SURFACE TRACKING WITHIN THREE DIMENSIONAL DATASETS USING A GENERALISED MESSAGE-PASSING SUB-SYSTEM

M G Norman
R B Fisher

DAI RESEARCH PAPER NO. 387

In Developments using Occam. Ed. Kerridge.
IOS Publishers, Amsterdam, 1988.

Surface Tracking within Three Dimensional Datasets Using a Generalised Message-passing Sub-System

M.G. Norman
Department of Physics
University of Edinburgh

R.B. Fisher
Department of Artificial Intelligence
University of Edinburgh

Abstract

This paper describes an implementation of an efficient surface tracking operation within three dimensional medical images on a transputer network. The operation is an instance of a more general class, where processing is widely localised (but not global) within a dataset that itself must be distributed across a network of processors, because of size and a need to distribute processing. The communications strategy used in the program is essentially independent of the application and has been abstracted to form a general purpose communications harness.

Three dimensional medical images are analogous to two dimensional images stacked on top of each other. Surface tracking within these images spreads out unpredictably from a seed point connecting new points according to purely local criteria.

It is not possible to divide the data between processors according to the simple approaches used in dividing two dimensional images, not least because the images are three dimensional whereas four-connected transputers tend to form two dimensional networks. In addition the surface tracking operation is local, and efficient parallelisation requires that many processors are active when processing is occurring in only a small part of the dataset. The quantity of data also entails distribution.

The above considerations mean that the distribution of data between processors is relatively complex. Hence, as the surface tracking (and other actions) occurs, communications must pass between processors which are not necessarily nearest neighbours in the processor network.

To support this, a double buffered communications harness was developed to route messages across an arbitrary network of processors. Message routing is performed along the shortest route(s) across the processor network. Decisions are made locally at each processor, and take into account the recent loading of the respective channels. There is also a facility to use the space in the double buffering of the surrounding network if the local buffering at any processor becomes full.

Each processor generates in parallel a representation of the network during the initial stages of the program by searching the physical configuration of processors on which the program has been implemented. It uses this information to route messages between processors, making the calls to the communications harness completely independent of processor configuration.

Introduction

Over the last few years there has been a great deal of interest in the possibility of three dimensional graphical presentation of medical images. [1,2,3] This paper concentrates on the data distribution and interprocessor communication required to solve this problem on arrays of transputers. Three medical diagnostic tools (magnetic resonance imaging, positron emission tomography and computerised tomography) produce tomographic or three dimensional images. These images are analogous to two dimensional images stacked on top of each other, and correspond to the measured intensity of various parameters within the space being imaged. Each stacked image is referred to as a **section**, and image elements (which are considered to be three dimensional) are referred to as **voxels**.

Although there are many different ways of generating graphical three dimensional presentations of such datasets. Our research has concentrated on the approach of identifying surfaces

within the dataset and using conventional three dimensional graphical techniques of polygon display with hidden surface removal and shading to display the detected surface.

Operators for surface detection within three dimensional datasets have been developed by extending two dimensional edge detection operators such as the Sobel [3,4,5,6] Canny [7,8] and Walsh based [9,8] operators to detect gradients in three dimensions. Such operators measure the strength of surface and the orientation of surface at a given voxel within the dataset.

In order to generate a representation of a continuous surface for polygon based display from such measures of surface intensity it is necessary to track surfaces within the dataset. Tracking is a purely local process where neighbours of voxels which have been found to be on the surface being generated are considered for inclusion in the surface. Neighbours will be included in the growing surface if:

1. The detected surface strength is greater than some threshold.

2. The detected surface orientation matches the orientation of their parent voxel.

3. The intensity of the data bounded by the surface is consistent with a locally weighted average of intensities of bounded data.

There remains, of course, the problem of determining the first voxel which to be included in the surface. This is supplied interactively by the user who specifies a voxel which is considered to be on a surface, and the program finds all the voxels which form a surface continuous with that voxel.

To simplify the problem of comparing a neighbour with its parent, and also to reduce the space requirement for queues etc. every immediate neighbour of a voxel is considered for inclusion before any more distant neighbour. The result is analogous to a breadth first search through the dataset for all voxels which are on the surface.

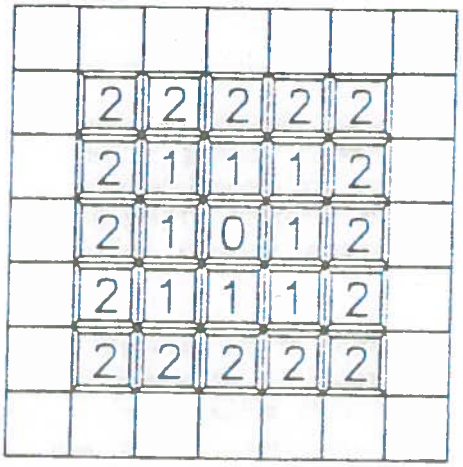

Fig 1: Surface tracking from a point on a flat surface: Generations of a breadth first search process.

With such a surface tracking algorithm all the processing is local to neighbourhoods of voxels under consideration, but that initially, at least, processing is restricted to a small part of the dataset which is close to the seed voxel. In order to visualise the way in which the processing spreads, it is useful to consider the case of surface tracking of a sphere within the dataset. Ignoring the discrete nature of the data, the spread of processing may be visualised as a circle moving across the surface of the sphere starting at the seed point. Each point on the advancing circle may only be determined by consideration of points closer to the seed point, and the breadth first nature of the tracking process ensures that no part of the circle may advance at a faster rate than any other point.

Distribution of Data

The large size of medical tomographic datasets mean that in any practical transputer based system, the data must be distributed between processors. At present there is simply not enough available RAM on each processor to store the whole dataset. The division of processing between processors must therefore exploit some form of geometric parallelism. Simple approaches to geometric parallelism within three dimensional datasets would assign to each processor a single contiguous portion of the dataset but it is found to be inefficient to use such a simple data distribution. The surface tracking algorithm is an instance of a general class of algorithm identified by Fox [10] as asynchronous but spatially connected, where the flow of processing within a computation space is determined by the processing itself. To efficiently divide the processing between processors for such a computation more complex approaches are required.

In determining the optimal assignment of data to processors for a three dimensional image processing package it was also necessary to consider operations other than surface tracking. Examples of such operations were global smoothing and thresholding operations where the whole of the dataset was to be operated upon. And two dimensional display operations where a single section was to be operated upon. The problems of moving the large datasets among processors between operations seemed to indicate a single assignment of data should be made for all the operations which were to be performed upon the data. The approach to distributing data which was implemented centered around partitioning the dataset and allocating many spatially separated blocks to the individual processors according to certain constraints

1. The proportion of the dataset assigned to each processor was approximately equal.

2. The proportion of each section assigned to each processor was approximately equal.

3. The blocks of data assigned to each processor were widely distributed in the dataset.

The optimal grain-size of the scattering is critically dependent upon the relative overheads of processing within a block and communicating between processors at the boundaries of each block. For twenty five processors processing a dataset of 128×128×12 voxels, a block size of 8×8×4 voxels was chosen. Each block was supplied along with a shell of voxels around it one voxel thick to allow local surface-detection operators to be applied without any communication between processors to which adjacent blocks were assigned. As an aid to improving the levels of processor loading during surface tracking it was decided to stagger the block origins between layers in the Z dimension. This results in each block being face-connected to one block across four of its six faces, and to four blocks across each of the other two faces. Each face-connected neighbouring block was assigned to a different processor, and to a different one to that to which the block itself was assigned.

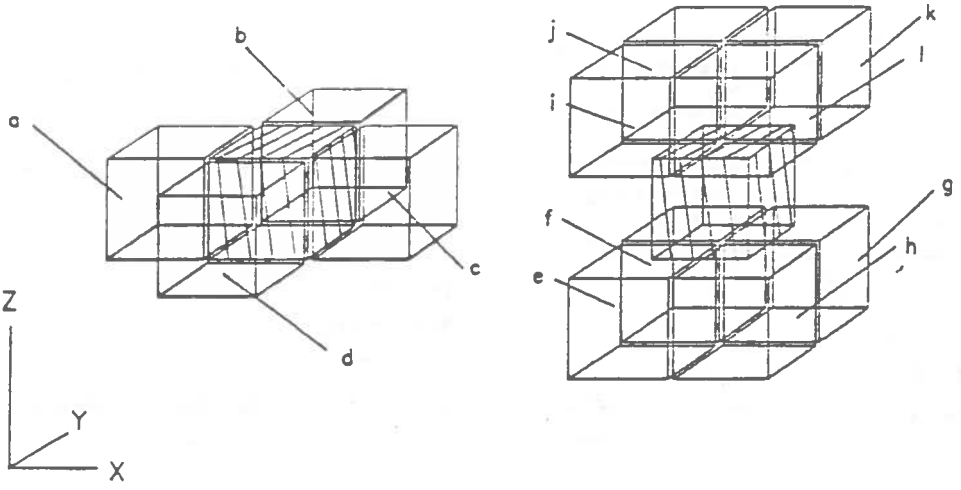

Fig 2: The assignment of data to processors
Each of the blocks shown contains 8×8×4 voxels in the x, y, z dimensions.
All the blocks shown are assigned to different processors.

Communications Between Processors

The surface tracking operation, as it tracks across boundaries of blocks, requires communications between the processors to which those blocks have been assigned. The distribution of data outlined above clearly aims to have each block of data face-connected to blocks of data assigned to twelve distinct processors, each of which is different to the processor to which the block is itself assigned. Each processor may therefore be forced to communicate with twelve other processors, which may not be merely its nearest neighbours in an array of processors. In advance of the program running and such parameters as the size of the dataset being processed becoming known, it is not possible to predict the exact identities of the processors to and from which a given processor will have to communicate. It was therefore decided to allow any processor to communicate with any other.

The surface tracking process and indeed the other parts of the program allow for processing to proceed on an communications driven basis. In this programming style, the arrival of a message at a processor causes a processing event to happen, which may optionally cause the generation of more messages to be sent to other processors. There is a master processor with which the file-system and the user communicates and which inputs the data onto the network and controls the flow of computation according to user input via a menu. It was arranged for all possible messages to make sense at any stage of the program (although not necessarily to cause execution of the same process), and for each messages to be transmitted as a complete unit. This allowed the use of a simple input buffer at each processor and obviated the necessity for a handshake between communicating processors.

The communications software allows a processor to specify that two packets it was sending to another processor should arrive in the same order as they were sent, by assuring that they would be sent along exactly the same route along which there would be no overtaking. This mechanism for allows a degree of loose synchronisation between processors.

In addition to the input buffer mentioned earlier, several other buffers were required to allow the communications to work without deadlock. Each of the physical transputer links was given a software buffer to allow for the possibility of bidirectional asynchronous communications on all four links. Each processor was also given a transit buffer to avoid the program deadlocking when communication was required around loops in the network. The communications processes may be described as below.

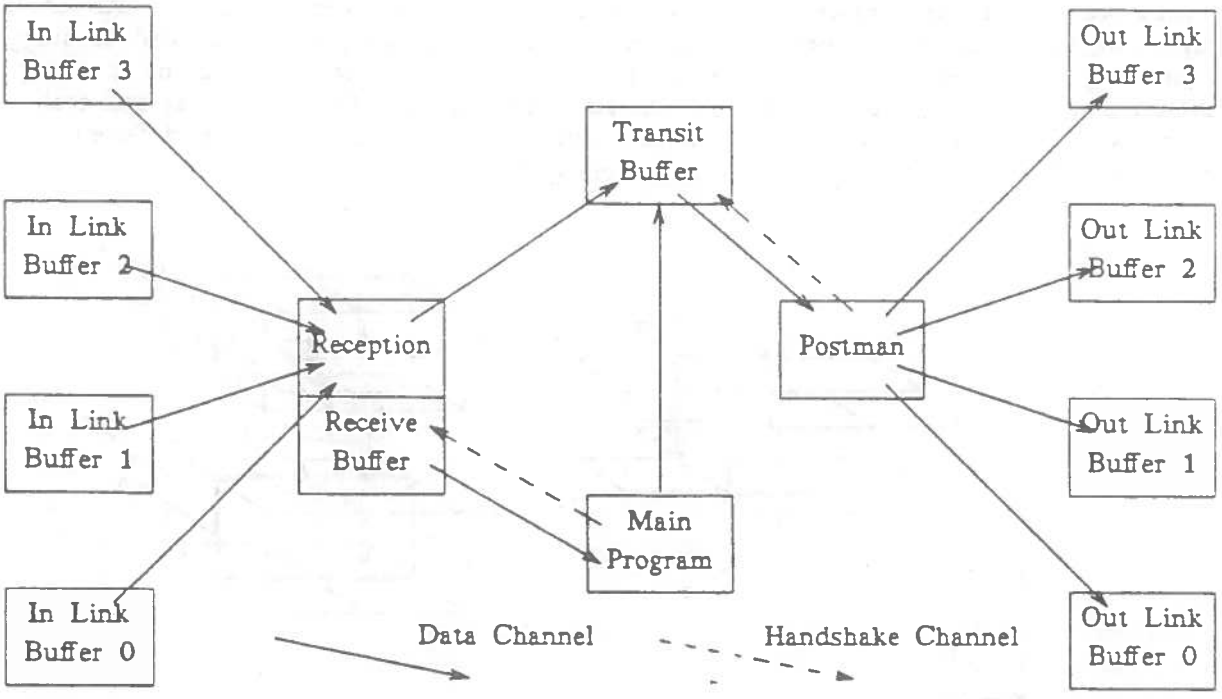

Fig 3: Communications processes and channels.

The above discussion has omitted to mention the problem of routing messages within a network communicating with such software. In order to allow experimentation with processor network topology it was decided to allow the network of processors itself to determine the optimal routing of packets between processors, and in later versions of the program to work out the topology of the network directly, without resort to any information supplied by the programmer. In practise this means that one may change the topology of processors upon which the program is running simply by changing the top-level PLACED PAR statement read by the configurer.

The algorithm used by the network to search the configuration on which it has been implemented approximates to:

```
PLACED PAR i = 0 FOR processors

  ...  some arbitrary channel placement statement

    SEQ
      ... determine which of my links are connected to other processors

      PAR
        ... broadcast my identity along those connected links

        WHILE processors.known < processors

          SEQ
            ... read in on my incoming links

            IF
              ... I have only heard about this packet's source by longer routes (if at all)
                ... record how I came by it and pass it out on all connected links

              ... I have heard about the packet's source  by a shorter route
                ... absorb it

              ... I have heard about the packet's source by an equally short route
                ... absorb it but record how I came by it

      ... read in and transmit data packets according to known shortest routes
```

Each processor records about each other processor a list of links via which packets may be sent to take them one step closer to their destination. This information is acquired since during the start up phase each processor hears at least once from each other processor, and records the links upon which the messages were received.

In order to visualise the network search algorithm, which is highly parallel, each processor may be thought to be sending out a signal of its identity which propagates outwards like a wave across the network of processors,. The propagation is, however, determined by a decision made at every node in the network. The wave is never allowed to back-track and as soon as it reaches areas of the network that it has already covered it is blocked. Each processor is eventually touched by the wave, and stores a list of the direction(s) from which the wave came. It may then assume that a signal passed out in those direction(s) will be moving one step closer to the source of the wave.

Results and Conclusions

The program was implemented in occam on a Meiko M40 Computing surface with twenty five inmos T414 transputers involved in processing the image and three others used for controlling

the program and displaying results. The assignment of data to processors described above, produces high levels of processor loading for global and section based operations. Such operations tend to benchmark over one hundred times faster than a similar program running serially in C on a Sun 2 workstation. While it has not proved possible to exhaustively benchmark the surface tracking operation, models of processing would indicate a processor utilisation rate in excess of 50% for tracking moderate sized objects.

The communications software has proved to be reliable in operation and to organise itself effectively for arbitrary networks of processors. As there is nothing application specific about the communications software it has been successfully abstracted into a utility which is available to users of the Edinburgh Concurrent Supercomputer, and is currently incorporated in a number of other working programs.

List of References

1. "Special Issue on CT," *Proceedings IEEE.* , Vol. 71 (3) pp 291-448.

2. G.T. Herman, H.K. Lieu, "Three dimensional display of human organs from computer tomograms," *Computer graphics and Image Processing*, Vol. 9 pp1-121, 1979.

3. A. Nelson, "Body scan data surface detection and presentation," *MSc Dissertation 1985,University of Edinburgh, Department of Artificial Intelligence.*

4. I. Sobel, "Camera Models and Machine Perception," *AIM-21*, Stanford A.I. Lab, (May 1970).

5. S.W. Zucker, R.A. Hummel, "An optimal three-dimensional edge-operator," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-4 pp41-50, 1982.

6. M.G. Norman, "A three dimensional image processing program for a parallel computer," *MSc Dissertation 1987, University of Edinburgh, Department of Artificial Intelligence.*

7. J.F. Canny, "Finding edges and lines in images," *MSc Thesis. M.I.T. , Cambridge, 1983..*

8. D.P. Reeve, "Walsh and Canny based surface detection in body scan data," *MSc Dissertation 1986, University of Edinburgh, Department of Artificial Intelligence.*

9. F. O'Gorman, "Edge detection using Walsh functions," *Artificial Intelligence*, Vol. 10 pp215-223, 1978.

10. G.C. Fox, W. Furmanski, "The Physical Structure of Concurrent Programs and Concurrent Computers," *Proceedings Royal Society A.*, Vol. (To be Published).