A Parallel 3D Piecewise Constant Reconstruction Algorithm for Asymmetric Virus Structures

Robert E. Lynch¹, Yongchang Ji², Dan C. Marinescu², and Hong Lin³

¹ Department of Computer Sciences, Purdue University, West Lafayette, IN, 47907, USA
rel@cs.purdue.edu

² School of Electrical and Computer Engineering, University of Central Florida, Orlando, Florida, 32816, USA
{yji, dcm}@cs.ucf.edu

³ Department of Computer and Mathematical Sciences, University of Houston-Downtown, Houston, Texas 77002, USA
LinH@uhd.edu

Abstract. In this paper we present an improved method for computing the electron density of virus structures; it uses $O(N^2)$ less work and is based on piecewise constant approximation of the Discrete Fourier Transform of the electron density. In these algorithms, the computations to determine the values of the 3D Discrete Fourier Transforms of the density of an asymmetric object can be naturally distributed over the nodes of a parallel system. Some computational results, pertaining to the accuracy, are also given.

1 Introduction

There are several practical methods for reconstructing a 3D object from a set of its 2D projections. These include use of Fourier Transforms, back projection, and numerical inversion of the Radon Transform. See [Gor74] for a review of these and other methods. For descriptions of (sequential) methods for 3D reconstruction and related tasks, see [Dea93], [Fra96], [Gra96], and [Her79] four of several books containing clear explanations and many references; see also the extensive review [Bak98]. A popular sequential algorithm implemented in the 3D reconstruction programs used by the structural biology community employs cylindrical coordinates and Fourier Bessel transforms, see [Cro70]; this method is especially effective for the reconstruction of particles having symmetry, such as icosahedral symmetry.

In [Lyn99] an outline is given of our first parallel algorithm for 3D reconstruction. It was based on one of five methods presented in [Cro70]. The parallel algorithm uses Cartesian coordinates, the ‘sinc’ function for interpolating the Fourier Transform, and requires the solution of an algebraic least squares
problem. The method permits the reconstruction of objects that do not possess
symmetry. For reconstruction at points of an $N \times N \times N$ grid, the algorithm
uses $O(N^5)$ arithmetic operations.

In this paper we discuss an improvement of the algorithm in [Lyn99]. The new
parallel algorithm uses piecewise constant interpolation to estimate the Discrete
Fourier Transform of the electron density. It uses $O(N^3)$ operations and, as in
our previous algorithm, it can be used to reconstruct asymmetric particles.

2 Reconstruction and Parallel Processing

The experimental information for reconstruction is gathered from digitized im-
age s of many identical macromolecules obtained with a cryo-electron microscope
and individual particle projections are identified. For each individual 2D image,
which we call a ‘pixel frame’, the location of the projection of the centroid of
the molecule’s electron density is determined, as is the orientation of the image with
respect to a standard Cartesian System. See [Lyn99] for more details. Then the
reconstruction of the asymmetric object is carried out as follows:

Step 1 Compute the 2D Discrete Fourier Transform (DFT) of each projection.
Step 2 Using the 2D DFTs and their orientations, compute an approximation
to the 3D DFT of the electron density.
Step 3 Compute the inverse 3D DFT to get an approximation to the electron
density.

In this paper we describe the use of piecewise constant approximation to
carry out Step 2; see Section 3. Here we outline the use of parallel processing.

Some of the computations can be done independently from each other. For
example, the evaluation of the 2D DFT of one pixel frame does not depend on any
other pixel frame. In Step 1, each processor is assigned a set of projections and
that processor computes the 2D DFTs of those pixel frames in its set, concurrently
with transformations taking place in the other processors.

To use efficiently a parallel computer or a cluster of workstations, we use
a parallel algorithm that partitions the data and computations evenly among
nodes to ensure load balance and, moreover, which minimizes the communication
among processors by maintaining a high level of locality of reference.

Our implementation of the algorithm for 3D reconstruction was designed to
work well on workstation clusters and on computers with many processors. The
program uses MPI (message passing interface) and consists of several phases
including initialization, evaluation of 2D DFT’s, estimation of the 3D DFT, and
evaluation of the inverse 3D DFT. To avoid input/output conflict, a single node
reads all the pixel frames and distributes groups of frames to other processors.

As explained in Section 3, knowing the orientation, the center of projection,
and the 2D DFT of a pixel frame, we can find values of its piecewise constant
interpolant to the 3D DFT, $F$, of the electron density at grid points, $(h, k, \ell)$,
which are sufficiently close to the plane of projection. When more than one
projection gives an interpolant at the same point, then an average is computed
(see Section 3). Some of this computation can be included in Step 1, after the 2D DFT of a pixel frame has been obtained. Each processor computes estimates of the 3D DFT of the electron density, $F(h, k, \ell)$, at those grid points which are near the plane of projection of the pixel frame. After all the pixel frames have been processed, an exchange of data takes place, to put all $F(h, k, \ell)$ having $k$ in the interval $k_i \leq k < k_{i+1}$ onto Node $i$; i.e., the $i$-th ‘$k$-slab’ is put onto Node $i$. When the value of $F(h, k, \ell)$ has been estimated by more than one node, then the values are averaged, see Section 3.

The inverse 3D DFT gives the estimate of the electron density; it is obtained in two steps. Node $i$ has values of $F(h, k, \ell)$ for all $h$ and $\ell$, and that node computes the inverse 2D DFT, $h \rightarrow x$, $\ell \rightarrow z$, $F(h, k, \ell) \rightarrow G(x, k, z)$, for each $k$, $k_i \leq k < k_{i+1}$. This gives a ‘$k$-slab’ of values $G(x, k, z)$. Another exchange of data among nodes is done to form ‘$z$-slabs’: $G(x, k, z)$ for all values of $x$ and $k$ and values of $z$ in $z_i \leq z < z_{i+1}$. After the computation of all of the 1D inverse DFTs with respect to $k$, $k \rightarrow y$, we have the final estimate, $\rho(x, y, z)$, of the electron density at points in a $z$-slab. Each $z$-slab is written onto the output device.

3 Projections onto Slabs

The Fourier Transform, $F$, of the electron density, $\rho$, is

$$F(h, k, \ell) = \int \int \int \rho(x, y, z) \exp(-2\pi i [hx + ky + \ell z]) \, dx \, dy \, dz,$$

where the integration is over a finite domain containing the nonzero values of $\rho$. Following [Cro70], we use the ‘projection theorem’ to relate $F$ to the 2D information given by the experimental data. It states that on any plane containing the origin, $F$ is equal to the Fourier Transform (FT) of the projection of $\rho$ onto that plane. This is evident for the plane $\ell = 0$: (1) shows that $F(h, k, 0)$ is equal to the 2D FT with respect to $x$ and $y$ of $\int \rho(x, y, z) \, dz$; rotations of the coordinate systems gives the general result.

We approximate the 3D FT with the 3D Discrete Fourier Transform (DFT). We use $F(h, k, \ell)$ to denote also values of the discrete transform.

The experimental data consist of a collection of digitized electron micrographs, or ‘pixel frames’. We assume that each square pixel frame represents the image of a projection of $\rho$ onto a plane. Then its 2D FT gives values $P(u, v, 0)$ on the plane $w = 0$ with respect to a $(u, v, w)$-coordinate system having its $w$-axis normal to the plane of projection. By preprocessing, the orientation and the location of the center of each pixel frame is determined, see [Bak98], [Ji02]. Having the orientation, one can evaluate the entries of a $3 \times 3$ invertible matrix which transforms any point $(u, v, w)$ to the corresponding point $(h, k, \ell)$. Thus one can take any point $(h, k, \ell)$ and find the corresponding point $(u, v, w)$, and conversely.

We approximate the FT of a pixel frame by its DFT, which we also denote by $P(u, v, 0)$. Multiplication by appropriate fractional powers of $\exp(-2\pi i)$ translates the center of projection to the origin of the $(h, k, \ell)$-system. The values of
\( P(u,v,0) \) can be corrected for astigmatism and the contrast transfer function can be applied, etc., see [Bak98], [Fra96].

Instead of using the \( \text{sinc}(t) = \frac{\sin(t)}{t} \) function, as in [Cro70] and in our earlier algorithms, we now use piecewise polynomial approximation to relate values of \( P(u,v,0) \) to values of \( F(h,k,\ell) \). In this paper we report on \textit{piecewise constant} approximation. We extend \( P(u,v,0) \) defined at integer grid points, \( u \) and \( v \), to a piecewise constant function defined at noninteger points by

\[
P(u+\Delta u,v+\Delta v,\Delta w) = P(u,v,0), \quad u,v \text{ integers}, -0.5 < \Delta u, \Delta v, \Delta w \leq 0.5.
\]

(2)

This gives a function defined at points on a \textit{slab} containing the \( w = 0 \) plane. Figure 1 illustrates this when the 3D domains \((h,k,\ell)\) and \((u,v,w)\) are simplified to 2D domains \((h,k)\) and \((u,v)\); the pixel frame slab becomes a “stick” with \(-1/2 \leq v \leq 1/2\).

![Slab Projection Simplified to the 2D Case](image)

\textbf{Fig. 1.} Slab projection simplified to the 2D case. The domain of the ‘pixel frame,’ shown on the left, is in its correct orientation in the 2D \((h,k)\)-domain on the right. The Discrete Fourier Transform at a grid point \((h,k)\) is estimated by interpolating values from the pixel frame; interpolation at two points are indicated by the shaded squares.

From the known orientation of the pixel frame, one can find all integer grid points \((h,k,\ell)\) which are in the slab (see Figure 1). Each of these is used to adjust the current estimate of \( F(h,k,\ell) \):

\[
\begin{align*}
\text{IF this is first estimate} & \quad \{ F(h,k,\ell) \leftarrow P(u,v,0) \quad s \leftarrow 1 \} \\
\text{ELSE} & \quad \{ F(h,k,\ell) \leftarrow [s F(h,k,\ell) + P(u,v,0)]/(s+1) \quad s \leftarrow s + 1 \}
\end{align*}
\]

Here we have described the case in which the grid spacings of the \((h,k,\ell)\) and the \((u,v,w)\) spaces are the same. However, in general, these spacings are different, for example, when we use ‘zero-fill’, see Section 4. It is easy to make this adjustment, which is just a change of scale, when the coordinate systems are rotated.
4 Interpolation by Zero-Fill (Zero Padding)

The DFT of a 2D $M \times M$ pixel frame can be computed by two calls to a subroutine which computes a set of $M$ 1D DFT’s. Thus it is sufficient to consider 1D transformations of length $M$. If the subroutine does not support transformations of length $M$, then it is customary to increase the length to $N$, the smallest integer greater than $M$ which is supported; the additional $N - M$ entries equal are set equal to zero. More generally, any $N$ that can be transformed by the subroutine and is greater than $M$ could be used. We call this padding with zeros “zero-fill” and call $k = N/M$ the “aspect ratio”. We now show that zero-fill results in an (exact) interpolant to the original DFT.

We use $f_j$ to denote the values of a function which is given at $M$ equal spaced points: $f_j = f(j \Delta x)$. When $M$ is even, the DFT of $f$ is

$$ F_h = \sum_{j=-M/2+1}^{M/2} f_j \exp(-2\pi j h/M), \quad h = 0, \pm 1, \ldots, \pm M/2-1, M/2, \quad (3) $$

and when $M$ is odd, the lower limit of the summation is $-M/2$ and then $F_{-M/2}$ is also defined. Here we have taken unit normalization for the sum; other popular normalizations include multiplication of the sum by $1/\sqrt{M}$ or by $1/M$.

We use $G$ to denote the DFT of the transform of the zero-filled function; for even $N$:

$$ G_{h'} = \sum_{j=-M/2+1}^{M/2} f_j \exp(-2\pi j h'/N), \quad (4) $$

where the limits of the sum are the same as in (3) because values of $f_j$ are zero for $j$ outside the limits of the sum. Values of $G$ are given for $h' = 0, \pm 1, \ldots, \pm N/2 - 1, N/2$ when $N$ is even; when $N$ is odd, $G_{-N/2}$ also defined.

Because $\exp(-2\pi i j h'/N) = \exp(-2\pi i j [h' M/N]/M)$, it follows that whenever $h'M/N$ is equal to an integer $h$, $G_{h'}$ is equal to $F_h$. For example, if $M/N = 1/2$, then the values $G_{h'}$ at $h' = 0, \pm 2, \pm 4, \ldots$ are equal, respectively, to $F_h$ at $h = 0, \pm 1, \pm 2, \ldots$. Thus, $G$ is an interpolant to $F$: the values $F_h$ are equal to $G_{2h}$ for the integers $h = 0, \pm 1, \pm 2, \ldots$. This is illustrated in Figure 2 for a simple case with $M = 8$ and $N = 16$.

The preceding remarks summarize the discussion in [Bri95], pp. 90–91. Problem 53 posed on pp. 113 of [Bri95] extends the result: “... show that it is possible to interpolate at $k - 1$ points between each [pair of] term[s] by padding the original sequence with zeros to a length of $k N$.” ([Bri95] use “p” where we use “k”).

But, $G$ not only interpolates to $F$ at a discrete set of points, it is equal to $F$ at all points; i.e., $G$ is the same function as $F$. When first stated, this result might seem surprising. However, one sees from (3) that the $M$ given values of $f_j$ are the coefficients of a trigonometric polynomial of degree $M - 1$. A demonstration of the equality of $F$ and $G$ is obtained by observing that when $h/M$ is replaced with $h'/N$, the right side of (3) becomes the right side of (4);
Fig. 2. The effect of zero-fill, 1D case. (1a): An 8-point step function; the circles represent the density inside and outside a uniform sphere; the function is periodic, period 8. (1b): The real part of the 8-point DFT of (1a), the imaginary part is zero because the step function is symmetric. (2a): The function (1a) and zero-fill with aspect ratio $k = 2$. (1a) is augmented by 8 zero values, indicated by crosses, the spacing is the same as in (1a); the function is periodic, period 16. (2b): The real part of the 16-point DFT of (2a). The spacing in (2b) is half that in (1b). The values, indicated by circles, at $0, \pm1, \pm2, \pm3, 4$ are the same as those in (1b); the 8 values indicate by ‘x’ s are due to the zero-fill. This illustrates that zero-fill in real space results in interpolation in reciprocal space.

and when $h/M = h'/N = t$ we have

$$F_h = F(h/M) = F(t) = \sum_{j = -M/2 + 1}^{M/2} f_j \exp(-2\pi i j t) = G(t) = G(h'/N) = G_h',$$

which holds for all $t$.

Following [Bri95], we use $A$ to denote the length of the domain of $f$, $-A/2 < x \leq A/2$, we take the grid spacing to be $\Delta x = A/M$, and grid points $x_j = j \Delta x$. Also, we use $\Delta \omega$ to denote the grid spacing and $\omega_h = h \Delta \omega$ to denote the grid points in the domain of $F(\omega)$; then we have

$$jh/M = (j A/M)(h/A) = (j \Delta x)(h \Delta \omega) = x_j \omega_h,$$

so that $\Delta \omega = 1/A$. Consequently, if the units of the $x$-axis are nanometers, then the units of $\omega$ are reciprocal nanometers; the domain of $x$ is in ‘real’ space and that of $\omega$ is in ‘reciprocal’ space. The transform, $F(\omega)$, of $f(x)$ is defined on the interval $-M/(2A) < \omega \leq M/(2A)$. Both $f(x)$ and $F(\omega)$ are periodic: $f(x)$ is periodic with period $A$ and its transform, $F(\omega)$, is periodic with period $M/A$. 
As an application, consider the 2D case illustrated in Figure 1. Suppose the spacing of the 2D grid is the same as that of the 1D stick. Let \( \zeta \) denote the angle between the \( h \)-axis and the centerline of the stick. When \( \zeta = 0 \), there is no interpolation error because the grid points in the stick are also grid points on the \( h \)-axis. As \( \zeta \) increases, the separation between points in the two sets increases. When \( \zeta = 45^\circ \), the grid points \( (h, h) \) along the diagonal of the 2D grid are non-grid points in the stick. Because there is unit spacing between adjacent grid points in the stick, the distance between a grid point \( (h, h) \) and one that is in the stick is at most \( 1/2 \). If the zero-fill factor is \( 2 \), then the maximum distance decreases by a factor of \( 2 \), and, neglecting errors in the pixel values, so does the maximum piecewise constant interpolation error.

5 Numerical Errors in 3D Reconstruction

In this section we consider some of the numerical errors. To determine the accuracy of the reconstruction method, we took a known object and generated its projections for randomly selected orientations. Then we reconstructed the object from these projections and compared the known values with the computed ones. Our experiments were conducted for a uniform sphere: unit density inside the sphere and zero outside.

As described in Sections 3 and 4 we used piecewise constant interpolation and averaging at grid points to compute estimates \( F(h, k, \ell) \) of the 3D DFT of the test object at grid points. We did experiments for several sets of values of parameters: \( P \) the number of grid points on a side of the \( P \times P \) pixel frame; the diameter \( D = \alpha P \), \( 0 < \alpha < 1 \), of the sphere; the number of \( V \) (views) of randomly oriented frames; and the aspect ratio \( k \) of zero-fill. The input \( P \times P \) pixel frame was enlarged to a \( kP \times kP \) array, with zero-fill. The orientation of its 2D DFT was selected randomly for \( V \) views. The object was reconstructed at points of a \( P \times P \times P \) grid. We determined the maximum and minimum pointwise error for various values of the parameters. To learn how the error varied with radius, we computed the mean square error (square root of the sum of the squares of the error) in a set of annular regions inside the test object.

Table 1. Mean square percent error inside reconstructed uniform spheres and the effect of the zero-fill aspect ratio \( k \) on the minimum/maximum reconstructed density of a sphere. Input: \( V \) projections, onto \( P \times P \) pixel frames, of a sphere having diameter \( D \) and unit density.

<table>
<thead>
<tr>
<th>Aspect ratio ( k )</th>
<th>( P \times P = 41 \times 41 )</th>
<th>( P \times P = 61 \times 61 )</th>
<th>( P \times P = 81 \times 81 )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Diameter ( D = 32 )</td>
<td>Diameter ( D = 48 )</td>
<td>Diameter ( D = 64 )</td>
</tr>
<tr>
<td></td>
<td>Views ( V = 20100 )</td>
<td>Views ( V = 45150 )</td>
<td>Views ( V = 80802 )</td>
</tr>
<tr>
<td>1</td>
<td>20.16</td>
<td>0.57/1.00</td>
<td>23.38</td>
</tr>
<tr>
<td>2</td>
<td>6.92</td>
<td>0.86/1.00</td>
<td>7.97</td>
</tr>
<tr>
<td>4</td>
<td>1.63</td>
<td>0.97/1.00</td>
<td>2.11</td>
</tr>
</tbody>
</table>
Table 2. The effect on the least square % error of the number of projections and the effect on the minimum/maximum of the density inside a reconstructed sphere: $V$ is the number of projections of size $P \times P$, the uniform sphere has diameter $D$ the zero-fill aspect ratio is $k = 4$

<table>
<thead>
<tr>
<th>Views $V$</th>
<th>$P \times P = 21 \times 21$ Diameter $D = 16$</th>
<th>$P \times P = 41 \times 41$ Diameter $D = 32$</th>
<th>$P \times P = 61 \times 61$ Diameter $D = 48$</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>1.68 0.97/1.00</td>
<td>2.01 0.96/1.00</td>
<td>2.21 0.96/1.00</td>
</tr>
<tr>
<td>1250</td>
<td>1.41 0.98/1.00</td>
<td>1.77 0.97/1.00</td>
<td>1.88 0.96/1.00</td>
</tr>
<tr>
<td>2775</td>
<td>1.39 0.97/1.00</td>
<td>1.78 0.97/1.00</td>
<td>1.90 0.96/1.00</td>
</tr>
<tr>
<td>5000</td>
<td>1.34 0.98/1.00</td>
<td>1.72 0.97/1.00</td>
<td>1.88 0.96/1.00</td>
</tr>
<tr>
<td>11250</td>
<td></td>
<td>1.67 0.97/1.00</td>
<td>1.89 0.96/1.00</td>
</tr>
<tr>
<td>20100</td>
<td></td>
<td>1.63 0.97/1.00</td>
<td>1.95 0.96/1.00</td>
</tr>
<tr>
<td>45150</td>
<td></td>
<td></td>
<td>2.11 0.96/1.01</td>
</tr>
</tbody>
</table>

5.1 The effect of zero-fill ratio $k$. Before the 2D DFT of the pixel frame is computed, the pixel values are put into a larger frame with the extra array entries set equal to zero. Errors as a function of the zero-fill aspect ratio $k$ are listed in Table 1. The mean square error decrease from $20 \rightarrow 25\%$ for $k = 1$ to about $2\%$ for $k = 4$. For a fixed $k$, the variation in the error is small as the size of the pixel frame changes with the ratio [diameter/(frame edge)] kept nearly constant (about 0.79). In Table 1 the values of the minimum and maximum of the computed density inside the sphere are also listed. The variation of the density becomes smaller as the aspect ratio increases.

5.2 The effect of the number of views $V$. The effect of the number of views $V$ on the least square error and on the minimum and maximum errors are listed in Table 2 for the aspect ratio $k = 4$. For fixed pixel size and diameter, the variation in the percent error varies very little as the number of views increases. Similarly, there is little change in the range of computed values inside the sphere.

We expect that increasing the aspect ratio of the zero-fill allows us to use fewer projections for the 3D reconstruction. Table 3 and Figure 3 show the results when the size of pixel frames is $41 \times 41$ and the number of pixel frames varies from 25 to 200 and for zero-fill aspect ratios of 1 and 4.

5.3 Radial distribution of the errors. Figure 4 indicates that the magnitude of errors increases with the radius. This increase is quite obvious in Figures 4.1b, 4.2b, 4.3b where we display the density as a function of the radius for $k = 1$. Figure 4, with $k = 4$, indicates that the results are improved slightly as the number of views increases.

6 Summary

In this paper we present a parallel algorithm for computing the electron density of virus structures. The computations to determine the values of the 3D Discrete Fourier Transforms of the density of an asymmetric object can be naturally
Table 3. The effect of the zero-fill aspect ratio, $k$, and the number of projections, $V$, on the percent errors inside the reconstructed sphere.

<table>
<thead>
<tr>
<th>$k$ \ $V$</th>
<th>25</th>
<th>50</th>
<th>75</th>
<th>100</th>
<th>150</th>
<th>200</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21.67</td>
<td>20.69</td>
<td>20.39</td>
<td>20.81</td>
<td>20.68</td>
<td>20.62</td>
</tr>
<tr>
<td>4</td>
<td>8.84</td>
<td>4.47</td>
<td>2.08</td>
<td>1.68</td>
<td>1.85</td>
<td>1.97</td>
</tr>
</tbody>
</table>

Fig. 3. The effect of the number of views, $V$, and the zero-fill aspect ratio $k$ on a reconstructed sphere. Figures (1a)–(1f): $k = 1$. Figures (2a)–(2f): $k = 4$. The size of pixel frames is $41 \times 41$ and the diameter of the sphere is 32.

distributed over the nodes of a parallel system. Some computational results, pertaining to the accuracy, are presented given.

7 Acknowledgments

The authors are grateful for many insightful discussions with Timothy S. Baker and Michael G. Rossmann. The research reported in this paper was partially supported by National Science Foundation grants MCB9527131 and DBI0296107 ACI0296035 and EIA0296179.

References

[Bak98] Baker, T. S., Olson, N. H., Fuller, S. D.: Adding the Third Dimension to Virus Life Cycles: Three-Dimensional Reconstruction of Icosahedral Viruses from...
Fig. 4. The effect of zero-fill on the computed density. The computed values for aspect ratios of zero-fill $k = 1$, 2 and 4. For a $P \times P$ pixel frame, for $V$ pixel frames, and for diameter, $D$: (1a)--(1c) $P = 41$, $V = 20100$, $D = 32$; (2a)--(2c) $P = 61$, $V = 45150$, $D = 48$; (3a)--(3c) $P = 81$, $V = 80802$, $D = 64$.


