1998

Parallel Algorithms for 3D Reconstruction of Asymmetric Objects from Electron Micrographs

Robert E. Lynch
Purdue University, rel@cs.purdue.edu

Dan C. Marinescu

Hong Lin

Timothy S. Baker

Report Number:
98-039

PARALLEL ALGORITHMS FOR 3D RECONSTRUCTION OF ASYMMETRIC OBJECTS FROM ELECTRON MICROGRAPHS

Robert E. Lynch
Dan C. Marinescu
Hong Lin

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR #98-039
November 1998

Parallel Algorithms for 3D Reconstruction of Asymmetric Objects from Electron Micrographs

Robert E. Lynch, Dan C. Marinescu, Hong Lin
(rel, dcm, linh@cs.purdue.edu)
Computer Sciences Department
and
Timothy S. Baker (tsb@bio.purdue.edu)
Biological Sciences Department
Purdue University
West Lafayette, IN, 47907

Abstract:

We present new parallel algorithms for 3D reconstruction of objects from 2D projections and their application for the determination of the structure of viruses from electron micrographs. The multi-resolution orientation determination algorithm uses a parallel search to determine the "best fit" of a given image with images in a reference database. The 3D reconstruction algorithm uses Cartesian coordinates and permits the reconstruction of objects that do not posses symmetries. Our method decomposes a large problem into a number of smaller problems that can be solved independently on different processors of a parallel computer or on a cluster of workstations. The paper outlines the data partitioning and load balancing issues pertinent to the parallel implementation of the algorithm and presents preliminary results obtained on a SGI Origin 2000 system.

Key words: parallel processing, structural biology, electron microscopy, scientific computing, 3D -reconstruction.

Contents:

1. Introduction

There are several practical methods to reconstruct a 3D object from a set of its 2D projections. We use a Fourier transform method [Crow70] implemented for parallel processing (see Section 4). Other methods include 'back projection' [Crow70] and numerical inversion of the Radon transform [She80]. For descriptions of (sequential) methods for 3D reconstruction and many necessary related tasks, see [Dea93] or [Fra96], two of several books containing clear explanations and many references.

We are concerned with the 3D reconstruction of virus particles. The experimental observations come from cryo-electron micrographs and corresponding projections of many identical particles. These projections allow us to reconstruct the 3-D electron density of the virus. Essential features include the preparation of the specimens, isolation of particular projections, the determination of the orientation of each projection, and so on. After an iterative process that involves improvement of the orientation and the reconstruction, an atomic model must be constructed which fits the reconstructed electron density.

3D reconstruction is a computationally and data intensive problem. The 3D reconstruction in the Fourier domain requires solution of a linear least squares problem with M^3 unknowns. The size of the grid, M, can be as large as 1,000 and routinely has values around 500. This justifies the interest in parallel methods for 3D reconstruction.

The paper is organized as follows. Section 2 outlines the steps for virus particle reconstruction from micrographs. Section 3 describes the parallel algorithm for determination of the orientation of particle projections in a micrograph. Section 4 presents the parallel algorithm for 3D reconstruction. Section 5 outlines implementational issues. Section 6 presents preliminary performance data.

2. Virus Particle Reconstruction in Cryo-Electron Microscopy

The atomic structure determination of macromolecules based upon electron microscopy is an important application of 3D reconstruction. Electron microscopy, X-ray crystallography, and NMR, Nuclear Magnetic Resonance, are the methods currently used for gathering experimental information about the 3D atomic structure of biological macromolecules. NMR methods are used for relatively small macromolecules. X-ray crystallography is capable of providing high-resolution electron density maps of large macromolecules, like viruses, and electron microscopy provides medium to high-resolution maps. A small macromolecule may consist of thousands of residues while a large one may consist of hundreds of thousands of amino acids or millions of atoms. X-ray crystallography allows construction of electron density maps, to 2-2.5Å, while traditionally, electron microscopy produced lower resolution maps to say 20-30Å. More recently, researchers at Cambridge and NIH were able to produce maps to 7-7.5Å resolution and there is hope that this limit can be further pushed by increasing the number of projections used for reconstruction, from the current level of few hundreds, to thousands or tens of thousands.

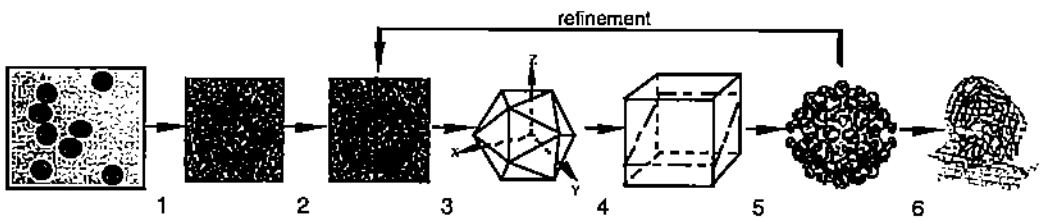
The procedure for structure determination is illustrated in Figure 1. This figure also indicates the potential impact of the algorithms and methods we are developing upon the time frame for structure determination. The algorithms described in this paper are part of

an ambitious effort to design an environment for "real-time electron microscopy", where results can be obtained in hours/days rather than weeks/months.

The first step of the process, identification of particle projection and our algorithms for automatic identification are discussed elsewhere [Mar97]. Once particle projections are identified and extracted from micrographs (Step 1), the center (Step 2) and orientation (Step 3) of each particle projection is determined. Then each projection is Fourier transformed to reciprocal space and the 3D reconstruction is computed by interpolation in the Fourier domain (Step 4) using the information about the orientation of each individual projection. Finally, an inverse Fourier transformation yields an electron density map (Step 5).

This map can then be further improved by various refinement procedures and the inclusion of more particle images and of higher resolution information, if available. Refinement involves using an intermediate 3D reconstruction as a model to define better the center (x,y) and orientation parameters for all particle images, followed by Steps 4 and 5 and additional cycles as needed to include more particle images and extend resolution.

Not depicted here are the steps involving specimen preparation, obtaining the micrographs, and digitization of the images (all of which precede Step 1), which can take as few as several hours to complete for ideal samples, or, more typically, days or weeks for difficult specimens. Digitization may be performed at the microscope by recording images on a slow-scan CCD camera or by scanning images recorded on photographic film with a microdensitometer. The time frame increases approximately linearly with the number of cycles of refinement (Steps 2-5). Typically, four or less cycles are sufficient for processing "good" data at 20Å resolution. Model building, the process of fitting atomic level models into electron density maps, is achieved using atomic level information gathered from protein data banks (Step 6). Approximate time frames for each step in the process are shown for the analysis of either 100 or 2000 particle images, for images in the 150^2 to 300^2 pixels size range.



		1	2	3	4	5	6
Current	n= 100	1-4 hrs	mins	hrs	mins	mins	hrs/weeks
	n=2000	days	mins/hrs	days	mins/hrs	hrs/day	hrs/weeks
Goal	n= 100	secs	~1 sec	secs	< 10 secs	< 10 secs	secs/hrs
	n=2000	mins	< 1 min	mins	< 1 min	< 10 mins	mins/hrs

Figure 1: Schematic representation of the processing 'pipeline' from 2D images to 3D modeling. Step 1: extracting individual particle images from electron micrographs or CCD images. Step 2: determining initial locations of all particle centers. Step 3: determine view orientations for all particles. Step 4: fill in 3D transform. Step 5: compute 3D reconstruction (shown is rhinovirus-Fab complex, with Fabs colored blue). Step 6: dock atomic model into cryoTEM 3D density map (example shows an atomic antibody Fab model docked into the corresponding Fab portion of the virus-Fab 3D reconstruction).

For an idea of the computational effort required by the problems addressed in this paper, consider the 3D reconstruction step in the structure determination process (Fig.1, Steps 4 and 5). Assuming there are 2,000 particle projections each of size 300 x 300 pixels, we need to solve about $3 \times 300^2 = 270{,}000$ linear least squares problems each having 2000 equations and 300 complex unknowns. As explained in Section 4, the number of operations required is of the order of $270{,}000 \times 2000 \times 300^2 = 5 \times 10^{13}$. If the particle has known symmetry, then these values can be reduced. But, much of the computation can be done independently from other parts; specifically, each of the linearly systems can be solved independently of the others. This provides ample justification for concurrent computations. Yet, to use efficiently a parallel computer or a cluster of workstations, we need parallel algorithms which partition the computation uniformly, which minimize the communication among processors, and at the same time maintain a high level of locality of reference. This must be done not only for the 3D reconstruction, but also for the determination of the orientation. Similar efforts have been reported in the past [Joh94], but the performance data available to us suggest that new algorithms have to be designed to reduce dramatically the computation time.

3. Concurrent Computations to Determine Particle Orientations

After isolating the particle projections (Fig. 1 Step 2), the next step in determining the 3D structures of biological molecules from transmission electron micrographs is to identify particle orientations (Fig.1, Step 3). Determination of the view orientation for individual, particles from noisy, low dose electron micrographs is problematic for particles with no internal symmetry, such as the ribosome, or with high internal symmetry such as icosahedral viruses. The Common Lines Method developed by Crowther [Crow70; Crow71], and the modified version of it [Ful96] are among the best known approaches. A group from NIH has developed parallel programs that implement the common lines techniques [Joh94].

We are developing parallel algorithms and programs based upon the Polar Fourier Transform, PFT, method [Bak96]. A major advantage of the algorithm is that raw image data is compared to a relatively noise-free 3D model, rather than other noisy data, as is the case of the Common Lines techniques. In addition, the PFT method makes use of all the available data, whereas in the Common Lines only a fraction of the Fourier data is sampled. In our experience, these features of the PFT technique have led to more consistent and reliable results in the analysis of many virus structures.

We review the basic idea of the sequential algorithm that is implemented in an iterative scheme. At each iteration we start with a model of the virus particle obtained by 3D reconstruction during the previous iteration. Knowing the electron density at each grid point of a 3D lattice we are able to construct a Reference Data Base (RDB) consisting of m projection images, with m dependent upon the resolution as discussed below. We have an image ('raw') data set consisting of n images whose orientations we want to determine. These raw images are obtained by isolating individual virus projections in micrographs. Each raw image from the data set is compared with all projection images in the RDB to determine a "best fit". The unknown particle orientation is determined by the orientation corresponding to the "best fit" image in the RDB. The time to determine the orientations of all the particles in the data set at a given resolution

is $T_0 = F \times n \times m$, with F being the time to evaluate the "fitness" of a pair (one raw image against one RDB projection).

The algorithm we are designing includes several enhancements discussed separately. The speedup, S, defined as the ratio of the execution time *without* the enhancement and *with* the enhancement is estimated for each case.

Image Compression. The time to evaluate the "fitness" of a pair of images and the space required to store an image are a function of the image size. Both can be reduced if we use compressed images instead of original ones. If C is the compression factor, the speedup due to compression, S_C, is: $S_C = C$.

Multi-phase, multi-resolution search. Instead of constructing a high-resolution RDB and looking for the best match, we use a two-phase scheme. In Phase 1, a "low-resolution RDB" is constructed, consisting of m1 images and the search is performed over a large search window (a half asymmetric unit of 3D space for particles with 532 symmetry). In Phase 2, a "high-resolution RDB" is formed with *m2* images, where $m1 \ll m2$. We then limit the search to a small region consisting of $(r \times m2)$ images, where r represents the fraction of the search window used in Phase 1 $(r \ll 1)$, and hence represents a region restricted around the area of the "best fit" found in Phase 1. The speedup due to multi-phase, multi-resolution search is:

$$S_M = (F \times n \times m_1)/(F \times n\,(m_1 + r \times m_2)) \times 1/r.$$

Parallel search. If we have P processors, then each one can process a fraction, n/P, of the n images concurrently with the other processors. The speedup due to parallel processing is $S_P = P$.

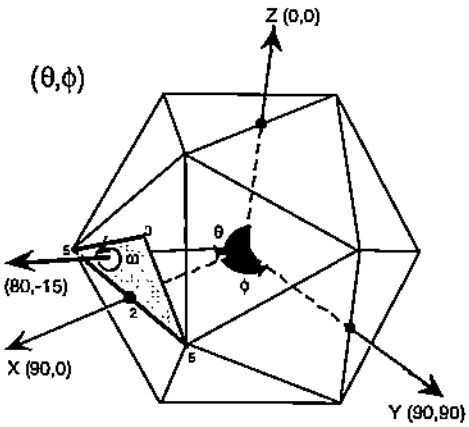
All the improvements discussed above are multiplicative and the total speedup of the method described above is:

$$S = S_C \times S_M \times S_P = (C \times P)/r.$$

If, for example, $C = 4$, $P = 20$, and $r_2 = 0.01$, then $S = 8{,}000$. This ideal speedup is unlikely to be attained in practice because of additional overheads, *e.g.* for replicating the RDBs in all nodes of a parallel system.

The PFT analysis begins with a 3D model used to generate the RDB, consisting of m different projected views [Bak96]. For an icosahedron, these views cover one half of the asymmetric unit of the structure (Fig. 2). Three angles define the orientation of the view direction: θ, the rotation in the xz plane, positive from z towards x, φ, rotation in the xy plane, positive from x towards y, and ω, the rotation of the object about the (θ,φ) view direction. Each Cartesian view, centered at coordinates (x,y) is interpolated onto a polar grid (r, λ) which subdivides the data into a set of equally spaced annuli. The polar data consists of annuli organized in rows (r directions), all sampled the same number of times (λ direction). For an icosahedral particle, the RDB consists of $m = 52$ projections at 3Å angular resolution, 382 projections at 1Å resolution, and 3,943 projections at 0.3 resolution. The space needed for one 220 by 512 polar Fourier transform is about 1 MB,

hence, the space needed for the entire reference database sampled at 0.3Å intervals is about 4 GB. The method discussed in the previous section produces a data set of n raw images, with n ranging from 100 to 2000, or even more for high-resolution data analysis. For each of these n images the PFT algorithm produces a tuple of values $(\theta,\varphi,\omega,x,y)$ by searching the RDB for the "best fit".



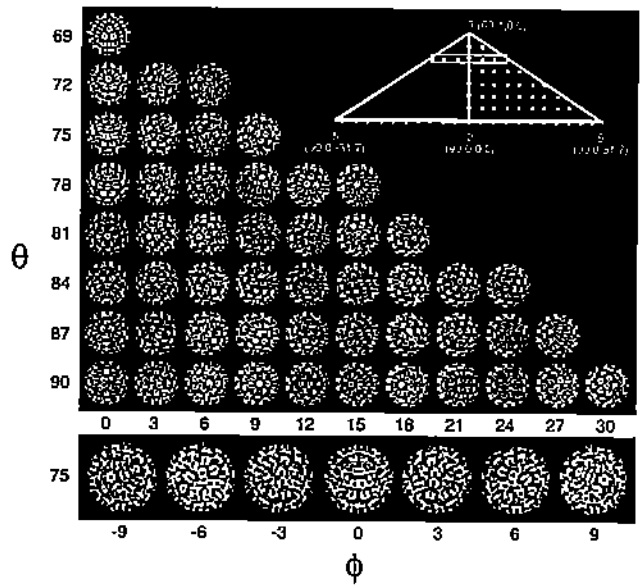


Figure 2. (Left) Representation of an icosa-hedron with three, mutually perpendicular two-fold axes aligned with an x,y,z Cartesian coordinate system. One, triangular asym-metric unit (1/60th of the icosahedron) is shaded and one representative view orien-tation (θ,• = 80,-15) is depicted. The angles θ and • define the view orientation and ω the rotation of the object about that view direction. (Right) Series of projections of a 3D model (reconstruction of bovine papilloma virus) at 3° angular intervals in θ and • demonstrate how the projected view changes with view orientation. At the bottom are enlarged views of the 3D model projections for θ = 75° and for • from –9° to +9°.

The algorithm we are developing is multi-phase, pipelined, and uses a sliding window approach, see Figure 3. In the first phase, we compute a coarse reference database (at 3Å angular increments, the database consists of 52 views) then label each of the n angular views by the region (θ,φ) of the best fit. In the second phase we use a pipelined approach: while several processors compute a finer reference data base (e.g. 3,943 projections at 0.3Å angular resolution), the other processors start the search process in a sliding window. A sliding window, means processing at one time only images in a given range of (θ,φ) values, from $(\theta_{min},\varphi_{min})$ to $(\theta_{max},\varphi_{max})$. We use a self-scheduling approach, in which all unprocessed images in the current window are inserted in an image queue and processors remove them from this queue when they become ready to process a new image. This approach improves the locality of reference. Only the images in the queue waiting to be processed and those already being processed need to be in core at a given time. Only the region of the RDB corresponding to the range of (θ,φ) values of the sliding window needs to be in core at the same time. Pipelining allows us to produce the new projections in the RDB only when the window slides, covering angular views previously labeled in that region.

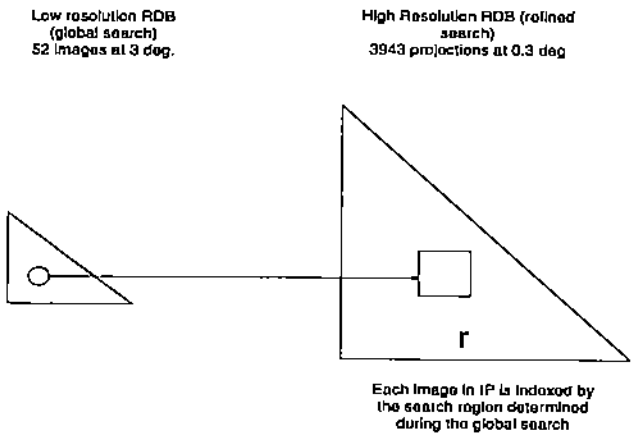


Figure 3. The orientation determination. Each image in the Image Pool is compared against all images in the RDB during the global search. During the refined search phase only a small region of size r determined by the global search results is explored.

4. Parallel 3D Reconstruction

Our method for 3D Reconstruction is modeled on one of the five methods suggested by [Crow70]; we compare it with two of their other five methods. We do not discuss their method using cylindrical coordinates nor their back projection method, both of which have produced many useful 3D reconstructions; see, for example [Dea93] and [Fra96]. A preliminary version of our method is given in [Lyn97].

We use a Cartesian coordinate system with points $\mathbf{x} = (x\ y,z\ )^T$, where T denotes transpose. The Fourier transform of the electron density, ρ, is given by

$$F(\mathbf{h}) = \int \rho(\mathbf{x}) e^{-2\pi i \mathbf{h}^T x} d\mathbf{x}$$

The observational data consist of electron micrographs of an object; each can be considered as the image of the projections of the integral of ρ along lines normal to the micrograph. We know the orientation, θ, φ, ω, of the image (Step 3, Figure 1). A 3 by 3 matrix, M, defined in terms of θ, φ, ω, rotates a point $\mathbf{x}$ to a point $\mathbf{p} = (p\ ,q\ ,r)^T = M\mathbf{x}$. M is an orthogonal matrix: its transpose, M^T, is equal to its inverse, M^{-1}.

Points $\mathbf{h}$ in the transform space are rotated in the same way as the points $\mathbf{x}$. This can be seen as follows. Because M is orthogonal,

$$\mathbf{h}^T\mathbf{x} = \mathbf{h}^T M^{-1} M\mathbf{x} = \mathbf{h}^T M^T M\mathbf{x} = (M\mathbf{h})^T M\mathbf{x} = \mathbf{u}^T\mathbf{p}$$

where $\mathbf{h} \rightarrow \mathbf{u} = (u\ ,v\ ,w)^T = M\mathbf{h}$ and $\mathbf{x} \rightarrow \mathbf{p} = (p\ ,q\ ,r)^T = M\mathbf{x}$. Because the modulus of the determinant of M is unity, $d\mathbf{x} = d\mathbf{p}$, and by setting $\rho(\mathbf{x}) = \rho(M^{-1}\mathbf{p}) = R(\mathbf{p})$ we obtain

$$F(\mathbf{h}) = \int R(\mathbf{p}) e^{-2\pi i \mathbf{u}^T \mathbf{p}} d\mathbf{p} = f(\mathbf{u})$$

where f is denotes the Fourier transform, in the rotated coordinate system, of ρ.

For coordinates of the transform $f(u,v,w)$, take u, v to be parallel to the pixel frame edges. Then the w axis is normal the plane; finally, take the origin to be in the pixel frame so that $f(u,v,0) = P(u,v)$ is the Fourier transform at (u,v) of the pixel frame. We have

$$P(u,v) = \int R(p,q,r)e^{-2\pi i(up+uq)}dpdqdr = \int \left\{ \int R(p,q,r)dr \right\} e^{-2\pi i(up+uq)}dpdq$$

Because $\mathbf{p} = \mathrm{M}^{-1}\mathbf{x}$ and $\mathbf{u} = M^{-1}\mathbf{h}$, P is the transform of the projection of the integral of the density ρ onto the plane perpendicular to the w-axis.

The particle has finite size and we can extend its density periodically and represent the density with a Fourier series:

$$\rho(x) = \sum_{\mathbf{h}} F(\mathbf{h})e^{2\pi \mathbf{h}^T x}$$

so that

$$P(u,v) = \int \sum_{\mathbf{h}} F(\mathrm{h})e^{2\pi i(\mathrm{h}-\mathrm{h}')^T x}d\mathbf{x}, \quad \mathrm{h}' = M^{-1}\mathrm{u} \qquad (1)$$

We compute the 2D Fast Fourier Transform (FFT) of the pixels frames. In general, the grid point (u,v) in the plane of the image is a point (h', k', ℓ') in space which does not coincide with a mesh point $\mathbf{h} = (h,k,\ell)$ of the 3D grid. By direct integration, (1) leads to

$$P(u,v) = \sum_{h} F(\mathrm{h})\mathrm{sinc}(\pi[h-h'])\mathrm{sinc}(\pi[k-k'])\mathrm{sinc}(\pi[\ell=\ell']) \quad (2)$$

where $\mathrm{sinc}(x) = \sin(x)/x$. This leads to one of the methods suggested in [Crow70].

Suppose each frame contains N-by-N pixel values. and suppose we have an N-by-N-by-N set of $\mathbf{h}$ grid points. There are N^3 unknowns, $F(\mathbf{h})$, and thus we must have at least N pixel frames. Because the pixels are obtained experimentally, we must have more than just N frames; suppose there are νN frames, with $\nu>1$. Then there are νN^3 equations each having N^3 unknowns. We would solve this system as a least squares problem, using a singular value decomposition least squares solver. The number of operations required is the order of magnitude of the number of equations times the square of the number of unknowns; i.e., it is of the order $(\nu \mathbf{N}^3)\ (N^3)^2 = \nu N^9$; for detailed operations counts, see [Gol96]. For $N = 300$ and $\nu=2$, this is about 4×10^{22}. As pointed out in [Crow70], this is computationally unfeasible without some kind of additional restrictions. Rossmann uses symmetry and special properties of the coefficients of the resulting linear system to reduce the amount of computation; his ingenious scheme is presented in [Ros98].

[Crow70] suggests the use of interpolation of the transform of the pixel values. By interpolation, one can obtain an estimate of P at (u',v') so that $\ell' = \ell$ in (2)and then that equation reduces to

$$P(u',v') = \sum_{\mathrm{h}} F(\mathrm{h})\mathrm{sinc}(\pi[h-h'])\mathrm{sinc}(\pi[k-k']) \quad (3)$$

One obtains one system for each of the N values of ℓ, so one has N systems each having νN^2 equations with N^2 unknowns. The order of magnitude of the number of operations to solve the systems is $N(\nu N^2)(N^2)^2 = \nu N^7$, a factor of N^2 smaller than solving (2). For $N = 300$ and ν=2, this is about 4×10^{17}.

[Crow70] also points out that one could use interpolation so that the point (u',v') corresponds to a point $\mathbf{h}'$ having components k'= k and $\ell' = \ell$. Then (2) reduces to

$$P(u',v') = \sum_{\mathbf{h}} F(\mathrm{h})\mathrm{sinc}(\pi[h-h']) \quad (4)$$

Here there are N^2 systems each having νN equations and N unknowns. The order of magnitude of the number of operations is $(N^2)(\nu N)(N)^2 = \nu N^5$. This is two orders of magnitude smaller than used for the solution of (3) and four orders of magnitude smaller than for (2). For $N = 300$ and $\nu = 2$, this is about 5×10^{12}.

This third formulation of [Crow70] is the one we chose to implement. Moreover, we noticed that the experimental data can be used three times, once to solve (4) and once each to solve the next two systems

$$P(u'',v'') = \sum_{\mathbf{h}} F(\mathrm{h})\mathrm{sinc}(\pi[k-k']) \quad (5)$$

$$P(u''',v''') = \sum_{\mathbf{h}} F(\mathrm{h})\mathrm{sinc}(\pi[\ell-\ell']) \quad (6)$$

In summary, the calculation proceeds as follows for (4) for given k'= k and ℓ'= ℓ. We use the rotation matrix M to determine h' and corresponding (u',v') so that this point on the plane of pixels intersects the set of 3D grid lines at $(h',k,\ \ell)$. The bilinear interpolant of the transform of the pixel values is used to obtain the estimate $P(u',v')$. For each k, ℓ, the value of h' and the real and the imaginary parts of the complex number $P(u',v')$ are stored. This is done for all pairs of (k,ℓ). Each node does this for the set of pixel frames assigned to it. When this is completed the information, h', $\Re[P(u',v')]$ and $\Im[P(u',v')]$ are exchanged among nodes so that all information for a fixed pair $(k,\ \ell)$ resides on a single node. The (real) singular value decomposition least squares solver SGELSS from LAPACK [And92] is used to solve each of the systems.

This is done in such a way that each processor has a complete set of of (h,k) for those values of ℓ assigned to it. The node can, therefore, carry our a 2D FFT synthesis to transform the h and k to x and y, respectively. A second exchange of information is required to collect a complete set of ℓ-data for a subset of the $(x\ ,y)$-data and then a 1D FFT synthesis transforms the ℓ to z.

The solution of the systems (5) an (6) are obtained in a similar way, and at the same time. Once all three set of estimates of ρ are obtained, the three are averaged to get the final output.

One of the 'tricky' parts of the calculation occurs because the solution of (4) does not produce a complete set of estimates of $F(h\ ,k,\ \ell)$. When k = ℓ = 0 one has the single equation $P(0,0) = F(0,0,0)$ and there are no equations which lead to estimates for example, of $F(0,0,\ell)$ with $\ell \neq 0$. However, the solution of (6) does produce such estimates (but none for $F(h,0,0)$ with $h \neq 0$). Consequently, to get a complete set of solutions, $F(\mathbf{h})$

of one of the three sets of equations, one must use some of the results obtained by solving the other two. To accomplish this without additional exchange of data among nodes requires careful distribution of the equations.

SVD, [Crow70] solves the least squares systems by forming the normal equations, inverting the resulting matrix, and multiplying by the inverse; that is, for the least squares problem $A\mathbf{x} = \mathbf{b}$, where A has m columns and n rows with $m > n$, one forms half of the symmetric matrix $B = A^T A$ with about $(mn^2)/2$ operations, inverts B with about $2n^3$ operations, and then forming the product $B^{-1}\mathbf{x}$ with n^2 operations. Since the direct solution of the symmetric system $B\mathbf{x} = A^T \mathbf{b}$ takes only $n^3/6$ operations, one saves a factor of 4 by *not* inverting the matrix. Instead of using the normal equations, we use a least squares solver which makes use of the Singular Value Decomposition (SVD). The amount of arithmetic is a bit more than with normal equations (without inverting B), but the components of the SVD solution do have, typically, *twice* as many digits of accuracy than the solution obtained with the normal equations. This is of importance especially when using REAL*4 arithmetic where the numbers have about 8 decimal digits; if the condition number of A is about 10^4, then one loses all accuracy when using the normal equations, but still has about 4 digits of accuracy with SVD.

A major challenge in the design of a parallel algorithm is finding a decomposition of a large problem into P smaller problems that can be solved as independently as possible on the P processors of a parallel system. For the first equation above we can solve M/2P equations on each processor. We can partition the second and third set of equations in a similar fashion.

5. Implementation of the parallel algorithm. Communication and Load Balancing.

The implementation of both algorithms is based upon the Same Program Multiple Data paradigm, SPMD. The critical issues for the performance of the parallel implementation of the algorithms are data partitioning, load balancing, and hiding the communication latency. Both programs consist of several computation phases. At the end of each computation phase global exchanges of messages occur. To minimize synchronization delays the load should be balanced among processors for every execution phase.

We outline the implementation of the algorithms for orientation determination and 3D-reconstruction respectively. Orientation determination consists of construction of the database and the database search. For any given iteration we first construct a low-resolution database and determine the best fit, during the global step of the algorithm. With hints from this step we construct a high-resolution database and conduct the search for each projection within a much smaller region of the entire database in the refinement step, as shown in Figure 3. For the global search the database is rather small and it is constructed independently by each node. In the refinement step the database is much larger and each node builds a section of it, concurrently with other nodes. In both cases each node is assigned a region of projections and determines the "best fit" for each projection.

We have designed the algorithm for 3D reconstruction to work well on Origin 2000. The program uses MPI and consists of several phases, initialization, 2D Fourier

transformation and forming the systems of linear equations, solving linear systems and Fourier synthesis. Table 2 presents a more refined view of the computation, with 10 phases. In the second phase of the algorithm, pixel frames are distributed evenly among nodes and processed independently in each node. The interpolants used to form the linear equations are calculated and collected for all three directions in a node. These data need to be put together to form a linear system of equations for a grid point. After solving the linear systems, the Fourier coefficients on the 3-D reciprocal space are obtained and naturally distributed among nodes. Therefore, 2-D Fourier synthesis can be done in each node in parallel without data exchange.

6. Experimental results

6.1 The execution profile for the orientation determination

Table 1 presents the phases of the global search step of the orientation determination algorithm and the time spent in each phase for one particular run. The refinement step is yet to be implemented.

Phases	***Execution time*** (Sec)
(1) Initialization (seq, I/O)	1.52
(2) Construct the database (par)	7.52
(3) Data exchange (com)	2.98
(4) Database search (par)	26.42
(5) Collect/write results (com,I/O)	1.80

Table 1. The execution time in seconds for each phase of the global search in orientation determination. (seq) indicates a sequential execution phase of the program, (par) a parallel execution phase, and (com) a communication phase, (I/O) input-output.

These measurements are used to identify the time-consuming phases of the execution, to estimate the potential speedup of the algorithm, and to estimate the impact of communication upon performance. The measurements indicate that the database search dominates the execution time. Fortunately, this phase can be carried out in parallel therefore we expect decent speedups for large databases. Let *Seq* denotes the sum of the execution time of "seq" and "com" phases and *Par* that of "par" phases, then the ratio *Par/Seq* is an indicator on the potential speedup when the number of nodes increases. In this case the ratio is 5.387 indicating that if we double the number of nodes the speedup will increase by a factor of approximately 1.7.

6.2 The execution profile for the 3D reconstruction algorithm.

Table 2 presents the phases of the 3D reconstruction program and reports preliminary measurements indicating the time spent in each phase for a particular run. The problem uses 5000 pixel frames of size of 51 x 51 pixels to reconstruct the object in a grid of 51 x 51 x 51 pixels. Because the program for 3D-reconstruction is much more complex than that for orientation determination, we profile the phases in more detail and with more

accurate measurements. This profile indicates that the computation of the 2D Fourier transforms and solving of the linear systems are dominant. Any optimization of the implementation should focus on these two phases.

Phases	*Execution time* (Secs)
(1) Initialization (seq, I/O, com)	0.09
(2) Read/Broadcast pixel frames (I/O, com)	5.79
(3) 2D Fourier transform (par)	25.21
(4) Interpolation for setting linear systems (par)	5.69
(5) Data exchange for solving linear system (com)	7.43
(6) Solve linear systems (par)	107.73
(7) 2-D Fourier synthesis (par)	0.60
(8) Data exchange for 1-D Fourier synthesis (com)	0.01
(9) 1-D Fourier synthesis (par)	0.31
(10) Gather & write density (seq, I/O, com)	41.98

Table 2. The execution time in seconds for each phase of the 3D reconstruction reported by the master node. (seq) indicates a sequential execution phase of the program, (par) a parallel execution phase, and (com) a communication phase, I/O an Input/Output phase.

I/O times are significant. The tests reported here are done with data generated internally and there is no I/O involved in phase (2). Soon we will start testing with external (experimental) data and we expect the execution time to increase depending upon the number of projections in the input file. We have two choices for reading the input data. The first is to have one node read individual images and send each image to the node assigned to process it. The second is parallel I/O, allowing each node to read the images assigned to it. The second choice is optimal for a parallel file system when input files are stripped across multiple I/O nodes. This solution is impractical for Unix file systems where I/O contention leads to a severe performance penalty. The I/O becomes critical for high-resolution image reconstruction where we could have 10,000 images, each with 100 by 100 pixels, i.e. 300 MB of input data.

The same strategies can be used to write out the results. One node may gather the results from all the nodes and write the density file. This is the method we are currently using. The second is parallel I/O, allowing each node to write the data it has generated. The second solution is unfeasible for Unix file systems accessed via the Network File System, because of the internal buffering performed by NFS.

The communication time is about 5% of the total execution time. It will probably increase when running the program with external data due to the need to distribute the frames. We plan to compare two version of the program, one running on a parallel system and one running on a cluster of workstations. We do not expect substantial performance degradation when running on a cluster because the relatively low weight of the communication in the overall execution times.

6.3 Load balancing and speedup for the 3D reconstruction algorithm.

Preliminary results obtained on an Origin 2000 system indicate that our load balancing strategy for the 3D reconstruction discussed above works well. Table 3 shows the execution time for two problems running on 8 nodes of an Origin 2000 system. Our results indicate that the algorithm is capable of quasi-linear speed up when the last phase for displaying/writing results is omitted. For both problems, we obtained a near six fold speed up with eight nodes.

	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8
A	44.1	41.0	41.0	41.0	41.0	41.0	41.0	41.0
B	195.0	153.3	153.3	153.3	153.3	153.3	153.3	153.3

Table 3. The execution time in seconds on each of the eight nodes of an Origin 2000 system for two problems. Problem A, 2,000 projections (each of size 21×21) and Problem B, 5,000 projections each of size 51×51.

Table 3 indicates that the master (node 1) needs about 195-153 = 42 seconds to write the results collected from all other nodes. We can still reduce the execution time by 20% or more by parallel I/O. Unfortunately this is not possible with clusters of workstations connected via NFS to Unix file systems.

7. Conclusions

The 3D reconstruction of asymmetric objects is a computationally and data intensive problem. It consists of iterative cycles of orientation determination and 3D reconstruction. In this paper we present parallel algorithms for both orientation determination and 3D reconstruction in Cartesian coordinates.

The algorithm for orientation determination consists of two steps, the global search and the refined search. For both steps a reference database is constructed and each projection is matched against all database components to get a "best fit". For the global search the rather small database is constructed independently by every node and then each node is assigned an equal number of images. For the refined search both the database construction and the search are done in parallel.

The algorithm for 3D reconstruction is based upon an entirely new implementation of the 1D method for interpolation in the Fourier domain [Crow70]. Instead of solving a large least squares problem we solve a number of independent problems of a smaller size. Preliminary results indicate that the computational load is evenly distributed among nodes.

The profiling of the 3D reconstruction program indicates that algorithmic load balance is a necessary but not a sufficient condition for optimal execution. Though we have designed an algorithm that enables us to distribute the computational load evenly among nodes, practical considerations require I/O operations to be carried out by a single

node when running the program on a cluster of workstations. The time for reading the input and writing the output data can be reduced significantly on a system supporting a parallel file system.

8. Acknowledgments

The authors express their gratitude to Prof. Michael Rossmann for an early version of his manuscript on 3D reconstruction. K.C. vanZandt has contributed to a version of the parallel orientation algorithm. The work reported in this paper was partially supported by a grant from the National Science Foundation, MCB-9527131, by the Scalable I/O Initiative, by a grant from the Computational Science Alliance, and by a grant from the Intel Corporation.

9. References

[And92] Anderson, E., Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A, McKenney, S. Ostrouchov, D. Sorenson, *LAPACK, Users' Guide*, SIAM Publications, 1992.

[Bak96] Baker, T. S., and R. H. Cheng, "A Model-Based Approach for Determining Orientations of Biological Macromolecules Imaged by Cryoelectron Microscopy", J. Structural Biology, 116, 1996, **120--130**.

[Crow70] Crowther, R. A., D. J. DeRosier, and A. Klug, "The Reconstruction of a Three-Dimensional Structure from Projections and its Application to Electron Microscopy", Proc. Roy. Soc. Lond. A 317, 1970, **319-340**.

[Crow71] Crowther, R. A., "Procedures for Three-Dimensional Reconstruction of Spherical Viruses by Fourier Synthesis from Electron Micrographs", Philos. Trans. Roy. Soc. London Ser. B 261, 1971, **221-230**.

[Dea93] Deans, S. R., *The Radon Transform and Some of Its Applications*, 2nd Edit., Krieger Publishing Company, 1993.

[Ful96] Fuller, S. D., S. J. Butcher, R. H. Cheng, and T. S. Baker, "Three-Dimensional Reconstruction of Icosahedral Particles. The Uncommon Line", J. Structural Biology, 116, 1996, **48-55**.

[Gol96] Golub, G. H., and C. F. Van Loan, *Matrix Computations*, 3rd. Edit., The Johns Hopkins Univ. Press, 1996.

[Joh94] Johnson, C. A., N. I. Weisenfeld, B. L. Trus, J. F. Conway, R. L. Martino, and A. C. Steven, "Orientation Determination in the 3D Reconstruction of Icosahedral Viruses Using a Parallel Computer", IEEE Computational Science and Enginering ,1994, **550-559**.

[Lyn97] Lynch, R. E., and D.C. Marinescu, "Parallel 3D reconstruction of spherical virus particles from digitized images of entire electron micrographs using Cartesian coordinates and Fourier analysis", CSD-TR #97-042, Department of Computer Sciences, Purdue University, August 1997.

[Mar97] Martin, I. M., D. C. Marinescu, T. S. Baker, and R. E. Lynch, "Identification of Spherical Particles in Digitized Images of Entire Micrographs", Journal of Structural Biology, 120, 1997, **146-157**.

[Ros98] Rossman, M. G., and Y. Tao, "Cryo-Electron Microscopy Reconstruction of Partially Symmetric Objects", Department of Biological Sciences, Purdue University, July 1998 (submitted for publication).

[She80] Shepp, L. A., "Computerized Tomography and Nuclear Magnetic Resonance", J. Comput. Assit. Tomog. 4, 1980, **94-107**.