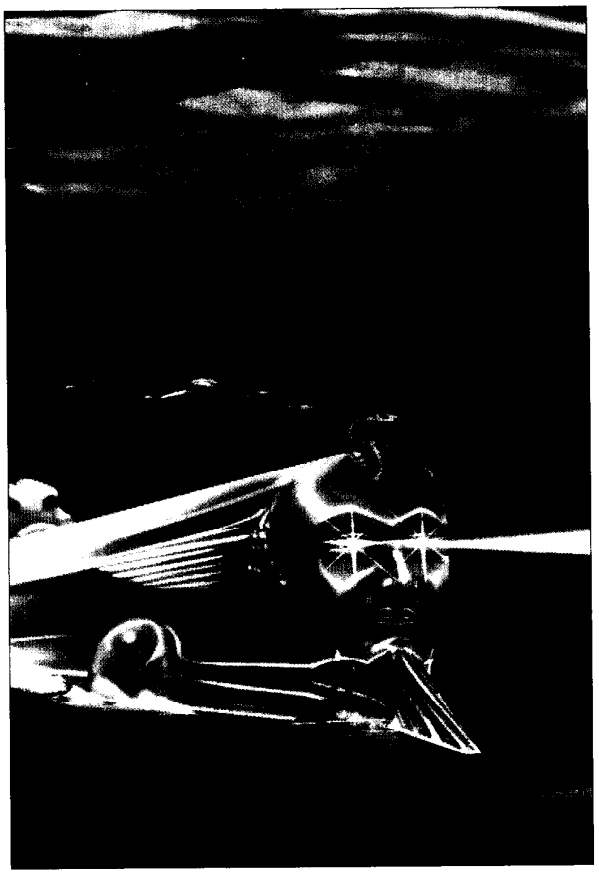
GUEST EDITORS' INTRODUCTION



Parallel Processing for Computer Vision and Image Understanding

Alok Choudhary and Sanjay Ranka, Syracuse University

Vision has long fascinated researchers from such disciplines as psychology, neural science, computer science, and engineering.[1] What exactly is vision? It has been defined as the process of recognizing objects of interest from images. The word *process* refers to some form of processing performed on the input data, which may be one or more images. The phrase *objects of interest* implies a context within which this processing takes place, as well as the presence of a representation used in this processing. For example, if we were asked, Is there a table in this room? we would compare the representation of a table contained in our minds with something similar to it in the room. In that process, we would ignore objects that did not look like a table. That is, we start with a model and look for some instance of the model in the room. On the other hand, if we were told, "Describe all the objects in this room," we would scan the room, form representations of the objects, and match them to objects in our "knowledge bases." However, we may or may not know what to expect in the room. Both problems are vision problems.

A general vision problem is considered to be an ill-posed problem from a computational perspective. The ultimate achievement of computer vision will be to perform the tasks normally performed by human vision.

The role of parallel processing

Problems in computer vision are computationally intensive. Consider a sequence of images at medium resolution (512 × 512 pixels) and standard frame rate (30 frames per second) in color (3 bytes per pixel).[2] This represents a rate of almost 24 million bytes of data per second. A simple feature-extraction algorithm may require thousands of basic operations per pixel, and a typical vi-

0018-9162/92/0200-0007$03.00 © 1992 IEEE

Levels of processing

Low. This level is normally termed bottom-up processing.[2] Most image-processing operations fall into this category. Input data includes images or simple transformations of images. Computations in low-level processing are regular, exhibit high spatial parallelism, and mainly involve numeric processing. These computations are well suited for both SIMD (single instruction, multiple data) and MIMD (multiple instruction, multiple data) architectures. Example algorithms include edge detection, filtering operations, and connected component labeling.

Intermediate. This category conveniently bridges bottom-up (low-level) and top-down (high-level) processing.[2] Computations in this category manipulate symbolic and numeric data. Examples of symbolic data include edges and lines commonly referred to as tokens. Processing on this level attempts to build a coalition of tokens to extract meaningful entities, for example, forming rectangles from lines. Computations are normally data dependent and irregular. They are suitable for medium- to coarse-grain parallelism in MIMD mode, although a subset of computations also can be performed efficiently on SIMD architectures.

High. Tasks on this level are normally top-down (or model-directed). However, processing is not so well defined as on the other two levels. The model of the "world" drives the processing. The world represents a database of objects, their possible poses, and interrelationships in a context. For example, a model of a car has descriptions of wheels, doors, etc., and constraints describing their relationships. Furthermore, processing in this domain may require reexecuting algorithms from the other two levels. Although parallelism on computations at this level is not well understood, it is believed necessary to dynamically schedule computations. The diversity and highly data-dependent nature of computations make this level of processing largely suitable for MIMD parallelism.

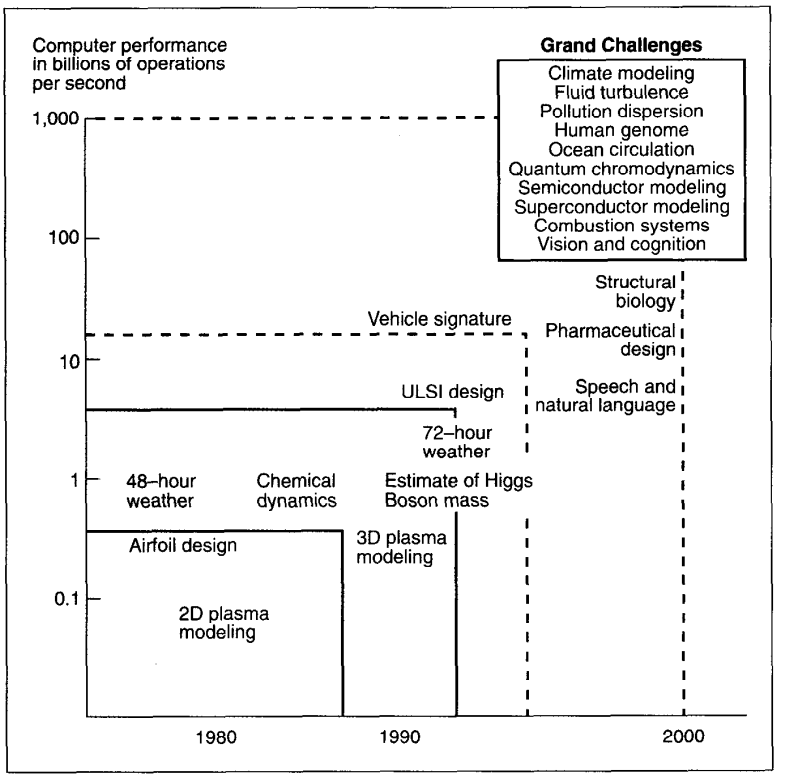


Figure 1. Performance requirements for "Grand Challenge" problems.[4]

sion system requires significantly more complex computations. As you can see, parallel computing is essential to solving such a problem.

In fact, the need to speed up image-processing computations brought parallel processing into the computer vision domain. Most image-processing algorithms are inherently parallel because they involve similar computations for all pixels in an image. This has inspired the development of array processors. For example, NASA Goddard Center's MPP[3] has a 128×128 processor mesh-connected array specifically built for image processing.

A mesh architecture connects each processor to four neighboring processors: north, south, east, and west. It is suitable for image processing because its structure closely mimics that of a two-dimensional image. It also provides an efficient *local* communication structure.

However, the lack of efficient *global* communication capabilities in a mesh architecture — and the requirement for fast top-down and bottom-up image processing — led researchers to propose another architecture known as a "pyramid." This architecture normally consists of several levels of meshes in which the top level has one processor and each succeeding level has four times as many processors as its parent array. In addition to the mesh interconnections within each level, each processor is also connected to its four children (except in the bottom layer) and to its parent (except for the root). Therefore, a pyramid architecture maintains several levels of image representations simultaneously. Both mesh and pyramid architectures have contributed significantly to the understanding and development of new algorithms for image analysis and vision, and have considerably influenced the subsequent designs of parallel architectures for vision.

Current status. Parallel processing has taken tremendous strides in the last

decade, enabling scientists to perform very large scientific computations that were impractical a few years ago. But the immense computational challenge presented by vision is still to be met. In fact, compared to the impact of parallel processing in other areas, its impact in the vision domain has been minimal for several reasons.

A typical vision system requires integrating algorithms from diverse areas such as image processing, numerical analysis, graph theory, artificial intelligence, and databases. There is no clear understanding and consensus on how to achieve this. Specific problems in integration can also be attributed to a lack of understanding of the vision process itself, even if the computations and parallelism of some individual components are well understood.

Currently, the dominant approach to characterizing vision computations is to classify the processing requirements into three levels: low, intermediate, and high. The most recent image-understanding benchmark[2] embodies this characterization. The accompanying sidebar presents a brief overview of each level.

Recent efforts in architectural design and development have embedded architectural components for each level of processing into one integrated architecture (as explained in the sidebar). Compared to the progress in architectural advances in general-purpose parallel processing for other scientific disciplines, however, architectural advances for vision systems are in their infancy.

Future directions

Artificial vision is one of the "Grand Challenges" identified by the federal government's High-Performance Computing and Communication (HPCC) initiative[4] (see Figure 1). Solving these problems is expected to require raw computational power between 100 and 1,000 billions of operations per second. Besides raw computational needs, other issues must be addressed to efficiently use parallel processing in solving vision problems.

Architectures. As discussed, vision problems span a broad spectrum of computations suitable for different types of architecture (like special-purpose, SIMD, and MIMD). The computations in various stages of processing not only need to execute concurrently in a system but also must interact with each other. Hence, the future entails some form of heterogeneity in an architecture for vision. The challenge for researchers is to capitalize on the advancements in multiprocessor technology and incorporate them optimally into an architecture suitable for vision problems.

System integration. A straightforward integration of different types of architectures into one organization may not be efficient for solving vision problems. The integration must incorporate appropriate control and communication structures suitable for providing the necessary interaction between various parts. It should also reflect the peculiarities of vision systems. In simple terms, an integrated organization should not be visible to a user as a collection of heterogeneous components but should provide a unified view of a machine in which heterogeneity is transparent to the user.

Programming models and software tools. Development of these items has lagged far behind progress made in architectures. Vision-processing requirements present a greater challenge because vision requires programming models and subsequent languages powerful and flexible enough to handle different types of processing while hiding the details of a heterogeneous system.

Software tools are necessary to provide users a friendly platform on which they can develop applications. If users are burdened with the responsibility of learning machine organization intimately in order to use these tools, the gains from parallel processing will be limited. Tools are also important for users to rapidly prototype and experiment with their algorithms. This is critical to developing and testing new techniques to solve problems in the vision domain.

Real-time vision. Examples of real-time vision applications include robotics and autonomous land vehicles. Such applications not only demand high performance but also require predictable performance and a capability to interact with the environment (as in active vision) through efficient interface with sensors. They also place physical constraints such as size, power consumption, cost, and robustness on a parallel system. Therefore, special-purpose parallel computing may yet have a role in such systems. ■

Acknowledgments

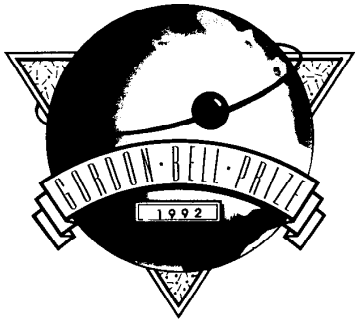
We thank Jon T. Butler, editor-in-chief of *Computer*, for his invaluable help in all aspects of this issue, including gathering reviews, interpreting their results, selecting papers, and shaping the theme. We also thank Bruce Shriver, the former EIC, for his encouragement and subsequent approval of this special issue. We acknowledge the support of the Center for Applications and Software Engineering, a New York State Technology Center at Syracuse University, Department of Electrical and Computer Engineering and School of Computer Science. We also thank the reviewers for their timely and invaluable reviews, and the authors for their contributions and discussions with us. We are grateful to Azriel Rosenfeld for sharing his views on the issue theme and for many enlightening discussions. We acknowledge support from Texas Instruments and IBM. Finally, we thank Grace Lanni and Carla Shaw for their professional handling of manuscripts and reviews.

In total, 25 submissions were received for this special issue of *Computer*. The eight articles selected are of high quality, but unfortunately, many other interesting articles could not be included.

This work was partially supported by DARPA under contract No. DABT63-91-C-0028. The ideas expressed do not necessarily reflect the position or the policy of the US government, and no official endorsement should be inferred.

References

1. D. Marr, *Vision*, W.H. Freeman and Co., San Francisco, Calif., 1982.
2. C. Weems et al., "The DARPA Image Understanding Benchmark for Parallel Computers," *J. Parallel and Distributed Computing*, Jan. 1991, pp. 1-24.
3. K.E. Batcher, "Design of a Massively Parallel Processor," *IEEE Trans. Computers*, Vol. C29, No. 9, Sept. 1980, pp. 836-840.
4. "Grand Challenges: High Performance Computing and Communications," report by the Committee on Physical, Mathematical, and Engineering Sciences, Office of Science and Technology Policy, 1991.



1992 Gordon Bell Prize

For Outstanding Achievements in the Application of Parallel Processing to Scientific and Engineering Problems

Entries are due **May 1, 1992,** with finalists to be announced by June 30 and winners announced at the Supercomputing 92 conference in November 1992. Prizes of $1,000 each will be awarded in two of three categories:

- Performance, based on megaflop rate on a machine with known performance compared against similar applications. If this is not possible, entrants should document their performance claims.
- Price/performance, based on performance divided by the cost of the smallest practical computational engine, including critical peripherals. Performance measurements will be evaluated as for the performance category.
- Compiler parallelization, based on the most speedup, measured by dividing the wall-clock time of the parallel run by that of a good serial implementation of the same job.

General conditions include demonstrating the utility of the program and machine. The judges will also consider how much the entry advances the state of the art in some field.

For more information or to enter, contact:

1992 Gordon Bell Prize
c/o Marilyn Potes
IEEE Computer Society
10662 Los Vaqueros Circle
PO Box 3014
Los Alamitos, CA 90720-1264
Phone: (714) 821-8380

Further reading

For more information on computer vision and image understanding, consult the following sources.

Journals

Computer

IEEE Transactions on Computers

IEEE Transactions on Pattern Analysis and Machine Intelligence

IEEE Transactions on Parallel and Distributed Processing

Journal of Parallel and Distributed Computing

Journal of Supercomputing

International Journal on Computer Vision

Computer Vision Graphics and Image Processing

Conference proceedings

Proc. Image Understanding Workshop, Morgan Kaufmann, San Mateo, Calif., (ongoing).

Proc. Conf. Computer Vision and Pattern Recognition, IEEE Computer Society Press, Los Alamitos, Calif., Order No. 2148, 1991.

Int'l Conf. Pattern Recognition, Vols. 1 and 2, IEEE Computer Society Press, Los Alamitos, Calif., Order Nos. 2062 and 2063, 1990.

Int'l Conf. Parallel Processing, Penn State Press, (ongoing).

IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Machine Intelligence, IEEE Computer Society Press, Los Alamitos, Calif., (ongoing).

Books

Choudhary, A.N., and J.H. Patel, *Parallel Architectures and Parallel Algorithms for Integrated Vision Systems*, Kluwer Academic Publishers, Boston, 1990.

Kumar, V., P.S. Gopalakrishnan, and L.N. Kanal, *Parallel Algorithms for Machine Vision and Intelligence*, Springer-Verlag, Berlin, 1990.

Kumar, V.K.P., *Parallel Architectures and Algorithms for Image Understanding*, Academic Press, Boston, 1991.

Ranka, S., and S. Sahni, *Hypercube Algorithms for Image Processing and Pattern Recognition*, Springer-Verlag, Berlin, 1990.

Parallel Computer Vision, L. Uhr, ed., Academic Press, Boston, 1987.

Alok Choudhary has been on the faculty of the Department of Electrical and Computer Engineering at Syracuse University, Syracuse, New York, since 1989. His research interests include parallel computer architectures, software development environments for parallel computers, and computer vision.

Choudhary received his BE degree in electrical and electronics engineering from Birla Institute of Technology and Science, Pilani, India. He obtained the MS degree from the University of Massachusetts, Amherst, and the PhD degree from the University of Illinois, Urbana-Champaign, both in electrical and computer engineering. He was a visiting scientist at IBM's T.J. Watson Research Center during the summers of 1987 and 1988. He is a member of the IEEE Computer Society and the Association for Computing Machinery.

Sanjay Ranka has been an assistant professor in the School of Computer Science at Syracuse University since August of 1988. He was a visiting scientist at the T.J. Watson Research Center during the summer of 1991. His main areas of interest are parallel and distributed computing, algorithms, software development environments for parallel computers, and neural networks.

Ranka completed his BTech in computer science and engineering at the Indian Institute of Technology, Kanpur, in 1985 and the PhD in computer and information science at the University of Minnesota, Minneapolis, in 1988. He is a member of the IEEE Computer Society.

Readers can contact Sanjay Ranka at 4-116 Center for Science and Technology, School of Computer Science, Syracuse University, Syracuse, NY 13244.