so that they can be used, along with the standard serial computer's instructions, as another (albeit unusual, and unusually powerful) resource.

B. Networks of Independent Processors

A number of structures are being explored for networks of many computers. Here a variety of different architectures are being examined, e.g., 8 to 64 processors communicating through one or a few crosspoint switches[86] or an n-cube, lattice or star of processors, each connected to a small number of neighbors.[65,66,87,88] Thurber[72] and Uhr[80] describe and examine a wide variety of such networks in some detail. Among the most promising architectures are X-trees[11] and Hyper-trees[18] and also pyramids that combine arrays and networks.[83] Essentially, these are trees augmented with links connecting their buds and internal nodes to draw them closer together.

Systems of this sort are today being built with a few dozen, or even a few hundred, processors. But systems are being proposed that would exploit the extremely cheap processors that can now be fabricated, by using thousands, or even millions, of processors. Such systems pose enormous problems as to how they should be programmed and how configured. But they are extremely attractive as potential substrates for embedding cognitive models.

The possibility of ''reconfigurable'' sets of processors, in which the physical interconnections among the large number of individual processors are actually changed (quite analogous to telephone switching networks)[6] in order to best handle the flow of processes of a particular program, is also being examined by Lipovski and Tripathi,[34] Briggs et al.,[7] Siegel et al.,[62] and others.

Such computers would seem to be especially appropriate for exploring the underlying structure of the set of processes best suited to a particular type of system. Image analysis and at least the ''lower'' preprocessing levels of perception, where the very large array of input information (e.g., from a TV camera) must be contended with, can potentially be speeded up by several orders of magnitude if computed on large arrays of processors. The ''higher'' levels of perception, where symbolic, semantic interactions must be assessed, to build the more global structure of information needed to achieve recognition and description, are probably more appropriately handled by a network. These new arrays and networks should for the first time make it possible to run really adequate tests, and to handle moving objects.

V. ARRAY/NETWORK ARCHITECTURES WITH PYRAMID STRUCTURES

The very large arrays are already very fast. But the potential for further increases in speed during the next 10 to 20 years is enormous. VLSI fabrication techniques will almost certainly continue to increase into the foreseeable future at the rate of a doubling of the density with which transistor-equivalent devices are packed onto each silicon chip (the basic building-block of today's computers). An individual chip, when sold in large quantities so that its very expensive design costs can be paid back, costs only a few dollars. A computer built from 100, 1000, or even 10,000 chips is a relatively simple, straightforward, and potentially reasonably cheap affair.

In 1978 a packing density of 100,000 devices on a chip about 6 mm^2 was achieved, and by 1980 such chips were being sold in large quantities. Every 18 to 24 months this packing density can be expected to double (as has been the case for the past 10 or 20 years). So by 1985 or 1990 1 million device chips should be achieved, and by 1990 or 2001 we will see 10 million device chips.

A simple processor of the sort used in today's large arrays uses roughly 100 to 500 devices. So it will be possible to pack large numbers of such processors on a single chip, and to build relatively small and cheap computers, using a few hundred or a few thousand such chips, that are extremely large and powerful arrays.

The biggest problem will be the interconnections between chips. Today's chips are severely limited in the number of pins that can be used to connect the chip to its outside world. Only 64 or so pins are economically feasible. It is expected that this number will increase much more slowly than will the packing of devices on the chip. With the technologies that can be foreseen as relatively certain, it is likely that at most 128 or 256 pins will be possible.

Each chip will contain a subarray of processors. (Today's chips for arrays contain 4, 8, or in one case 32, processors.) The processors at the borders of each subarray must, when they must effect a window or other operation that fetches data from a pixel outside that border, fetch that data through the chip's pins. Thus, although a 16 by 16 array of processors (each made of 500 devices, each with, say a 1000 bit memory made of 1000 devices) could be packed onto a 1 million device chip, the 256 processors would probably have to share pins (pins must be used for power, control signals, and a variety of other purposes also).

On the other hand, once the processor array on a single chip can be made large enough, it can serve as a self-contained module, much like a facet in a larger eye. By large enough, I mean large enough to surround at least one complete object at a resolving power great enough to give the array computer enough detail to recognize most objects. For some objects, 20 by 20 or even 16 by 16 is sufficient (as attested to by the many programs developed in the 1960s to recognize letters of the alphabet and other objects using arrays of that size). A 32 by 32 or a 64 by 64 array, both of which may be feasible on a 10 million device chip, could certainly surround clearly resolved objects. Such a module could then work relatively independently, with only occasional need to pass data to other modules.

A severe limitation to arrays is the need to pass data only by shifting to the nearest neighbor, and then continuing to make successive near-neighbor shifts until the destination is reached. Reconfiguring networks have been designed to improve upon this (typically giving logN shifts rather than 2N shifts for an array with N processors). But such reconfiguring networks are very expensive, and have not yet been widely used.

Several attractive ways to overcome this difficulty have been proposed. Essentially, these suggest that the structure of the network mirror the structure of the processes that will be executed on the data the program is designed to handle, as those data flow through the processors onto which the program's processes are mapped.

Since this may sound cumbersome, when in fact (and partly because) it is almost transparently simple, I will restate it from a slightly different perspective: A program is a set of procedures, each of which acts upon its appropriate set of input data, and outputs its results. These results are either intermediate results, which are the input data for subsequent procedures, or final results, which are input data for some ''output device'' like a printer. The flow of data among procedures can be thought of as giving a structure to the program.

Consider the situation for a moment in a somewhat more abstract way: The procedures are vertices in a graph, and wherever a procedure outputs data that is input to a subsequent procedure, an edge is drawn to link those two procedures. Now concretize that graph by replacing each vertex with a microprocessor (that is, a little computer—let's ignore the issues of where the memory is placed, assuming that each processor has its own sufficiency of memory), and by replacing each edge with a cable of wires sufficiently large and fast to transmit the data output by one and input by the other fast enough so that procedures can execute without delay. Now this whole graph (let's call it a ''tightly coupled network'') of computers is the system that executes our program.

This suggests that the structure of the data-flow through the system should determine the structure of the total system, which therefore mirrors the (class of) problem(s) it was designed to handle. At least today, such a computer network will be to some extent specialized, rather than being equally efficient on all possible problems. For that reason, many people think of parallel networks and especially arrays as ''special-purpose''. But they are all general-purpose in the usual sense of that word—they are capable of executing any program that any other ''general-purpose'' computer (sometimes called ''Turing-machine equivalent'') can execute. Indeed all computers are to some extent specialized. That is especially true of the single-processor serial computer that today dominates. It is very good for the range of problems for which it was designed, but very bad and inefficient for others (e.g., searches for symbolic information in large list structures).

For pattern recognition and scene description, the very large array is (in my opinion) almost certainly extremely useful, and will be widely used once it becomes commercially available at a price that fits into the typical research budget for this kind of research. The only commercially available array today is the DAP, which costs $1,300,000 as an add-on to ICL's large $2 million to $5 million computers. CLIP4® was designed to be as cheap as possible (which is why it is a good bit slower than DAP, since it is built using a much cheaper technology), and negotiations are under way that may lead to its production for around $100,000 to $200,000 (it will use an LSI-11, costing $10,000 to $20,000, as host).

By 1983 arrays with 10,000 processors or more could certainly be built and sold at a profit for $100,000 or less—if their development costs could be amortized over 100 or more arrays. They would be extremely useful for many basic image enhancing and image processing tasks. If used in conjunction with a ''host'' computer that executed the higher-level procedures needed by a full-blown pattern recognition or scene description program, arrays offer the possibility of speeding up the program by several orders of magnitude.

That is, if the program lends itself to parallel computation. I would argue that visual perception is highly parallel. The observation that a living eye has millions of parallel input sensors (the rods and cones) that input information in parallel to a visual system of neurons that is also highly parallel, and that the TV cameras we typically use input similarly parallel data to the computer seems to me all that needs to be said.

Probably most people developing programs for image processing, pattern recognition, and scene analysis would agree. But at least one very important group, those researchers attempting to use typical ''artificial intelligence'' problem-solving techniques for perception, might possibly still disagree. I mention this because many researchers will find that their programs, or at least many of the procedures in their programs that take the most time (because they iterate pixel-cell by pixel-cell through the very large image array) can be converted without too much trouble to execute on a parallel array.

But there will always be some trouble. Arrays and networks are so different from traditional 1-CPU serial computers that one must be willing to learn and gain some experience in how to rethink programs for them. Often we have converted procedures that are, essentially, highly parallel into a more serial form that maps nicely onto a serial computer. Many of us have grown so accustomed to this process that converting back to a parallel form seems strange.

A. Systems with Several Converging Layers

A Single Instruction Multiple Data Stream Array alone will rarely be sufficient. One major reason, in my view, is because as an image is processed it is inevitably reduced. To put this another way, the central purpose of perception is to convert the very large raw input image to a very small set of symbols that name and describe the significant parts of that image.

This suggests a very simple extension to an array: several additional arrays, each successively smaller, so that each array outputs to a suitably positioned next array, that is, a multilayered structure of arrays. For example, a 64 by 64 array might output to a 32 by 32, which outputs to a 16 by 16, then an 8 by 8, then the more powerful host computer (which effects any additional operations). A single array could handle the successively smaller images, but only by slowly shifting them in each direction, nearest neighbor to nearest neighbor.

Such a system can be built with a variety of different plausible degrees of convergence. One of the most straightforward would have the following configuration: The image would be input to an array of small memories, e.g., 128 by 128 at which the 64 by 64 array of processors looked. If the image were larger, e.g., 512 by 512, one