1988-09-25

A Parallel-Processing Subsystem for Rapid 3-D Interpolation of CT Images

William A. Barrett
william_barrett@byu.edu

Stephen J. Allan

Scott R. Cannon

Original Publication Citation

Scott R. Cannon, Steven J. Allan, and William A. Barrett. "A Parallel-Processing Subsystem for Rapid 3-D Interpolation of CT Images," IEEE Proceedings of Computers in Cardiology, pp. 75-78, Bethesda, MD., 1988.

A PARALLEL–PROCESSING SUBSYSTEM FOR RAPID 3–D INTERPOLATION OF CT IMAGES

Scott R. Cannon, Stephen J. Allan, William A. Barrett†

Computer Science Department, Utah State University, Logan UT
† Computer Science Department, Brigham Young University, Provo UT

Abstract

An inexpensive parallel–processing subsystem for the rapid interpolation of CT image planes is demonstrated with a variety of node topologies. The subsystem is based on a tree network of INMOS T414 Transputer processors and is hosted by an AT–based image workstation. The subsystem accepts a stack of eight arbitrarily–spaced 256 × 256 image planes from the host. Subsystem output to the host consists of a stack of 32 scaled and evenly–spaced image planes (256 × 256 × 32 with cubic voxels). Benchmark execution times ranged from 12.3 seconds for three nodes to 5.8 seconds for eight nodes.

Introduction

Left ventricular angiography is generally considered the standard of reference in assessing ventricular performance and regional wall motion in the left heart. Analysis based upon 2–D projections however, is marginal at best in its ability to describe 3–D geometry. Recently, significant interest has been generated in less invasive, more repeatable, and more informative Cine CT technologies and displays for assessing cardiac function [1, 2].

Cine CT is able to generate a series of eight contiguous axial image planes through the heart with significant resolution through intravenous contrast injection. Image planes are approximately eight mm thick and generally not uniformly spaced. Cine CT images are however, still generally viewed and analyzed in conventional 2–D formats.

Barrett [3] recently demonstrated a system for the generation and dynamic display of true 3–D ventricular surface anatomy. For each static 3–D image in a dynamic series, this system consists of three tasks;

1. the interpolation and scaling of a stack of eight 256 × 256 arbitrarily–spaced image planes into a stack of 32 evenly–spaced 256 × 256 image planes scaled to represent cubic voxels.
2. the detection of surfaces within the image plane stack, and
3. the display or rendering of the 3–D surface geometry.

Time requirements for these processes are considerable. On an AT–class (80286) machine, more than one hour may be required for the interpolation and scaling task alone. An 80386–80387 class computer still requires greater than 14 minutes for this same task.

This paper describes the demonstration and testing of an inexpensive parallel–processing subsystem for an AT–class host intended to significantly reduce processing time requirements for the image interpolation and scaling step in order to make dynamic 3–D imaging of ventricular anatomy more clinically practical.

Methods

In order to interface with the existing dynamic 3–D CT imaging system previously described [3], the subsystem requirements were the following:

1. provide a simple interface to the existing AT–class image workstation and software,
2. be easily programmable in a high–level language, and
3. be reasonably inexpensive with respect to the image workstation.

The parallel–processing subsystem was constructed using eight INMOS T414 Transputer processor boards developed at Utah State University. (A similar version of the Transputer board is now commercially available at approximately $1K/node.) Each T414 is a 32–bit 10 MIPS processor with four inter–transputer 10 Mbit/sec DMA links and 2k on–chip cache RAM. Each processor board contains one MB memory and provides a standard backplane bus connection. Each board runs asynchronously and independently of others. Inter–transputer link connectors are provided at the front of each board so that boards may be configured into a wide variety of topologies. A separate AT–bus interface board provides one inter–transputer link to the network.

Software provided by Logic Systems supports a C compiler, linker, and network loader for software development. A device driver and MS–DOS interface for the AT–based network link (again provided by Logic Systems) allows software control and tasking of the network subsystem during execution.

The interpolation and scaling step is inherently parallel if interpolation is performed without regard to pixel neighbors (Figure 1). Execution was performed using integer math (shifts and

0276-6574/89/0000/0075$01.00 © 1989 IEEE

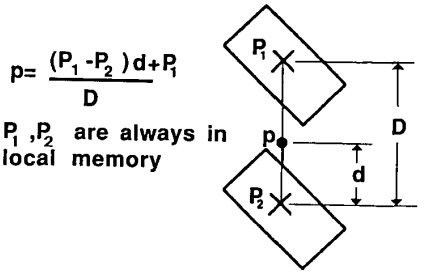

Figure 1: Interpolation formula used in interpolating nodes.

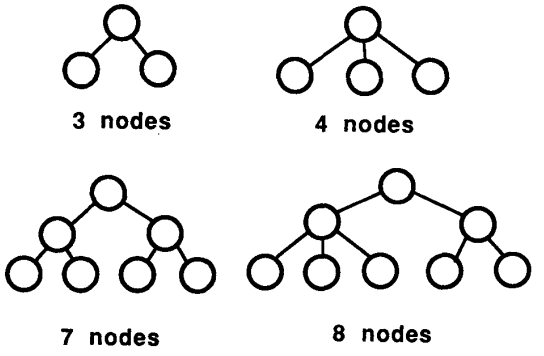

Figure 2: Four tree topologies tested.

adds). If anchor pixels P_1 and P_2 (image pixels from the original image planes surrounding pixel p being interpolated) are in local memory, no inter–process communication or synchronization is required during processing.

Subsystem topology was determined by attempts to efficiently load and unload image plane data to and from network processing nodes. Since the host can communicate directly with only one subsystem processor, a tree structure was chosen. During the first phase of this work, four trees of three, four, seven, and eight processor nodes were tested (Figure 2). Root nodes are interfaced to the host processor.

The root partitions the original image–planes and distributes data (with the help of intermediate nodes) to the leaves for interpolation. Leaves in turn pass interpolated plane segments back through intermediates to the root for re–assembly and transfer to the host. Using the seven–node topology as an example, Figure 3 demonstrates how the original eight image planes are partitioned into four columns with each leaf node receiving a column consisting of 1/4 of each of the eight planes. This insures that interpolation in each node can be done entirely in local memory.

Current support software dictated that data transfers between processor nodes consist of 16K blocks. After determining appropriate partitions, data was loaded down into the processing leaves through an efficient parallel strategy (Figure 4). This process continues until each leaf receives 8 blocks. After inter-

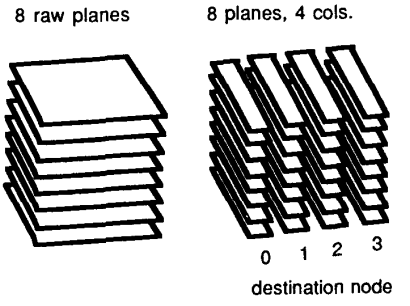

Figure 3: Data partitioning strategy.

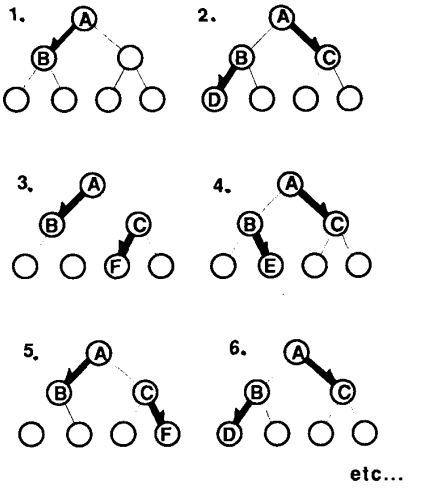

Figure 4: Parallel data transfer sequences.

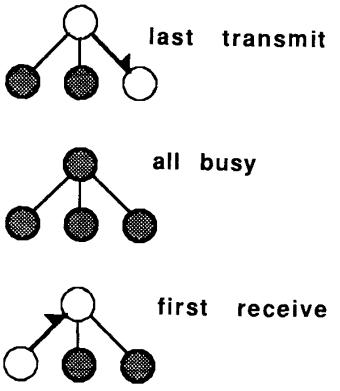

Figure 5: Root node retains a portion of the raw data.

polation, the process reverses until each leaf returns 31 blocks. After transferring a block to intermediate node A, the root begins to transfer a block to intermediate node B. During this transfer, intermediate node A transfers the first block to leaf D. After moving a block to intermediate node B, the root again begins to move a block to intermediate node A. During this transfer, intermediate node B is free to move the second block to leaf F. This process continues each leaf receives the entire 1/4 column of the original eight–image stack.

Simple linear interpolation and scaling is then performed in the leaves using binary integer operations (shifts and adds). After processing, generated image data is then routed back up the tree in reverse of the process described above. A total of 31 image planes are received by the root and combined with the bottom plane of the original data to generate the complete set of 32 images.

Since the root and intermediate nodes are idle during processing by the leaves, we also investigated a strategy using the four node topology whereby the root is also involved in interpolation (Figure 5). After the last block transfer to the leaves, all nodes are interpolating. For this strategy, the root passes an image stack representing approximately 28% of the data to each leaf node and retains a stack of approximately 16%. The root is intended in this case to be able to finish processing its 16% in time to receive interpolated data back from leaf B which was the first to be loaded and begin work.

Results

Table 1 shows execution times broken into data transfer and processing components. Transfer times consist of both root–to–leaves and leaves–to–root data movement. Transfer times are represent all time at which data blocks are being moved between nodes. Processing times represent all times during which processing is ongoing in some node. Some overlap occurs in data transfer and processing times since processing in some leaf nodes begins prior to the completion of moving data to other leaves. This overlap also occurs near the end of processing since the first

Table 1: Processing times in seconds

Topology	Data transfer	Processing	Total
1 node	0.0	19.0	19.0
3 nodes	2.5	9.8	12.3
4 nodes	2.5	6.3	8.8
4 nodes*	2.5	5.8	8.3
7 nodes	2.6	4.2	6.8
8 nodes	2.6	3.2	5.8

nodes to finish begin data transfers back to the root prior to the completion of processing in the remaining leaves. Theoretical calculations indicate this overlap time to be less than 0.06 seconds in the worst–case 8–node topology.

The times given for the one node topology are estimated based upon transputer execution speeds. The second entry for the four node topology represents the strategy where 16% of the interpolation processing occurs in the root. Total subsystem execution times (the sum of data transfer and processing times) do not include the host–to–root and root–to–host DMA transfers of the raw and interpolated image stacks. These times would of course, be constant for all node topologies and are estimated to be less than four seconds.

Discussion

As expected, data transfer times do not significantly increase with larger topologies. Even with the simplest 3 node topology, the root is 100% busy during root–to–leaves and leaves- to–root transfers. Lower level transfers (intermediate nodes to leaves and vice–versa) occur in parallel with the root activity with the exception of the final few blocks. Significant reductions in image interpolation times were achieved with even the smallest topologies. The cost effective choices would probably be the three or four node systems since only slight increases in speed are achieved with corresponding node additions.

Improvements in execution times should of course, still be possible. In all but one case, the root and intermediate nodes are still idle during interpolation processing. More effective use of these nodes could be made. In the one case where we attempted to utilize the idle root node (in the four node topology), only a slight decrease in execution time was noted. One might expect that this case should more closely approximate the seven node topology since four nodes are performing the interpolation work. The discrepancy of 1.7 sec (8.5 sec for the four node system using the root and 6.8 sec for the seven node system using only leaves) is probably the result of several factors: The choice of 16% of the raw data for the root node was conservative. If the spacing of the eight raw data planes is known a–priori, more exact proportions could be used to keep the root busy until needed for returning data transfers. In addition, we found unexpected and unexplained system overhead in data transfers with the root which are being investigated. Using measured requirements for leaf node processing and root data transfers, we estimate that efficient software which permits no idle nodes should be able to achieve 4.8 seconds with the seven node topology.

Three–dimensional displays of cardiac anatomy have the po-

tential to provide significantly improved understanding of morphology and surface details over conventional 2–D axial images or projection angiography. Quantitative measurements of volumes, positions, and sizes in arbitrary planes are possible with little added cost. Processing times required for the generation and display of such images are however, impractical in a clinical setting using low–cost AT–class computers commonly found in imaging laboratories.

This work demonstrates a relatively inexpensive (less than $10K) parallel–processing subsystem configured into a variety of topologies to significantly reduce the interpolation and scaling efforts of 3–D image generation. Future work will involve developing software to implement the inherently parallel components of surface detection and image display rendering as well. We estimate the generation and display of high resolution 3–D images from eight 256 × 256 orthogonal planes will be done in less than 1 minute using the simple 3–node topology. Dynamic or animated displays of the cardiac cycle taken from Ultrafast Cine CT systems (10 frames/sec, 8 planes/frame) could be available in less than 10 minutes which would be clinically acceptable in most laboratories. These dynamic displays promise to be a valuable tool in accurately understanding regional wall motion abnormalities in their true 3–D form.

References

1. Udupa, J. K. 3D imaging in medicine. *Proceedings, Ninth Annual Convention and Exposition, National Computer Graphics Association*, pp. 73–104. Philadelphia, PA, Volume II. 1987.

2. Cook, L. T., Dwyer, S. J. III, Batnitzky, S., and Lee, K. R. A three–dimensional display system for diagnostic imaging application. *IEEE Computer Graphics and Applications* 3:13–19 (August) 1983.

3. Barrett, W. A. Dynamic three–dimensional shaded surface display of the rotating, beating left ventricle, atrium and aorta from Cine CT. *Computers in Cardiology*, pp. 491–494. Boston, MA: IEEE Computer Society. 1986.