Cone Beam Tomography using MPI on Heterogeneous Workstation Clusters

David A. Reimann[1,2], Vipin Chaudhary[3], Michael J. Flynn[1], and Ishwar K. Sethi[2]
[1] Department of Diagnostic Radiology, Henry Ford Health System
[2] Department of Computer Science, Wayne State University
[3] Department of Electrical and Computer Engineering, Wayne State University
Detroit, Michigan 48202

Abstract

With cone beam CT long computation and a large amount of memory is required. The majority of the computation is in the backprojection step. We have investigated the use of parallel methods on workstation clusters using MPI to overcome both the long processing time and lessen memory requirements on individual workstations. We used the asynchronous MPI implementation to reconstruct a 256^3 *volume from* 256^2 *projection views with 6 workstations. Our previous efforts in load balancing resulted in processor utilization of 81.8%. Use of asynchronous communication in the cone beam CT problem has improved processor utilization to 91.9%.*

1. Introduction

Tomographic reconstruction from projections using computed tomography (CT) provides a noninvasive measure of structure from external measurements. The information obtained describes both internal and external shapes and material densities. This is particularly useful when one cannot make internal measurements on the object of study for a variety of reasons. These reasons might be cost, no known noninvasive technique, or no physical means to make internal measurements.

1.1. Cone Beam Tomography

Cone beam tomography is a computed tomography method which efficiently acquires projectional data through a single rotation. Using a 2D detector, a 3D volume can be reconstructed with isotropic voxel sizes without the mechanical translation required for conventional CT. Our current application area involves 3D CT with microscopic resolution [1].

Cone beam microtomography systems are limited primarily by the CCD camera transfer rate and available x-ray flux. Microtomography acquisitions require as much as 1 hour [1], whereas new diagnostic medical imaging systems can acquire cone beam whole body data as fast as 10 seconds.

With cone beam CT long computation and a large amount of memory is required. Large voxel arrays are used, typically 256^3 to 512^3. Reconstruction times of many hours have been required on conventional workstations [1]. The goal of this work is to use parallel processing to reduce the total CT computation time.

1.2. Feldkamp Algorithm

The reconstruction for the cone beam geometry has been investigated by numerous groups. The most efficient algorithm in use is the one developed by Feldkamp [2]. In this algorithm, the projectional data is projected back onto an image buffer with each detected ray backprojected in it's direction. Pixels in the image buffer are incremented by the amount of the projection pixel. The projection must be filtered prior to backprojection to get constructive and destructive cancelling of signals in an appropriate manner.

The coordinates used in the discussion are shown in Figure 1. It is assumed that the projection of the object at angle θ $P_\theta()$ is indexed by detector coordinates u and v. The reconstructed voxel values μ are indexed by physical coordinates x, y, and z. The center of rotation is the z axis. The distance from the x-ray focal spot to the rotation axis is d_s. By scaling the projection pixel sizes, the vertical axis of the detector can be moved to the z axis. By subsequently scaling the geometry, the pixel sizes at the z axis can

0-8186-7533-0/96 $05.00 © 1996 IEEE

be made 1. These scalings simplify the computations in the reconstruction algorithm.

The Feldkamp algorithm falls into the class of filtered backprojection algorithms. The Feldkamp algorithm can be summarized as follows:

A. Weight projection

$$P'_\theta(u,v) = \frac{d_s}{\sqrt{d_s^2 + v^2 + u^2}} P_\theta(u,v) \quad (1)$$

B. Convolve weighted projection

$$P^*_\theta(u,v) = P'_\theta(u,v) * h(u) \quad (2)$$

C. Backproject the convolved weighted projection

$$\mu(x,y,z) = \int_0^{2\pi} \frac{d_s^2}{(d_s - y)^2} P^*_\theta \left(\frac{d_s x}{d_s - y}, \frac{d_s z}{d_s - y} \right) d\theta \quad (3)$$

where it is assumed that a modified geometry with the detector containing the z axis is used ($d_d = d_s$).

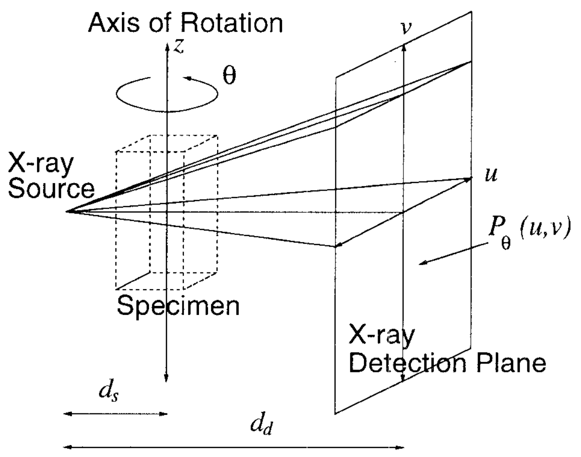


Figure 1: Cone Beam Space Variables. The variables associated with the cone beam geometry are shown and described in the text.

For the discretized problem, let the reconstruction volume μ, be $N_x \times N_y \times N_z$ voxels, in the x, y, and z directions. Let $d\theta = 2\pi/N_\theta$ and $P_\theta(u,v)$ be $N_u \times N_v$ pixels. The complexity of the problem is roughly linear with the total number of voxels and linear with the number of projection view angles. Doubling the number of voxels roughly doubles the processing time. Doubling the dimension in each direction produces an 8 fold increase in the required processing. Doubling the number of views doubles the processing. As the number of voxels increases, the number of angular views must also increase to maintain the same peripheral resolution. This is an important factor in reconstructing larger specimens or reconstructing with better resolution. For the case of $N = N_x = N_y = N_z = N_u = N_v$, the number of projections N_θ should be $\frac{\pi}{2}N$ [3], and thus the complexity of the problem is $O(N^4)$.

The majority of the computation is in the backprojection, Eq. (3). We have investigated the use of parallel methods on workstation clusters using MPI to overcome both the long processing time and lessen memory requirements on individual workstations.

1.3. Parallel Algorithms for Tomography

The large computation time for reconstruction algorithms is the major factor limiting reconstruction size. The earliest CT systems were limited to 2D images with 80×80 pixels primarily because of the computation time for the reconstruction. Clinical x-ray CT scanners today can still acquire data much faster than they can reconstruct images. For cone beam CT, our previous results indicate reconstruction times of 12 hours for a $512 \times 512 \times 128$ volume [1].

In filtered backprojection, the majority of the computation is in the backprojection step, where 98% of the operations are involved in backprojection when reconstructing a 512^3 volume from 800 512^2 projections. This suggests effort invested in improving backprojection time will be most fruitful.

Several types of data independence exist in the problem. Four forms of parallelism in 2D CT were defined by Nowinski [4]: pixel parallelism, projection parallelism, ray parallelism, and operation parallelism. Pixel parallelism uses the fact that all pixels are independent of others. Projection parallelism uses independence among projections. Ray parallelism notes that rays can be backprojected independently. Operation parallelism can be used when low level operations such as multiplications and additions can be computed in parallel. The filtering and backprojection steps can also be performed independently and the filtered projection can be pipelined to the backprojector [5]. However, the large data sizes warrant careful algorithm selection to minimize communications and RAM requirements.

Dedicated hardware implementations have been proposed [6, 7, 8, 9, 10, 11, 12, 13, 14, 15]. A dedicated hardware chip for backprojection has been recently developed which can backproject a 512×512 image in 4 seconds [8]. Many of these chips can be used in parallel to further reduce reconstruction times. It is doubtful if any of these systems are currently in use, either because of newer hardware solutions, or the hardware is not yet fully integrated. Furthermore, none of these systems was developed for cone beam tomography. Another drawback to a dedicated hardware system is the relatively low market demand and the lack of scalability of such a system.

Another active area of research has been in the area of employing commercially available parallel systems to the reconstruction problem. Several investigators have evaluated the Transputer [16, 17, 18, 19, 20]. The MasPar has been used [21, 22, 23]. One investigator used a Cray-1 [24]. The iPSC/2 has been used [5, 25, 26, 27]. Still others have used other specialized multiprocessors [28, 29, 30, 31]. Parallel efficiencies of greater than 88% have been reported [5]. These systems require a large investment ($50,000 - $300,00) compared with general purpose workstations.

However, none of this prior literature dealt specifically with large volume tomographic reconstruction from cone beam projections. A better understanding of the interplay between communications, processing power, and memory should be useful in predicting performance on dedicated hardware implementations and commercially available parallel systems.

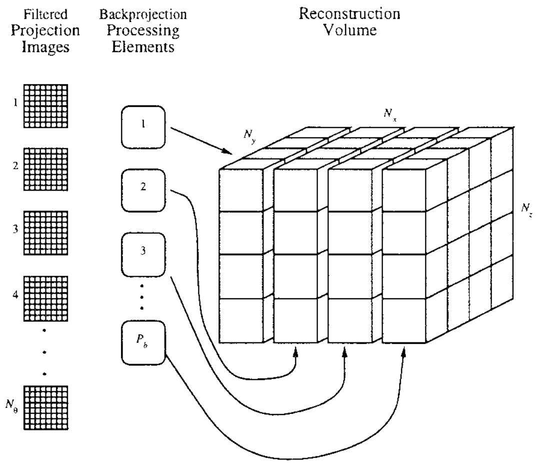

Filtered Projection Images
Backprojection Processing Elements
Reconstruction Volume
1
2
3
4
N_θ
1
2
3
P_b
N_x
N_y
N_z

Figure 2: Voxel Driven Partitioning. The voxel driven data partitioning associated with the cone beam reconstruction problem is shown. Each processing element reconstructs a disjoint slab of the total reconstruction volume. The slab width varies proportionally to processing speed. The Methods section describes the asynchronous communication of the filtered projection images to the backprojection PE's.

1.4. Voxel Driven Algorithm

The forms of Nowinski [4] can be extended to the three dimensional problem as follows. Pixel parallelism can be extended to voxel parallelism. Projection and ray parallelism remain the same, with the exception of increases in the size of the projections and the number of rays. The voxel/pixel driven approaches arise from the order the mathematical integration of the backprojection, with the angle θ varying in the outermost loop, and x, y, and z inside the θ loop. A projection image can be independently weighted, filtered, and backprojected into all voxels.

A voxel driven approach is taken where the volume is distributed over several processor elements (PE's) and each view is sent to the voxel PE's as shown in Figure 2. Each PE sees every projection, but only a small subset of the reconstructed voxels. In this discussion the processor elements are individual workstations, but would correspond to each CPU on SMP workstations. The total memory required for this implementation is approximately equal to the total number of voxels. One advantage of this method is that the data is acquired in a serial fashion and processing could be done in concert with acquisition. This type of parallelism was found to be superior in [16].

The problem has a high degree of independent operations, and should be well suited to parallel implementations. The major nontrivial aspects of the problem is the data sizes needed. An acquisition can be as large as 800 views of 512^2 16 bit integer images. These projectional images are used to create a volumetric data set on the order of $512 \times 512 \times 512$ 32 bit floating point voxels. The use of 32 bit floating point voxels is required to provide sufficient accuracy in the result. The memory requirement to store the entire $512\times512\times512$ 32 bit volume is 512 MB, which is at the upper limit of available memory on most tightly coupled parallel systems currently available. However, dividing this memory among 16 workstations requires only 32 MB per workstation.

The Feldkamp algorithm was implemented using the freely available MPICH [32] implementation of MPI. In addition to point-to-point send and receives, the reconstruction used asynchronous receives, collective broadcasts, derived data types, and synchronization barriers. The message passing algorithm for the voxel drive approach is pseudocoded as:

```
1.1  Initialize each PE
1.2    Read and Broadcast problem specifications
1.3    Partition memory
1.4    Allocate memory
1.5    Precomputation of weights
1.6  for each $\theta$ ($N_\theta$ views):
1.7    if (PE is ROOT)
1.8      Read Projection
1.9      Weight and Filter Projection
1.10   Broadcast Projection $P^*_\theta(u,v)$
1.11   Backproject Projection
1.12 Gather Reconstructed Voxels on ROOT PE
```

The important features of MPI used in the implementation are now discussed. MPI_Bcast was used to broadcast initial information from the root node, including volume allocation information. MPI_Bcast was used to send each filtered projection (32 bit pixels) to the backprojector processes. MPI_Send was used to send each completely reconstructed volume to the root node. MPI_Irecv was used to asynchronously send each completely reconstructed volume to the root node. The gathering of the reconstructed volume of voxels and filtering of the projections is all done on the initial processor.

2. Methods

Because of the large amounts of data involved in this problem, it was our interest to use an explicit message passing library for the implementation to achieve maximum performance. Our earlier results showed 81.8% processor utilization on a heterogeneous cluster of 6 Sun workstations [33]. This high degree of utilization was achieved by load balancing in the backprojection and in the filtering steps. In this work, we demonstrate improved performance using asynchronous communications to broadcast the projection.

2.1. Cluster of Workstations

A parallel implementation on a workstation cluster has several positive implementation aspects. Workstation clusters are common in industry, research, and universities and have been found useful in solving large problems [34, 35]. They are more cost effective than supercomputers both in terms of initial cost and maintenance [34]. The availability of networked systems is generally high throughout the day and the availability and throughput on such a system can be accurately predicted [36]. Since reconstruction is very deterministic, it is therefore a very good candidate for job scheduling. Some hardware aspects of workstation clusters are better suited for the large tomography problem than dedicated parallel processors, such as higher cache/RAM and performance/cost ratios. It is interesting that a proposal was made 10 years ago for using off the shelf computers with inexpensive memory for use in the reconstruction problem [37].

2.2. MPI Implementation

The reconstruction problem is very suitable for asynchronous communication because of the repeated deterministic nature of the views in the problem. By using asynchronous communication to send projectional data while backprojecting the previous projection. the communications overhead can be eliminated on the receiving processors as long as the backprojection time is greater than or equal to the communication time.

We replaced the synchronous broadcast (MPI_Bcast) with repeated asynchronous send/receive pairs (MPI_Isend/MPI_Irecv) and with double buffering of the projection to assure correctness. For the large projection sizes, the MPI_Isend became non-asynchronous. To achieve asynchronous communication, a repeated MPI_Test during backprojection on the receiving processors was needed to complete the send. The new pseudocode is a follows:

```
2.1  Initialize each PE
2.2    Read and Broadcast problem specifications
2.3    Partition memory
2.4    Allocate memory
2.5    Precomputation of weights
2.6  $\theta = 0$
2.7  if (PE is ROOT)
2.8    Read Projection
2.9    Weight and Filter Projection
2.10   Send Projection (MPI_Isend)
2.11 else
2.12   Receive Projection (MPI_Irecv)
2.13 for each $\theta > 0$ ($N_\theta$ views):
2.14   if (PE is ROOT)
2.15     Read Projection
2.16     Weight and Filter Projection
2.17     Send Projection (MPI_Isend)
2.18   else
2.19     Receive Projection (MPI_Irecv)
2.20   Backproject Projection
2.21     (Loop over all voxels in slab
2.22     with periodic (MPI_Test)
2.23 Gather Reconstructed Voxels on ROOT PE
```

Information on the workstations used in the following experiments are detailed in Table 1. The backprojection speeds and projection filtering time was determined *a priori* and stored in a file. The backprojection speed was used to define the voxel memory assigned to each workstation in such a way to balance the load among the processors.

3. Results

We used the asynchronous MPI implementation to reconstruct a 256^3 volume from 10 256^2 projection views. While the number of views is significantly less than what would be typically used, the computation of several view allows a good understanding of what

Table 1

Workstation Name	Operating System	RAM (MB)	Processor Architecture	Backprojection Speed (1000 Voxels/Second)
mulberry	SunOS 4	80	Sun 4/330	168
tomons	Solaris	64	Sun 4/470	160
ludy	SunOS 4	32	Sun 4/330	156
entropy	SunOS 4	160	Sun 4/490	222
grady	SunOS 4	64	SPARC 2	308
toshiba	SunOS 4	64	SPARC 2	311

is required in actual problems due to the its deterministic and repetitive nature. 32 bit floating point data types were used to for both voxels and filtered projection pixels. For this problem, the total amount of voxel memory was 64 MB, and each projection was .25 MB in size. On a 10 Mb ethernet, one would expect no less than .2 seconds of transfer time per projection per workstation used.

Because of the varying processor speeds used, processor utilization rather than speedup was used to evaluate the quality of the parallel implementation. The utilization was computed as follows. The total time for filtering F was measured using a only a single PE. For the ith PE, the elapsed time T_i for runs using multiple processors as well as individual backprojection time B_i was then measured. Utilization U, was then computed as

$$U = \frac{F + \sum_{i=1}^{P_b} B_i}{\sum_{i=1}^{P_b} T_i} \qquad (4)$$

where P_b is the number of processors used.

The reconstructions were repeated using only 6 workstations and 100 projection views. To demonstrate improved utilization, the execution timelines for this example are shown in Figure 4. Use of asynchronous broadcast in the cone beam CT problem has improved utilization from 81.8% to 91.9% for this case.

4. Discussion

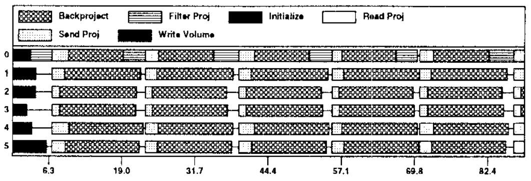
Asynchronous communication was very useful for this problem. However, the large message sizes used in this problem caused unexpected synchronous behavior from asynchronous communication calls. Synchronous behavior was restored by completing the receive with `MPI_Test`.

Utilization as a function of the number of workstations shows the point where the communication time dominates. This correlates well with the speed of the ethernet. One would expect to send a 256^2 projection (.25 MB) in around .2 second. With this communication time and the total number of voxels per second which can be backprojected, an estimate of the number of processors which will optimally solve the problem can be computed. This simple model does not account for message overhead, and further effort is needed for a more complete model addressing the interplay between workstation performances and communication speeds in a heterogeneous environment.

In addition to asynchronous communication, another possibility is a multiphase broadcast. Since many networks now isolate traffic into physical subnetworks with bridges and routers a broadcast method using forwarding can be employed. In this method, the master processes broadcast to a head slave in each subnet which then forwards to each workstation on the subnet. This can be easily implemented using communicator groups. A disadvantage of this method requires apriori knowledge of the network topology. However, this topology is likely to change slowly and could be maintained in a configuration file with processor speed and availability. A multiphase broadcast could be done both synchronously and asynchronously.

The use of an ethernet network has advantages in broadcasting a filtered projection to many backprojectors simultaneously. Many of the previous implementations were hampered by communications speed. However, none took advantage of the broadcast mechanism of a bus architecture, such as in ethernet. Use of a broadcast bus for transmission of the projectional data could significantly reduce the communications time when many workstations are used.

The issues involving large memory and large amounts of communication are inherent to other problems involving discretized representations of physical space. Examples include volumetric imaging applications (rendering, processing, etc.) and finite element analysis of 3D structures. MPI seems a good parallel platform on which to build these types of applications both from the ease of implementation and portability.



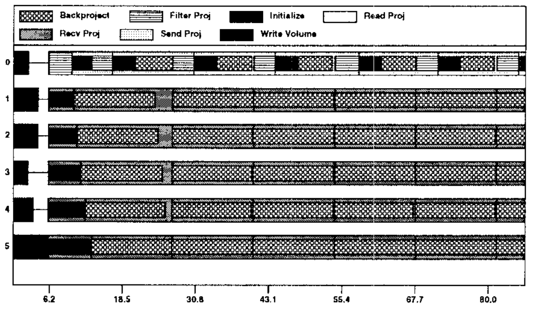


Figure 3: Execution Timelines. Execution timelines for 6 workstations with broadcast (top) and encorporating asynchronous communication (bottom) is shown. Backprojection into a 256^3 volume from 256^2 projections is performed for shown for the first 5 views.

5. Acknowledgements

Partial support of this work was provided by NIH Grant RO1-AR42101.

References

[1] D. A. Reimann, M. J. Flynn, and S. M. Hames, A flexible laboratory system for 3D x-ray microtomography of 3–50 mm specimens, in *3D Microscopy: Image Acquisition and Processing 1995*, Proceedings of the SPIE 2412, pages 186–195, San Jose, California, 1995.

[2] L. A. Feldkamp, L. C. Davis, and J. W. Kress, Journal of the Optical Society of America A **1**, 612 (1984).

[3] A. C. Kak and M. Slaney, *Principles of Computerized Tomographic Imaging*, IEEE Press, New York, 1988.

[4] W. L. Nowiński, Parallel implementation of the convolution method in image reconstruction, in *CONPAR 90 — VAPP IV*, edited by H. Burkhardt, volume 457 of *Lecture Notes in Computer Science*, pages 355–364, Springer-Verlag, 1990.

[5] C. Chen, S.-Y. Lee, and Z. Cho, IEEE Transactions on Nuclear Science **37**, 1333 (1990).

[6] B. Gilbert, R. Robb, and L. Krueger, Ultra high-speed reconstruction processors for x-ray computed tomography of the heart and circulation, in *Real-Time Medical Image Processing*, edited by M. Onoe, K. Preston, Jr., and A. Rosenfeld, pages 23–40, Plenum Press, New York, 1980.

[7] E. Di Sciascio, R. Guzzardi, and D. Marino, Proposal of a real-time reconstruction processor for 3D positron emission tomography, in *Conference Record of the 1992 Nuclear Science Symposium and Medical Imaging Conference*, volume 2, pages 921–923, Orlando, Florida USA, 1992.

[8] I. Agi, P. J. Hurst, and K. W. Current, IEEE Journal of Solid-State Circuits **28**, 210 (1993).

[9] S. Ruhman and I. Scherson, Associative processor for tomographic image reconstruction, in *MEDCOMP '82 Proceedings*, pages 353–358, Philadelphia, Pennsylvania USA, 1982, IEEE.

[10] H. Wani and H. Ishihara, Real-time image processing in CT—convolver and back projector, in *Real-Time Medical Image Processing*, edited by M. Onoe, K. Preston, Jr., and A. Rosenfeld, pages 99–106, Plenum Press, New York, 1980.

[11] S. L. Wood, Efficient MVE image reconstruction for arbitrary measurement geometries, in *ICASSP 82 Proceedings*, volume 2, pages 1158–1161, Paris, France, 1982, IEEE.

[12] N. Wilkinson, M. Atkins, and J. Rogers, IEEE Transactions on Nuclear Science **36**, 1047 (1989).

[13] W. Jones, L. Byars, and M. Casey, IEEE Transactions on Nuclear Science **37**, 800 (1990).

[14] C. Thompson and T. Peters, IEEE Transactions on Nuclear Science **28**, 3648 (1981).

[15] E. Sheih, K. W. Current, P. J. Hurst, and I. Agi, IEEE Transactions on Circuits and Systems for Video Technology **2**, 347 (1992).

[16] M. S. Atkins, D. Murray, and R. Harrop, IEEE Transactions on Medical Imaging **10**, 276 (1991).

[17] N. Kingswood et al., IEE Proceedings **133, Part** E, 139 (1986).

[18] S. Barresi, D. Bollini, and A. Del Guerra, IEEE Transactions on Nuclear Science **37**, 812 (1990).

[19] C. Comtat, C. Morel, M. Defrise, and D. Townsend, Physics in Medicine and Biology **38**, 929 (1993).

[20] K. Girodias, H. Barrett, and R. Shoemaker, Physics in Medicine and Biology **36**, 921 (1991).

[21] C. Butler and M. Miller, IEEE Transactions on Medical Imaging **12**, 84 (1993).

[22] C. Butler, M. Miller, T. Miller, and J. Wallis, Physics in Medicine and Biology **39**, 575 (1994).

[23] M. Miller and C. Butler, IEEE Transactions on Medical Imaging **12**, 560 (1993).

[24] L. Kaufman, IEEE Transactions on Medical Imaging **6**, 37 (1987).

[25] C. Chen, S.-Y. Lee, and Z. Cho, IEEE Transactions on Medical Imaging **10**, 513 (1991).

[26] C.-M. Chen and S.-Y. Lee, IEEE Transactions on Parallel and Distributed Systems **5**, 860 (1994).

[27] H.-P. Charles, J.-J. Li, and S. Miguet, 3D image processing on distributed memory parallel computers, in *Biomedical Image Processing and Biomedical Visualization*, edited by R. S. Acharya and D. B. Goldgof, Proceedings of the SPIE 1905, pages 379–390, San Jose, California USA, 1993.

[28] M. S. Atkins, M. Zastre, K. Buckley, and L. Byars, Evaluation of the use of the i860 supercard in a 3-D PET tomograph, in *Conference Record of the 1992 IEEE Nuclear Science Symposium and Medical Imaging Conference*, volume 2, pages 913–914, Orlando, Florida USA, 1992.

[29] T. Guerrero, S. Cherry, M. Dahlbom, A. Ricci, and E. Hoffman, Fast implementations of 3D PET reconstruction using vector and parallel programming techniques, in *Conference Record of the 1992 IEEE Nuclear Science Symposium and Medical Imaging Conference*, volume 2, pages 969–971, Orlando, Florida USA, 1992.

[30] T. Guerrero, A. Ricci, M. Dahlbom, S. Cherry, and E. Hoffman, Parallel image reconstruction for 3D positron emission tomography from incomplete 2D projection data, in *Biomedical Image Processing and Biomedical Visualization*, edited by R. S. Acharya and D. B. Goldgof, Proceedings of the SPIE 1905, pages 978–986, San Jose, California USA, 1993.

[31] M. I. Miller and B. Roysam, Proc. Natl. Acad. Sci USA **88**, 3223 (1991).

[32] Argonne National Lab and Mississippi State University, MPICH, http://info.mcs.anl.gov/pub/mpi.

[33] D. A. Reimann, V. Chaudhary, M. J. Flynn, and I. K. Sethi, Parallel implementation of cone beam tomography, in *1996 International Conference on Parallel Processing*, Bloomingdale, Illinois, 1996, Accepted for presentation (Paper 2050).

[34] B. Buzbee, Science **261**, 852 (1993).

[35] R. L. Martino, C. A. Johnson, E. B. Suh, B. L. Trus, and T. K. Yap, Science **265**, 902 (1994).

[36] M. W. Mutka, IEEE Transactions on Software Engineering **18**, 319 (1992).

[37] J. Llacer and J. D. Meng, IEEE Transactions on Nuclear Science **32**, 855 (1985).