North-Holland Publishing Company
Microprocessing and Microprogramming 7 (1981) 119–123

Object Recognition Using a Data-Flow Computing System

G.K. Egan and C.P. Richardson

Department of Computer Science, University of Manchester, Manchester M13 9PL, England

Considerations of cost, performance, reliability and flexibility are leading to the increasing decentralisation of computing systems in control applications. A decentralised computing system based on the data-flow model of computation is applied, in conjunction with a laser range-finder, to the task of recognising a number of simple objects in a work-space. The recognition process involves a combination of active interrogation and multiple-hypothesis testing. Processing-elements in the system need not be identical and may be based on conventional microprocessors.

1. Introduction

Considerations of cost, performance, reliability and flexibility considerations are leading to the increasing decentralisation of computing systems in control applications [11, 12, 15, 16].

An area of particular interest is that of production systems where attempts to reduce work-piece positioning costs, and increase performance and flexibility, demand more sophisticated manipulative and sensory mechanisms coupled with the computing systems to support them.

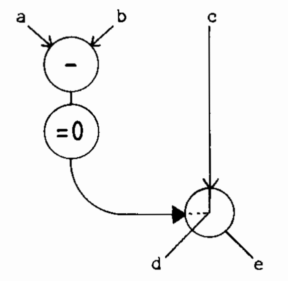
We describe how a decentralised computing system might be used in conjunction with a laser range-finder to recognise objects in a work-space [14]. The computing system [5, 6] is based on the Data-flow model of computation and seeks to overcome many of the problems inherent in conventional systems [2]. Processing-elements in the system need not be identical and may be based on conventional microprocessors. A critique of other major dataflow architectures may be found in [5].

2. Data-Flow

Data-flow [1, 10] uses a finite directed-graph to describe a computation. The edges or *arcs* of the graph are queues of data directed from one *node* to another. The nodes represent functions which map input data onto output data.

Data flows down the arcs as packets or *tokens,* each node requiring a specific number of tokens to *trigger* the node-function's evaluation. The evaluation or *firing* consumes tokens from the input arcs and places result tokens on the output arcs. The number of nodes eligible for firing at any instant depends *only* on the availability of data.

In addition to the more primitive node-functions (arithmetic, logical etc.), node-functions exist for the data-dependent control of computational paths through the graph. See Fig. 1.



`if (a-b) = 0 then c -> d else c -> e`

Fig. 1. Example of Computational Path Control.

3. The Computing System

The computing system used in this study [5, 6] may be characterised as follows:

1) Tokens are *strongly* typed and of variable length. They carry not only the data used by the graph during its evaluation but the *node-descriptions* which comprise the graph itself.

2) Node-functions are *weakly* typed and accept a set of argument token-types; this increases graph generality while reducing the size of the node-

function set. Type coercion is performed automatically where 'sensible'.

3) Input–output is accomplished using predefined nodes. The *names* of these nodes are associated with particular input or output devices, which in turn are associated with particular processing-elements. As these nodes already 'exist' within the system, they must be linked into the graph at load or evaluation times this is done by sending *response-destination* tokens to the appropriate nodes.

4) Because of the association of input–output devices with particular processing-elements, and the system's probable geographic distribution, graphs are partitioned and the partitions allocated *statically* to processing-elements.

5) Storage nodes are provided to allow the graph to retain 'semi-permanent' information.

6) Information describing *exceptions* occurring during graph evaluation is communicated to the graph using the token-type *?*, or don't-know.

7) The system supports shared sub-graphs in sufficient generality to allow multiple recursion. Tokens involved in concurrent invocations of a shared sub-graph are separated by means of a *copy* number. The copy number is computed and appended only to tokens actually sharing a sub-graph. Less sophisticated processing-elements at the periphery of the system need not support the mechanism.

8) The hardware underlying the system consists of an arbitrarily large number of processing-elements, communicating over channels which *may* be asynchronous and unsophisticated.

For this study the computing system, laser range-finder and object space were simulated on a large conventional computing system [7]. The simulator, written in Pascal [9], can evaluate graphs with the order of 16000 nodes while modelling 128 processing-elements. Graphs are evaluated by the simulator at approximately 500 nodes/second.

4. The Laser Range-Finder

Laser range-finders can be used to interrogate a work-space *directly* [3, 8]. In doing so they have distinct advantages over more common video-camera based techniques which tend to be both *indirect* and storage intensive.

For this study the range-finder is assumed to be at the origin of a cartesian work-space. Objects rest on the positive X–Y plane and the laser may be vectored in the positive X–Y (θ) and Y–Z planes ($\emptyset$). The range-finder returns the distance from the origin to the spot illuminated by the laser.

5. Object Recognition

Object recognition, like many control problems, exhibits a high degree of parallelism; a data-flow system should be capable of exploiting this parallelism.

5.1. An Approach Suitable for a Data-flow System

Employing a data-flow system precludes the use of any storage-intensive algorithms. Data is produced and consumed in the evaluation of a node-function; there are no global variables. Scene analysis techniques such as those proposed by Nitzan et al. [13] cannot be used on a data-flow system without introducing considerable inefficiency.

Ishii and Nagata used a laser-tracker to actively interrogate a work-space [8]. The interrogation was driven by a multiple hypothesis process, each hypothesis corresponding to an object in the set of recognisable objects. Their method, which is not storage intensive, lends itself well to data-flow systems where all hypotheses may be tested simultaneously.

5.2. The Range-finder and Object Set

The range-finder is represented in the system by a two-input node. This node accepts two integer operands and returns a real token corresponding to the distance to the spot illuminated by the laser; a negative range is returned when the laser misses.

The range-finder node is embedded in a shared sug-graph to accurately represent the bottleneck of a single resource.

To aid in producing the recognition graph the following assumptions were made about the visual environment:

1) All objects lie within a cube with the range-finder at one corner.
2) All objects are in the specified object set.
3) Objects may overlap.
4) Objects are not placed one on top of the other.
5) The objects do not move!

For this initial study the object set was kept to three objects mathematically easy to represent. These objects were a sphere, a cone and a cylinder.

5.3. The Processes Involved in Object Recognition

The task of recognition may be divided into *five* distinct processes:

1) Finding an object: This process was implemented as a *doubly* recursive sub-graph taking as input parameters an interval in θ. A vertical line bisecting the interval is scanned. Should a hit be obtained (indicated by a range discontinuity) the value of θ is returned. Otherwise the two intervals generated by bisection are processed simultaneously in the same manner.

2) Delimiting the object: Once a hit on an object is obtained, parameters must be generated for passing to the hypothesis tests. The parameters chosen for specifying an object to the tests were the four bounding angles θ.left, θ.right, Ø.top and Ø.bottom. These are obtained through literative routines which scan the object vertically or horizontally at a particular interval seeking the edge. When the edge is passed, the interval is halved and the direction of scanning reversed. The edge of the object is thus found and the bounding angles obtained.

3) Resolution of clusters: At this stage checks are made to ensure that the object is not in fact a cluster of two or more objects. This is done by inspecting the range profile for discontinuities. Should such a break be found the parameters are adjusted so that the objects furthest away in the cluster are ignored and the parameters for the nearest object are sent on to the hypothesis testing sub-graphs.

4) Hypothesis testing: Each hypothesis is tested in a similar way. A set of angular coordinates over the object are generated and the range profile for these points is predicted. The actual range profile is then obtained using the range-finder and the two sets of range values compared. Statistical analysis is then carried out on the differences to produce a probability that the object is the one being tested for.

5) Results: A result is then obtained by voting on the probabilities issued by the hypothesis sub-graphs. The appropriate action on recognition may then be taken.

6. Results

6.1. A Typical Run

The entire object recognition graph required about 2000 nodes of which 70% were one-input nodes. The large percentage of one-input nodes means less token matching overheads on two-input nodes than might intuitively be expected. Figure 2 shows the processing-element activity for a typical run.

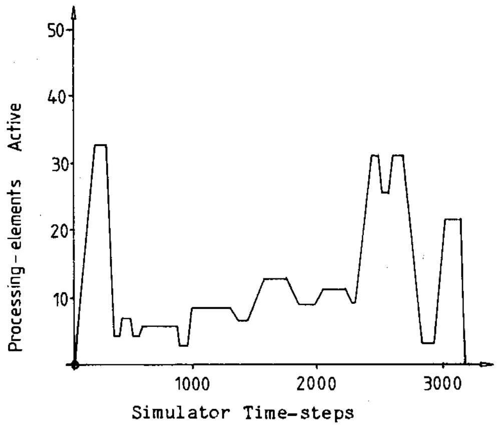


Fig. 2. Processing-element Activity for a Typical Run.

There were two objects present in the space which did not overlap. The activity shown is variable. Average achieved parallelism was 13.2 but at points on the graph the parallelism went as high as 30.0. These points were where multiple hypothesis testing (C) was carried out and the point where the recursive search for objects was active (A).

It is important to note that the object recognition process has, in approximately 3000 time steps, located and recognised *all* objects in the work-space; an average time step, or node evaluation time, of 20 uSec. is not unreasonable with bit-slice microprocessor-based processing-elements.

The variable nature of the activity indicates under-utilisation of the system which is to be expected given the small number of objects in the object set. Normally the object set would be larger and the system would support other sensors and devices including perhaps a manipulator.

Construction of the object recognition graph was straight-forward. A comprehensive macro-asembler capable of planting the primitives for shared sub-graphs was written [14], which enabled the graph design to proceed in a modular and structural fashion.

A point to note is the small size of the graph in terms of nodes or machine instructions; there is no need for a complex operating system to schedule the system's workload as this is determined solely by the availability of data.

6.2. *Effects of Partitioning and Resource Sharing*

The laser range-finder is controlled by a sub-graph in the overall object recognition graph or program. This sub-graph is in practice shared by the various hypothesis sub-graphs.

To illustrate the effect of graph partitioning and resource sharing (the laser range-finder) we consider three simple cases:

1) The normal case where the nodes of the sub-graph controlling the range-finder are assigned to separate processing-elements. No other graph nodes are assigned to these processing-elements.

2) The case where this simple partitioning measure is not performed and the range-finder sub-graph nodes are distributed randomly among the processing-elements along with the rest of the object recognition graph nodes.

3) The case where invocations, or calls, of the range-finder sub-graph are replaced by a copy of the sub-graph.

The graphs of average achieved parallelism against number of stimuli (external simultaneous requests to the graph to recognise an object in the space) was plotted for all three cases and is shown in Figure 3.

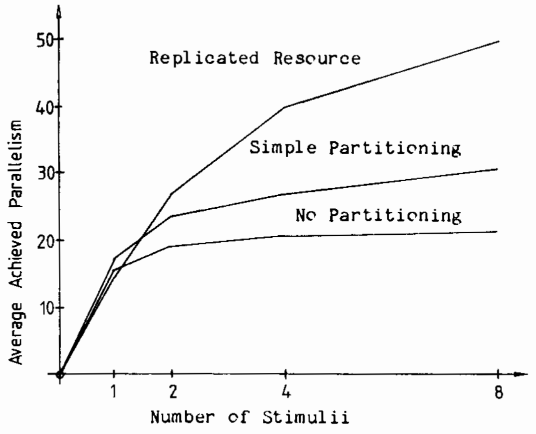


Fig. 3. Parallelism Against Number of Stimuli for Cases of Normal, No Partitioning and Replicated Resource.

7. Discussion

The initial results obtained in this study have been encouraging. The Data-flow computing system used seems quite adequate when dealing with what is a real problem; the class of similar problems is large.

While graph partitions are usually suggested by the problem structure, additional research into partitioning strategies is necessary if the full potential of the system is to be realised. The assembler used was adequate for this initial study but a need for languages better matched to the application is indicated.

Current research with the system includes the following:

1) Application-specific languages and their compilation on data-flow systems.
2) A detailed study of system aspects associated with time and non-determinacy.
3) 'Fast' processing-elements.
4) Communication techniques for systems of closely coupled processing-elements.
5) Computer assisted graph partitioning with practical constraints.
6) Manipulator control algorithms.

Acknowledgements

The authors wish to thank the University of

Manchester's Department of Computer Science for the provision of computing and engineering facilities during the course of the research leading to this paper.

C.P. Richardson would also like to thank the government of Barbados for its financial support.

References

[1] D.A. Adams, 'A Model for Parallel Computations', in Hobbs [ed] *Parallel Processor Systems, Technologies and Applications,* Spartan Books, 1970

[2] J. Backus, 'Can Programming be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs', CACM Vol. 21 No. 8, pp. 613–641, Aug. 1978.

[3] H.J. Caulfield et al., 'Laser Stereometry', Proceedings of the IEEE, Vol. 65, No. 1, pp. 84–88, Jan. 1977.

[4] P. Coiffet et al., 'Real Time Problems in Computer Control of Robots', Proceedings of the 7th International Symposium on Industrial Robots, pp. 145–152, Oct. 1977.

[5] G.K. Egan, 'A Study of Data-flow: Its Application to Decentralised Control', Ph.D. thesis, Dept. of Computer Science, University of Manchester, 1979.

[6] G.K. Egan, 'A Decentralised Computing System Based on Data-Flow', Proceedings of the IECI'80 Conference, Mar. 1980.

[7] R.N. Ibbett and P.C. Capon, 'The Development of the MU5 Computer System', CACM Vol. 21 No. 1, Jan. 1978.

[8] M. Ishii and T. Nagata, 'Feature Extraction of Three Dimensional Objects and Visual Processing in a Hand–Eye System Using a Laser Tracker', Pattern Recognition, Vol. 8, pp. 229–237, 1970.

[9] K. Jensen and N. Wirth, Pascal-User Manual and Report, Springer-Verlag, New York, 1975.

[10] R.M. Karp and R.E. Miller, 'Properties of a Model for Parallel Computations: Determinacy, Termination and Queueing', SIAM J. Applied Mathematics, Vol. 11 No. 6, pp. 1390–1411, Nov. 1966.

[11] J.Y.S. Luh and C.S. Lin, 'Multi-processor Controllers for Mechanical Manipulators', Proceedings of the COMPSAC'79 Conference', pp. 458–463, Nov. 1979.

[12] R. Mori et al., 'Microcomputer Applications in Japan', IEEE Computer, Vol. 12 No. 5, pp. 64–74, May 1979.

[13] D. Nitzan et al., 'The Measurement and Use of Registered Reflectance and Range Data in Scene Analysis', Proceedings of the IEEE, Vol. 65, No. 2, Feb. 1977.

[14] C.P. Richardson, 'Object Recognition using a Data-flow Machine: Algorithms for a Laser Range-finder', M.Sc. dissertation, Dept. of Computer Science, University of Manchester, 1979.

[15] N.R. Sandell et al., 'Survey of Decentralised Control Methods for Large Scale Systems', IEEE Transactions on Automatic Control, Vol. AC-23 No. 2, pp. 108–128, Apr. 1978.

[16] B. Shimano, 'VAL: A Versatile Programming and Control System', Proceedings of the COMPSAC'79 Conference, pp. 878–883, Nov. 1979.

Dr. Gregory K. Egan has been pursuing research into the design and application of *decentralised* data-driven multiprocessor systems since 1976. Initially his research was conducted at Manchester and is now being continued at the Royal Melbourne Institute of Technology's department of Communication and Electronic Engineering in Australia.

Christopher P. Richardson is a student of Computer Science at Manchester where he is pursuing research into the application of a dataflow system to the problems of manipulator control.