1997

Parallel 3D Reconstruction of Spherical Virus Particles from Digitized Images of Entire Electron Micrographs Using Cartesian Coordinates and Fourier Analysis

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

Dan C. Marinescu

Report Number:
97-042

PARALLEL 3D RECONSTRUCTION OF SPHERICAL VIRUS PARTICLES FROM DIGITIZED IMAGES OF ENTIRE ELECTRON MICROGRAPHS USING CARTESIAN COORDINATES AND FOURIER ANALYSIS

Robert E. Lynch
Dan C. Marinescu

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907

CSD-TR #97-042
August 1997
(Revised 12/97)
(Note: Title Change)

Parallel 3D Reconstruction of Spherical Virus Particles from Digitized Images of Entire Electron Micrographs Using Cartesian Coordinates and Fourier Analysis*

Robert E. Lynch and Dan C. Marinescu
Departments of Computer Sciences and Mathematics

1. Introduction. Cryo-electron microscopy and 3D image reconstruction are the methods of choice for 3D atomic structure determination of many non-crystalline biological macromolecules. The structure determination process is rather complex, it involves preparation of samples, imaging using a cryo-electron microscope, identification of individual virus particle projections or micrographs, determination of the orientation of each particle, and the 3D reconstruction using information from 2D projections. The last two steps are iterated until the desired resolution is obtained.

The algorithm described in this paper and the program based on it use as input 2D projections of particles identified by a method described by Martin et al in [5] and the relative orientation of each projection with respect to a Cartesian coordinate system based upon the method proposed by Baker et al [3].

The method we use for 2D reconstruction is one described in [9] (also see [10], [11], [12], [13], [14], [15], [23], [18]). which uses Cartesian coordinate systems and Fourier transforms. The method described in [9] that uses cylindrical coordinate systems and transforms has successfully reconstructed many structures, [6], [8], etc.

We know of three general reconstruction methods (Dean, [16], pp. 126–128, gives a finer partition of methods). The one we discuss in this report makes use of a transform of 2D data, finds values of the corresponding 3D transform on a grid by interpolation, and then inverts the 3D transform to get the density of the object. Many 2D projections are used and, where possible, symmetry of the object and the imposed symmetry of the projection is exploited to obtain multiple sets of information from each 2D projection.

The second method, also described in [9], is called "back projection" (also see Dean [16], pp. 127–128 and 131-142). One considers the density of the object to be unknowns at points of a 3D grid. For each pixel of a 2D projection, one finds a straight-line path perpendicular to the projection and determines which of the unknows are near this path. One combines these unknowns in a linear equation together with the experimentally determined pixel values. Doing this for all the pixels in

*This research has been partially supported by the National Science Foundation grants BIR-9301210 and MCB-9527131, by the Scalable I/O Initiative, by a grant from the Intel Corporation

a projection and for many projections, one obtains a large linear system for the unknown density values at points of the 3D grid. Also see [27].

The third method is based on results of Radon which appeared in 1917 [30] (see [26] for an English translation of Radon's paper, see [4] for an outline of the method and see [25] and [16] for many details and applications of the method).

In rough terms: Radon proved that values of a (sufficiently smooth) function f on a bounded region in n-dimensional space can be determined if one knows all projections of f onto $(n-1)$-dimensional planes through the origin; moreover, Radon gave an explicit expression for values of f in terms of an integral involving the projections. The transformation from n-dimensional space to $(n-1)$-dimensional space has become known as the Radon transform, and the integral giving the function is known as the inverse Radon transform; see [38], [28], [29], [22],

One obtains a numerical scheme for reconstruction by discretizing the integral representation of the inverse Radon transform.

This has been done for many application and is the basis for some programs used in 'Computer Aided Tomography' (CAT scans), see [34], [31]. For this, $n = 2$ and one is reconstructing 2D slices with many 1D projections. Not only is the dimension n of this problem one less than the one we deal with, but also in CAT analysis, the specific orientation of the projections is known.

The literature on the subject of reconstruction is immense. Dean's book, [16], a reissue of his work which appeared a decade earlier, includes about 1000 references to the literature. Several books and conference proceedings have appeared since Dean's original book; for example [39], [32], [20].

2. Outline of Method. We describe a method which reconstructs the electron density of a spherical virus from many 2D projections of the virus. The experimental data are obtained from an electron micrograph containing hundreds of images of the particular virus under investigation.

Our program is designed to execute on a parallel computer, on a collection of workstations, or on a beowoulf class system.

We rely on prior intensive computer processing of a micrograph [5] to select a set of pixel 'frames'. Each frame contains the projection of the image of a single particle. Each square frame contains N-by-N pixel values; N might be several hundred. Another computer analysis [3] of the set of frames generates for the j-th frame the coordinates of the projection of the center of the particle, $(c_x, c_y)_j$, and also the orientation of the 2D frame with respect to a standard 3D Cartesian coordinate system; this is expressed in terms of three angles, $(\theta, \phi, \Omega)_j$.

With this information as input, the program produces values of the density $\rho(x, y, z)$ of the particle at points of a cubic grid in a standard (x, y, z)-Cartesian coordinate system. The density is obtained by carrying out a discrete Fourier synthesis of a set of Fourier coefficients, F, at points (h, k, ℓ) of a cubic Cartesian grid; h, k, and ℓ are integers:

$$h_{\min} \le h \le h_{\max}, \qquad k_{\min} \le k \le k_{\max}, \qquad \ell_{\min} \le \ell \le \ell_{\max}.$$

A 2D discrete Fast Fourier analysis of the pixel values in a frame is carried out. The resulting coefficients are modified to account for the location, $(c_x, c_y)_j$, of the projection of the center so that the Fourier coefficients of each frame has the same origin. The resulting discrete Fourier coefficients

of the j-th frame are denoted by $F_j(U,V)$, where U and V are integers:

$$U_{\min} \le U \le U_{\max}, \qquad V_{\min} \le V \le V_{\max}.$$

Knowing the orientation, $(\theta,\phi,\Omega)_j$, of the frame, we determine the mapping (which is a rotation) from a point (U,V) on the domain of the pixel Fourier coefficients to a point (h,k,ℓ) in the 3D domain of density Fourier coefficients. We can select two of the three coordinates h, k, and ℓ to be integers.

Suppose we choose k and ℓ to be specific integers. Then, in general, the corresponding U', V', and H', which are defined by the mapping, are not integers. The known values $F_j(U,V)$, with integral U and V, and bilinear interpolation are used to obtain an estimate of $F_j(U',V')$; this is taken as the estimate of $F(H',k,\ell)$. A set of estimates of F along the line with k and ℓ fixed is obtained. This leads to an algebraic system for the Fourier coefficients, $F(h,k,\ell)$, at grid points along the line:

$$\sum_{h=h_{\min}}^{h=h_{\max}} \frac{\sin \pi(H'_j - h)}{\pi(H' - h)} F(h,k,\ell) = F_j(U'_j, V'_j) \qquad j = 1,\ldots,j_{\max}, \qquad k,\ell \quad \text{fixed.}$$

One gets such a system for the pair of integers k,ℓ for each pixel frame whose orientation results in an insertion of the frame and the line interval (H,k,ℓ), $H_{\min} \le H \le H_{\max}$. We use the singular value decomposition to find the least squares solution of the system.

Similarly, one gets systems for two other combinations: a pair of specific intergers h and ℓ and a pair of specific intergers and h and k.

When all the systems are solved, one has estimates of $F(h,k,l)$ at the points of a 3D grid.

3. Fourier Transform of Pixel Values. An excellent discussion of discrete Fourier transforms, including their relation to Fourier series and Fourier transforms, is given by Briggs and Henson [7].

The data-input includes a set of pixel frames. Each frame is a square M-by-M array of pixel values, $P_1(u,v)$, $u,v \in \{0,\ldots M-1\}$.

Here and below, we use Fortran arrays. In an array declared `DIMENSION A(0:P,-2:Q)`, the entries `A(j,k)` are ordered by columns and then by rows; e.g.:

$$(0,-2),\ (1,-2),\ \ldots,\ (P,-2),\ (0,-1),\ (1,-1),\ \ldots,\ (P,-1),\ \ldots,\ (P,Q).$$

The pixels in a frame on the data-input file are also ordered in this way.

On the data-input-file, the value stored at $(u,v) = (0,0)$ has been set equal to the zero-pixel-level of the frame and, as values are read into memory, this value is subtracted from the values on the file.

Preprocessing provides, for each frame, the location (c_u, c_v) of the projection of the center of the particle, as well as three angles, (θ,ϕ,Ω), which specify the orientation of the frame with respect to a standard Cartesian coordinate system.

The standard `FFT` routines which we use take the origin of the data to be at $(u,v) = (0,0)$ of and array declared `DIMENSION A(0:M-1,0:M-1)`. It is more convenient to have the origin at the projection of the center, (c_u, c_v). Let $([c_u],[c_v])$ denote the point, having integer coordinates, which is closest to (c_u, c_v). We partition the frame into four blocks, as indicated in Figure 1a:

$$
\begin{array}{lllll}
A: & P_1(u,v) \text{ with } & u \in \{[c_u], \ldots, M-1\} & v \in \{[c_v], \ldots, M-1\} \\
B: & P_1(u,v) \text{ with } & u \in \{\ \ 0, \ldots, [c_u]-1\} & v \in \{[c_v], \ldots, M-1\} \\
C: & P_1(u,v) \text{ with } & u \in \{\ \ 0, \ldots, [c_u]-1\} & v \in \{\ \ 0, \ldots, [c_v]-1\} \\
D: & P_1(u,v) \text{ with } & u \in \{[c_u], \ldots, M-1\} & v \in \{\ \ 0, \ldots, [c_v]-1\}
\end{array}
$$

The blocks are rearranged and put into a larger, N-by-N array with zero-fill. We take N to be even. The new arrangement, $P_2(u,v)$, $u,v \in \{0, \ldots, N-1\}$, is indicated in Figure 1b, where

$$
\begin{array}{lllll}
A: & P_2(u,v) \text{ with } & u \in \{\ \ 0, \ldots, \ \ M-[c_u]\} & v \in \{\ \ 0, \ldots, M-[c_v]\} \\
B: & P_2(u,v) \text{ with } & u \in \{N-[c_u], \ldots, N-1\} & v \in \{\ \ 0, \ldots, M-[c_u]\} \\
C: & P_2(u,v) \text{ with } & u \in \{N-[c_u], \ldots, N-1\} & v \in \{N-[c_v], \ldots, N-1\} \\
D: & P_2(u,v) \text{ with } & u \in \{\ \ 0, \ldots, M-[c_u]\} & v \in \{N-[c_v], \ldots, N-1\}
\end{array}
$$

We use the FFT routine VRFFTF from VFFTPK [36] to find the Fourier coefficients of a row of (real) pixel values, $v \in \{0, \ldots, N-1\}$; this is done for for each row $u \in \{0, \ldots, N-1\}$. $F_u(V)$ denotes these Fourier coefficients. Because the pixel values are real, F is conjugate symmetric: $F_u(-V)$ is equal to the complex conjugate of $F_u(V)$; thus only $F_u(V)$ for $V = 0, \ldots, N/2$ are computed and stored. Specifically:

$$
\begin{aligned}
\Re\{F_u(0)\} &= \frac{1}{\sqrt{N}} \sum_{v=0}^{N-1} P_2(u,v), & \Im\{F_u(0)\} = 0, \\
\Re\{F_u(V)\} &= \frac{1}{\sqrt{N}} \sum_{v=0}^{N-1} P_2(u,v)\, \cos(2\pi v V/N), & V = 1, 2, \ldots, N/2-1, \\
\Im\{F_u(V)\} &= \frac{1}{\sqrt{N}} \sum_{v=0}^{N-1} P_2(u,v)\, \sin(2\pi v V/N), & V = 1, 2, \ldots, N/2-1, \\
\Re\{F_u(N/2)\} &= \frac{1}{\sqrt{N}} \sum_{v=0}^{N-1} (-1)^v\, P_2(u,v), & \Im\{F_u(N/2)\} = 0.
\end{aligned}
$$

The inverse can be obtained by routine VRFFTB from VFFTPK [36]:

$$
\begin{aligned}
P_2(u,v) &= \frac{1}{\sqrt{N}} \left\{F_u(0) + (-1)^{-v} F_u(N/2)\right\} \\
&+ \frac{1}{\sqrt{N}} \sum_{V=0}^{N/2-1} \left[\Re\{F_u(V)\} \cos(2\pi v V/N) - \Im\{F_u(V)\} \sin(2\pi v V/N)\right]
\end{aligned} \tag{3.1}
$$

For each $V \in \{0, 1, \ldots, N/2\}$, we use the FFT routine CFFTF from [35] to find the complex-valued Fourier coefficients $F(U,V)$ of each column, $u \in \{0, \ldots, N-1\}$, of the complex values $F_u(V)$:

$$
F(U,V) = \sum_{u=0}^{N-1} F_u(V)\, e^{-2\pi i u U/N}.
$$

The inverse can be obtained with routine CFFTB from FFTPACK [35]:

$$F_u(V) = \frac{1}{N} \sum_{U=0}^{N-1} F(U,V)\, e^{2\pi i u U/N} \tag{3.2}$$

The discrete FFTs impose the periodicity $F(U+N,V) = F(U,V)$. Thus, instead of working with Fourier coefficients $F(U,V)$, having $U \in 0, \ldots, N-1$, we can use the coefficients with $U \in 0, \ldots, N/2$ and then complete the set of Fourier coefficients those having negative values of U:

$$F(U,V) = F(N+U,V), \qquad U = -1, -2, \ldots, -N/2+1.$$

It is convenient to have the coefficients $F(0,V)$ near the row $N/2$ in the array containing the coefficients and the computed coefficients are rearranged and put into the array declared

```
COMPLEX  F( -N/2+1:N/2, 0: N/2 )
```  (3.3)

Because of the rearrangement indicated in Figure 1, the coefficient $F(0,0)$ is the Fourier coefficient of the array of pixel values having the origin at the integer point $([c_u],[c_v])$. The translation of the origin of the transform of the pixel values from this point to (c_u, c_v) now requires translation to

$$(\gamma_u, \gamma_v) = (c_u, c_v) - ([c_u],[c_v])$$

This results in a change of phase of the Fourier coefficients. It is accomplished by multiplying rows and columns of entries in the array F by constants.

For example, because of (3.2) and the rearrangement, we have

$$F_{u+\gamma_u}(V) = \frac{1}{N} \sum_{U=-N/2+1}^{N/2} \left(F(U,V)\, e^{2\pi i \gamma_u U/N} \right)\, e^{2\pi i u U/N}.$$

If we had not rearranged the coefficients and made use of the array in (3.3), so that the values of U were still between 0 and $N-1$, then the value of $F(N-1,V)$ would be multiplied by $\exp(2\pi i \gamma_u (N-1)/N)$ which is not equal to $\exp(2\pi i \gamma_u (-1)/N)$ if γ_u is not an integer.

The same kind of change is made to translate the v-origin. This follows in a similar way from (3.1) and $F_u(-V) = F_u^*(V)$.

To carry out the translation, the entries in the array $F(U,V)$ are replaced:

$$F(U,V) \leftarrow F(U,V)\, e^{2\pi i \gamma_u U/N}\, e^{2\pi i \gamma_v V/N}, \quad U = -N/2+1, \ldots, N/2, \quad V = 0, \ldots, N/2.$$

Because the γ's are between $-1/2$ and $1/2$, the changes in the phases are between $-\pi/4$ and $\pi/4$.

4. Rotation of Pixel Frames. The pixel values are the result of projection through a particle in the direction normal to the (u,v)-plane; i.e., in the direction of w in a (u,v,w) orthogonal Cartesian coordinate system; see Figure 2a.

The input angles, (θ, ϕ, Ω), give the orientation of the pixel frame with respect to a standard (x,y,z) orthogonal Cartesian coordinate system. They specify a rotation, R, that takes a point

$\mathbf{u} = (u, v, w)^T$ (where T denotes transpose) to a point $\mathbf{x} = (x, y, z)^T$; R is a 3-by-3 matrix and $R\mathbf{u} = \mathbf{x}$.

θ is the angle of a right-hand rotation about the original v-axis — e.g., $\theta = 90°$ maps the original w and u axes onto the original u and negative w axes, respectively — see Figure 2b. This rotation can be represented by the matrix

$$R_\theta = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

ϕ is the angle of a right-hand rotation about the original w-axis; see Figure 2c. This rotation can be represented by the matrix

$$R_\phi = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

The rotation through θ followed by the rotation through ϕ is represented by the product $R_\phi R_\theta$.

Ω is the angle of a right-handed rotation about the image of the original w-axis after it has been transformed by $R_\phi R_\theta$; see Figure 2d.

Denote the j-th column of $R_\phi R_\theta$ by $(R_\phi R_\theta)_j$. Since the original unit w-axis is $(0, 0, 1)^T$, after the rotations by θ and ϕ, it is the unit vector $R_\phi R_\theta (0, 0, 1)^T$; i.e., this new axis of rotation is $(R_\phi R_\theta)_3$, the third column of $R_\phi R_\theta$. The columns of the matrix are unit orthogonal vectors, and thus after the rotation by Ω the other two axes are linear combinations of the first two columns of $R_\phi R_\theta$. Consequently, the original u-axis is mapped to

$$\cos\Omega\,(R_\phi R_\theta)_1 + \sin\Omega\,(R_\phi R_\theta)_2$$

and the original v-axis is mapped to

$$-\sin\Omega\,(R_\phi R_\theta)_1 + \cos\Omega\,(R_\phi R_\theta)_2$$

The image of a vector $\mathbf{u} = (u, v, w)^T$ after the rotations of θ, ϕ and Ω is

$$u\left[\cos\Omega\,(R_\phi R_\theta)_1 + \sin\Omega\,(R_\phi R_\theta)_2\right] + v\left[-\sin\Omega\,(R_\phi R_\theta)_1 + \cos\Omega\,(R_\phi R_\theta)_2\right] + w(R_\phi R_\theta)_3$$

The matrix representation of this transformation is R, where

$$R = R_\phi R_\theta R_\Omega, \qquad R_\Omega = \begin{pmatrix} \cos\Omega & -\sin\Omega & 0 \\ \sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

and the the image $\mathbf{U}$ of a vector $\mathbf{u}$ is $\mathbf{U} = R\mathbf{u}$.

5. Rotation of the Transform Domain. The Fourier transform of the density, ρ, is given by

$$F_\rho(\mathbf{X}) = \int\int\int \rho(\mathbf{x})\, e^{-2\pi i \mathbf{X}^T\mathbf{x}}\, d\mathbf{x}$$

where

$$\mathbf{X}^T = (X, Y, Z), \qquad \mathbf{x}^T = (x, y, z), \qquad \text{and} \qquad \mathbf{X}^T\mathbf{x} = X_1x_1 + Y_1y_1 + Z_1z_1.$$

The mapping R specified by (θ, ϕ, Ω) is orthogonal: $R^{-1} = R^T$; equivalently, the product RR^T is the identity matrix I. Consequently, the change of variables $\mathbf{x} = R\mathbf{u}$ leads to

$$\mathbf{X}^T\mathbf{x} = \mathbf{X}^T I\,\mathbf{x} = \mathbf{X}^T R\,R^T\mathbf{x} = (R^T\mathbf{X})^T\,(R^T\mathbf{x}) = (R^{-1}\mathbf{X})^T\,(R^{-1}\mathbf{x}) = \mathbf{U}^T\,\mathbf{u}$$

where

$$(R^{-1}\mathbf{X})^T = \mathbf{U}^T = (U, V, W)^T \qquad \text{and} \qquad (R^{-1}\mathbf{x}) = \mathbf{u} = (u, v, w)^T$$

Because R is an orthogonal transformation, $\det(R^TR) = 1$. Hence the mapping $\mathbf{x} \to \mathbf{u}$ defined by $\mathbf{x} = R\mathbf{u}$ preserves lengths so that $d\mathbf{x} = d\mathbf{u}$. Thus we have

$$F_\rho(\mathbf{X}) = F_\rho(R\mathbf{U}) = F(U, V, W) = \int\int\int \rho(R\mathbf{u})\; e^{-2\pi i\,\mathbf{U}^T\mathbf{u}}\,d\mathbf{u}$$

By setting $W = 0$, we obtain the Fourier transform of the projection of ρ on the $(U, V, 0)$ plane. This is the 'projection theorem' (see [17] and [9]).

Consequently, the discrete Fourier transform of the pixel values (see Section 3) gives values of $F(U, V, 0)$ from experimentally observed values of the projection of the density ρ. The point $(U, V, 0)$ corresponds to (X, Y, Z) in the standard transform space according to

$$R\begin{pmatrix} U \\ V \\ 0 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}. \tag{5.1}$$

6. Relationship between Two and Three Dimensional Transforms. We want to estimate values $F_\rho(X, Y, Z)$ of the transform of ρ at points of a cubic grid:

$$X = h\Delta X,\; Y = k\Delta X,\; Z = \ell\Delta X,\; h, k \in \{0, \pm1, \pm2, \ldots, \pm H/2\},\; \ell \in \{0, 1, 2, \ldots, H/2\}$$

in a standard orthogonal Cartesian system. We use the values of the transform $F(U, V)$ of a pixel frame at points of a square grid:

$$U = p\Delta U, \quad p \in \{-N/2 + 1, \ldots, -1, 0, 1, \ldots, N/2\} \qquad V = q\Delta U, \quad q \in \{0, 1, \ldots, N/2\}.$$

Only nonnegative values of V are required because F is the transform of a real-valued function, so F has conjugate symmetry:

$$F(-U, -V) = F^*(U, V),$$

where * denotes complex conjugation.

An arbitrary point (U, V) in the plane has coordinates (X, Y, Z) in space as given by (5.1). This can be written in terms of the first two columns, $\mathbf{R}_1$ and $\mathbf{R}_2$, of the 3-by-3 matrix R:

$$R\begin{pmatrix} U \\ V \\ 0 \end{pmatrix} = U\,\mathbf{R}_1 + V\,\mathbf{R}_2 = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = X\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} + Y\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} + Z\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}. \tag{6.1}$$

We obtain three relations among points on grid lines in the 3-dimensional (X, Y, Z)-space and points in the (U, V)-plane.

Given $X = h\Delta X$, $Y = k\Delta X$: find Z', U', and V'. First we take specific integers, h, k, and the corresponding grid points $X = h\Delta X$ and $Y = k\Delta X$. Substitution into (6.1) and rearrangement yields the equation

$$U'\,\mathbf{R}_1 + V'\,\mathbf{R}_2 - Z'\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix}. \tag{6.2}$$

In general, the components U', V', and Z', of the solution are not integers; primes denote components of the solution — components without primes indicate integers. Except for very few cases, in which the set of three vectors on the left side of the equation (6.2) is linearly dependent, this equation has a unique solution which gives the coordinates of a point (U', V') in the plane and the third coordinate of the point $(h\Delta X, k\Delta X, Z')$ in space. The point (X, Y, Z') will be a point on the grid line parallel to the Z-axis containing the grid point $(h\Delta X, k\Delta X, 0)$.

It is easy to solve (6.2). Because the columns $\mathbf{R}_1$ and $\mathbf{R}_2$ are orthogonal unit vectors, multiplication by ${\mathbf{R}_1}^T$ yields

$$U' - R_{3,1}Z' = R_{1,1}h\Delta X + R_{1,2}k\Delta X; \tag{6.3a}$$

where $R_{j,k}$ denotes the entry in the j-th row and k-th column of R. Multiplication by ${\mathbf{R}_2}^T$ yields

$$V' - R_{3,2}Z' = R_{2,1}h\Delta X + R_{2,2}k\Delta X. \tag{6.3b}$$

The third equation of (6.2) is

$$R_{3,1}U' + R_{3,2}V' - Z' = 0. \tag{6.3c}$$

Elimination of U' and V' from (6.3c) yields

$$(1 - {R_{3,1}}^2 - {R_{3,2}}^2)Z' = (R_{3,1}R_{1,1} + R_{3,2}R_{1,2})\,h\,\Delta X + (R_{3,1}R_{2,1} + R_{3,2}R_{2,2})\,k\,\Delta X. \tag{6.4}$$

If the magnitude of the coefficient of Z in (6.4) is less than a preassigned cut-off value, then this point is discarded; otherwise, Z' is obtained from (6.4) and used in (6.3a) and (6.3b) to obtain values for U' and V'.

In most cases, (U', V') is not a grid point, $(j\Delta U, k\Delta U)$, in the plane. Let U_N, U_S denote the two grid points on the 'vertical' axis nearest to U and let V_W, V_E denote grid points on the 'horizontal' axis nearest to V. For the value of $F(U', V')$, we take the value of the bilinear interpolant to F at the four nearest-neighbor grid points (see Figure 3):

$$\begin{aligned} F(U', V') \;=\; & \Big[\Big\{F(U_S, V_E)\,[U - U_N] - F(U_N, V_E)\,[U - U_S]\Big\}\,[V - V_W] \\ & + \Big\{F(U_N, V_W)\,[U - U_S] - F(U_S, V_W)\,[U - U_N]\Big\}\,[V - V_E]\Big]\,/\Delta U^2 \end{aligned}$$

The values of $F(U,V)$ are not stored for negative V. Whenever the value of $F(U,V)$ with negative V is required, we make use of the symmetry relation $F(U,-V) = F^*(-U,V)$.

Given $X = h\Delta X$, $Z = \ell\Delta X$: find Y', U', and V'. Second, we take specific integers, h, ℓ, and the corresponding grid points $X = h\Delta X$ and $Z = \ell\Delta X$. As above, substitution into (6.1), rearrangement, and solution yields an equation for Y':

$$(6.4') \qquad (1 - {R_{2,1}}^2 - {R_{2,2}}^2)Y' = (R_{2,1}R_{1,1} + R_{2,2}R_{1,2})\,h\,\Delta X + (R_{2,1}R_{3,1} + R_{2,2}R_{3,2})\,\ell\,\Delta X$$

Provided the magnitude of the coefficient of Y' is not too small, this equation is solved for Y' and the result is used in the following equations to obtain corresponding values of U' and V':

$$(6.3a') \qquad U' = R_{1,1}h\Delta X + R_{2,1}Y' + R_{3,1}\ell\Delta X;$$

$$(6.3b') \qquad V' = R_{1,2}h\Delta X + R_{2,2}Y' + R_{3,2}\ell\Delta X.$$

Given $Y = k\Delta X$, $Z = \ell\Delta X$: find X', U', and V'. Third, we take specific integers, k, ℓ, and the corresponding grid points $Y = k\Delta X$ and $Z = \ell\Delta X$. As above, substitution into (6.1), rearrangement, and solution yields an equation for X:

$$(6.4'') \qquad (1 - {R_{1,1}}^2 - {R_{1,2}}^2)X' = (R_{1,1}R_{2,1} + R_{1,2}R_{2,2})\,k\,\Delta X + (R_{1,1}R_{3,1} + R_{1,2}R_{3,2})\,\ell\,\Delta X$$

Provided the magnitude of the coefficient of X' is not too small, this equation is solved for X' and the result is used in the following equations to obtain corresponding values of U and V:

$$(6.3a'') \qquad U' = R_{1,1}X' + R_{2,1}k\Delta X + R_{3,1}\ell\Delta X;$$

$$(6.3b'') \qquad V' = R_{1,2}X' + R_{2,2}k\Delta X + R_{3,2}\ell\Delta X.$$

7. Linear Systems for Transform Values. Interpolated values of the Fourier transform are used to obtain a linear system whose solution gives values of the Fourier transform of the projected density. To explain, we review some definitions and relationships.

The Fourier transform, $\mathcal{F}$, of a function, f, defined on the x-axis and its inverse are given by

$$\mathcal{F}(\omega) = \int_{-\infty}^{\infty} f(x)\,e^{-2\pi i\omega x}\,dx, \qquad f(x) = \int_{-\infty}^{\infty} \mathcal{F}(\omega)\,e^{2\pi i\omega x}\,d\omega.$$

When f has support in $-A/2 \le x \le A/2$ (i.e., f is zero outside this interval), then

$$(7.1) \qquad \mathcal{F}(\omega) = \int_{-A/2}^{A/2} f(x)\,e^{-2\pi i\omega x}\,dx.$$

The Fourier coefficients, c_h, $h =$, 0, ±1, ±2, ... , of the periodic extension of f, period A, are

$$(7.2) \qquad c_h = \frac{1}{A}\int_{-A/2}^{A/2} f(x)\,e^{-2\pi ihx/A}\,dx = \frac{1}{A}\,\mathcal{F}(h/A)$$

and

$$f(x) = \sum_{h=-\infty}^{\infty} c_h \, e^{2\pi i h x/A}. \tag{7.3}$$

It follows from (7.1)–(7.3) that for arbitrary $\omega = h'/A$

$$\begin{aligned} \mathcal{F}(h'/A) &= \int_{-A/2}^{A/2} \left(\sum_{h=-\infty}^{\infty} c_h \, e^{2\pi i h x/A} \right) e^{-2\pi i h' x/A} \, dx \\ &= \sum_{h=-\infty}^{\infty} \frac{1}{A} \mathcal{F}(h/A) \int_{-A/2}^{A/2} e^{2\pi i (h-h') x/A} \, dx \qquad (7.4) \\ &= \sum_{h=-\infty}^{\infty} \mathcal{F}(h/A) \, \frac{\sin \pi(h-h')}{\pi(h-h')}. \end{aligned}$$

This result is 'Shannon's Sampling Theorem' [33]: If a function f has support in a finite interval, $-A/2 \le x \le A/2$, then its Fourier transform $\mathcal{F}$ is completely determined by its values at a discrete set of points, h/B, $h = 0, \pm 1, \pm 2, \ldots$, for any $B \ge A$. (Alternatively: if the transform of f has bounded support, then the values of f are completely determined by its values on a discrete set.)

Set $\mathcal{F}(\omega) = \mathcal{F}(h/A) = F(h)$; then (7.4) becomes

$$F(h') = \sum_{h=-\infty}^{\infty} B_{h',h} \, F(h), \qquad B_{h',h} = \frac{\sin \pi(h'-h)}{\pi(h'-h)}.$$

The limit of $\sin x/x$ as $x \to 0$ is unity.

Similarly, for f a function on 3-dimensional space which is zero outside the cube

$$-A/2 \le x, y, z \le A/2,$$

we have

$$F(h', k', \ell') = \sum_{h=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} \sum_{\ell=-\infty}^{\infty} B_{h',h} B_{k',k} B_{\ell',\ell} \, F(h, k, \ell).$$

If the values of the Fourier transform are negligible outside the cube

$$-H/2 + 1 \le h, k, \ell \le H/2,$$

then the ∞ in the lower and upper limits on the sums can be replaced with $-H/2 + 1$ and $H/2$, respectively.

The interpolants of the transform of a pixel frame give estimates of $F(h', k, \ell)$ for a set of integer points (k, ℓ). Let $R_{k,\ell}$ denote the number of values of h' that have been obtained after processing all

the the frames, Thus, we have a set of Thus, for fixed (k,ℓ) we have values $h' = h_r$, $r = 1, \ldots, R_{k,\ell}$. This leads to a system of $R_{k,\ell}$ equations in H unknowns:

$$\sum_{h_r=-H/2+1}^{H/2} B_{h_r,h}\, F(h,k,\ell) = F(h_r,k,\ell), \qquad r = 1,\ldots,R_{k,\ell}, \qquad (k,\ell) \text{ fixed.} \tag{7.5}$$

With other choices of interpolants (see §6), we get other sets of equations similar to (7.5):

$$\sum_{k_s=-H/2+1}^{H/2} B_{k_s,k}\, F(h,k,\ell) = F(h,k_s,\ell), \qquad s = 1,\ldots,S_{h,\ell}, \qquad (h,\ell) \text{ fixed.} \tag{7.6}$$

and

$$\sum_{\ell_t=-H/2+1}^{H/2} B_{\ell_t,\ell}\, F(h,k,\ell) = F(h,k,\ell_t), \qquad t = 1,\ldots,T_{h,k}, \qquad (h,k) \text{ fixed.} \tag{7.7}$$

Thus, we have three sets of linear systems which give values of the transform, $F(h,k,\ell)$, in terms of values obtained from experimental observations. Because the values, $F(h_j,k,\ell)$, $F(h,k_s,\ell)$, and $F(h,k,\ell_t)$, on the right side are obtained from experimental observations which contain noise and interoplation error, we want to have more equations than unknowns.

We obtain the least squares solutions of these systems by using the LAPACK routine SGELS (p. 141 of [1]) which makes use of the singular value decomposition of the coefficient matrix.

8. Singular Value Decomposition. Let $\mathcal{R}^n$ denote the set of column n-vectors with real components and let $\mathcal{R}^{m\times n}$ denote the set of matrices having m rows, n columns, and real entries. The Euclidean norm $||\mathbf{x}||_2$ of $\mathbf{x} \in \mathcal{R}^n$ is the square root of the sum of the squares of the components of $\mathbf{x}$.

The singular value decomposition (SVD) of $A \in \mathcal{R}^{m\times n}$ is a factorization $A = U\,\Sigma\,V^T$ where

$$U \in \mathcal{R}^{m\times m} \text{ is orthogonal}, \qquad \Sigma \in \mathcal{R}^{m\times n} \text{ is diagonal}, \qquad V \in \mathcal{R}^{n\times n} \text{ is orthogonal.}$$

A matrix U is 'orthogonal' when its columns are unit vectors of an orthogonal coordinate system, i.e., $U^TU = I$ so that U is invertible and $U^{-1} = U^T$. The diagonal entries, $\Sigma_{j,j} = \sigma_j$, of Σ are the 'singular values' of A. They are nonnegative and ordered $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min\{m,n\}}$.

The rank or A is also given by the SVD. The rank, r, of A is the dimension of its range; that is, r is the dimension of the subspace of $\mathcal{R}^m$, formed by the set of linear combinations of the columns of A: all vectors $A\mathbf{x}$ with $\mathbf{x} \in \mathcal{R}^n$. The rank of A is equal to the number of its positive singular values. That is, if $\sigma_{\min\{m,n\}} > 0$, then $r = \min\{m,n\}$; otherwise if $\sigma_k > 0$ and $0 = \sigma_{k+1} = \ldots = \sigma_{\min\{m,n\}}$, then $r = k$.

The SVD has many uses in computation (see [19], [37], or any other book on applied numerical linear algebra). In particular, it can be used to solve the least squares problem:

$$\text{given } A \in \mathcal{R}^{m\times n} \text{ and } b \in \mathcal{R}^m, \text{ find } x \in \mathcal{R}^n \text{ which minimizes } ||\,b - A\,x\,||_2.$$

The LAPACK routine SGELS solves this least squares problem by making use of the SVD of A. It does not need to find the complete SVD and uses, therefore, fewer arithmetic operations than is required for a complete factorization (see [19] pp. 254 and 263 for operation counts).

When the rank r is less than n, then there is an $(n-r)$-dimensional hyperplane of solutions of the least squares problem. In this situation SGELS returns that solution which has the shortest length: $||x||_2$ is also minimized. Thus SGELS returns a solution of a least squares problems that has fewer equations than unknowns or one that has a 'rank deficient' matrix (the rank of A is less than n).

The user can specify when a computed singular value should be considered zero by specifying the value of an input parameter, RCOND. Then singular values computed by SGELS that are less than RCOND$\times\sigma_1$ are set equal to zero.

The Normal Equations. It is well-known that the solution of the least squares problem by means of the 'normal equations'

$$A^T A\mathbf{x} = A^T\mathbf{b},$$

is unstable and during the calculation, the effect of roundoff error can grow rapidly (see [19], §§5.3.2 and 5.3.8; see [37], Lectures 11, 18, and, for a numerical example see Lecture 19; or see any other book on applied numerical linear algebra).

Despite this, it seems popular among some researchers to set up and solve the normal equations. Moreover, it also seems to be popular to solve the normal equations by computing the inverse $(A^T A\mathbf{x})^{-1}$, and then multiplying $A^T\mathbf{b}$ by the inverse. The amount of arithmetic operations in this process is about 4 times the number of operations necessary to solve the normal equations directly by Gauss elimination and about 8 times the number when they are solved by the Cholesky method which takes advantage of the symmetry of the matrix $A^T A$.

In more detail: The numerical accuracy of a problem solved by a particular method can often be measured by the 'condition number'. For the least squares problem solved with SVD, the condition number is the ratio σ_1/σ_r. Suppose one is using REAL*4 floating point arithmetic. Then the floating point values have about 8 significant digits of accuracy. Thus, when the condition number is about $\sigma_1/\sigma_r \approx 10^8$, then the computed solution is, typically, totally contaminated by the accumulation of roundoff error.

The condition number of the least squares problem solved by use of the normal equations is $(\sigma_1/\sigma_r)^2$ which is the square of the condition number when the problem in solved with SVD. Thus, when the normal equations are used, total contamination of accuracy by roundoff error accumulation usually occurs when $\sigma_1/\sigma_r \approx 10^4$. In contrast, if this same problem were solved with SVD, one would expect the solution to have about 4 significant digits of accuracy.

Detailed discussions of condition number and roundoff error accumulation are given in [19], [37], and most books on numerical linear algebra.

For our specific application, the right side of the linear system (7.5) contains entries $F(h_r, k, \ell)$. These values are subject to errors: experimental, noise, interpolation, etc. We give the relationship between condition number and such errors:

Suppose one is solving $A\mathbf{x} = \mathbf{b}$ in which the right side contains the error $\Delta\mathbf{b}$, Then the computed

solution has error $\Delta\mathbf{x}$ and

$$\frac{||\Delta\mathbf{x}||_2}{||\mathbf{x}||_2} \leq (\text{condition number})\frac{||\Delta\mathbf{b}||_2}{||\mathbf{b}||_2}.$$

Consequently, squaring the condition number squares the amplification factor multiplying the relative error of the right side and thus the upper bound on the relative error of the solution.

Although the number of arithmetic operations to solve the least squares problem by SVD is about twice the number for solving the normal equations by Cholesky factorization (or about a quarter of the number when they are solved by inverting $A^TA\mathbf{x}$) we believe that the greater numerical stability is well worth the extra cost.

9. Parallel Processing. As many processors as available are used to carry out the calculation. Each processor carries out an FFT analysis of a specific set of pixel frames and determines the Fourier coefficients on a 3D grid by interpolation. When this is completed, different pieces of information necessary to form the linear systems (7.5) have been stored on different processors. Information has to be exchanged among processors so that a particular processor acquires all the information needed to construct that set of equations assigned to it. We outline how this takes place for the system (7.5); the processing of the other two systems, (7.6) and (7.7), is handled in the same way and at the same time.

At the beginning, and whenever a processor has finished analyzing information from a pixel frame, it reads pixel data from the next frame in the set allocated to it. The processor carries out the Fourier analysis of the pixel values and determines the intersection (h_r, k, ℓ) of the plane of transformed values and the spatial grid line through $(0, k, \ell)$, as explained in Sections 6 and 7. Whenever a processor has computed the three items

$$h_r, \qquad \Re\{F(h_r, k, \ell)\}, \qquad \Im\{F(h_r, k, \ell)\}.$$

which we call a 'triple', it stores them in an array declared

```
REAL*4  INTERP( 1:3, 1:MAX_EQNS, K_MIN:K_MAX, L_MIN:L_MAX )
```

(Actually, an array is used which includes another index, 1:3, so that triples for the systems (7.6) and (7.7) can also be stored.) The total number of triples stored for each point (k, ℓ) is also stored in an array NUM_EQNS. An outline of the calculation which follows the Fourier analysis of a pixel frame is indicated below:

```
      DO 2010 L = L_MIN, L_MAX
        DO 2000 K = K_MIN, K_MAX
              ... ... ... ...
          IF( the interval (h,K,L), H_MIN <= h <= H_MAX, intersects
     +         the plane of transformed pixel values                )  THEN
C           Find the value of the Fourier transform by interpolation
              ... ... ... ...
            NUM_EQNS(L,K) = NUM_EQNS(L,K) + 1
            INTERP( 1, NUM_H_EQNS(L,K), K, L ) = H_R
```

```
            INTERP( 2, NUM_H_EQNS(L,K), K, L ) = REAL_FT
            INTERP( 3, NUM_H_EQNS(L,K), K, L ) = AIMAG_FT
                ... ... ... ...
          END IF
 2000   CONTINUE
 2010 CONTINUE
```

After all the processors have analyzed their allotted pixel frames, data have to be exchanged. For example, processor 1 might have stored 15 triples for, say, $(k,\ell) = (9,5)$. Processor 2, which analyzes a different set of pixel frames, might have stored 25 triples with the same k and ℓ, and so on. All such information is necessary to form the linear system (7.5) for this particular (h,k). An exchange of data takes place so that all the triples associated with a specific point $(k,\ \ell)$ are put onto the same processor; the set points $(k,\ \ell)$ is distributed among the set of processors.

The processor having all of the triples for the point (k,ℓ) computes the set of coefficients

$$B_{h_r,h} = \frac{\sin \pi(h_r - h)}{\pi(h_r - h)}, \qquad r = 1, ..., R_{k,\ell}, \quad h = -H/2+1, \ldots H/2$$

of the linear system (7.5).

The matrices $A \in \mathcal{R}^{R_{k,\ell} \times H}$, $b \in \mathcal{R}^{R_{k,\ell} \times 2}$, with entries

$$A_{r,h} = B_{h_r,h-H/2}, \qquad b_{r,1} = \Re\{F(h_r,k,\ell)\}, \qquad b_{r,2} = \Im\{F(h_r,k,\ell)\}$$

are passed to the LAPACK routine SGELS which computes the least square solution of the linear system to obtain estimates of

$$\Re\{F(h,k,\ell)\} \quad \text{and} \quad \Im\{F(h,k,\ell)\}, \qquad h = -H/2+1, \ldots, H/2, \quad (k,\ell) \text{ fixed.} \tag{9.1}$$

An FFT synthesis, using routine CFFTB from FFTPACK [35], with respect to h is applied to the complex values

$$F(h,k,\ell) \qquad h = -H/2+1, \ldots, H/2, \quad (k,\ell) \text{ fixed.}$$

to obtain the complex values

$$R(x,k,\ell), \qquad x = -X/2+1, \ldots, X/2, \quad (k,\ell) \text{ fixed.}$$

These are estimates of values of the Fourier transform with respect to y and z of the density $\rho(x,y,z)$.

After all the processors complete these FFT syntheses, data again have to be exchanged among processors. This is done in order to put the the values $R(x,k,\ell)$ for all the points (k,ℓ) with x fixed onto a single processor. For example, after the exchange is completed, processor P would have the values

$$R(x,k,\ell), \quad k = -K/2+1, \ldots, K/2, \quad \ell = 0, \ldots, L/2, \qquad x = X_{1,P}, \ldots, X_{2,P}.$$

where $X_{1,P}$, $X_{2,P}$, denote the limits on x for the planes to be synthesized by processor P. We do not need values corresponding to negative ℓ because of the conjugate symmetry of the transform values.

The next step in the calculation is the FFT synthesis with respect to k; the routine CFFTB from FFTPACK [35] is used. The result is a set of complex values

$$S(x, y, \ell), \quad \ell = 0, \ldots, L/2, \qquad y = Y_{\min} \ldots, Y_{\max}, \quad x = X_{1,P} \ldots, X_{2,P}.$$

Here, S is the transform of a real valued function.

Finally, a synthesis with respect to ℓ is carried out by using routine VRFFTB from VFFTPK [36]; this results in the estimates

$$\rho(x, y, z), \quad x = -X/2 + 1, \ldots, X/2, \; y = -Y/2 + 1, \ldots, Y/2, \; z = -Z/2 + 1, \ldots, Z/2.$$

The other sets of information, associated with the linear systems (7.6) and (7.7), are processed in a way similar to that described above for the the system (7.5). Currently we average the three resulting values for $\rho(x, y, z)$ and write the average onto the data-output file.

The final results are on different processors. The arrangement allows them to be written to the data-output-file in an efficient way.

It is important to distribute the major segments of the computation among the processors in such a way to that these segments end at nearly the same time for each processor. This 'load balancing' then allows each processor to be active most of the time. The distribution of the seqments depends on the particular types of processors executing the program. We discuss features of load balancing in another report.

References

[1] 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.

[2] Axelsson-Jacobson, C., R. Guillemaud, P.-E. Danielsson, P. Grangeat, M. Defrise, and R. Clack, "Comparison of Three 3D-Reconstruction Methods from Cone-Beam Data", in [20], 3–18.

[3] Baker, T. S., I. M. Boier Martin, and D. C. Marinescu, "A Parallel Algorithm for Determining orientations of Biological Macromolecules Images by Electron Microscopy," manuscript, Dec. 1997.

[4] Barrett, H. H., "Fundamentals of the Radon Transform", in *Mathematics and Computer Science in Medical Imaging*, M. A. Viergever and A. Todd-Pokropek, edits., Springer-Verlag, 1988, 105–125.

[5] Boier Martin, I. M., D. C. Marinescu, R. E. Lynch, and T. S. Baker, "Identification of Spherical Virus Particles in Digitized Images of Entire Electron Micrographs", to appear J. Struct. Biol.

[6] Bottcher, B., S. A. Wynne, and R. A. Crowther, "Determination of the Fold of the Core Protein of Hepatitis B Virus by Electron Microscopy", Nature 386 (1997), 88–91.

[7] Briggs, W. L. and Van E. Henson, *The DFT, An Owner's Manula for the Discrete Fourier Transform*, SIAM Publ., 1995.

[8] Conway, J. F., N. Cheng, A. Zlotnich, P. T. Wingfield, S. J. Stahl, and A. C. Steven, "Visualization of a 4-helix Bundle in the Hepatitis B Virus Capsid by Cryo-electron Microscopy", Nature 386 (1997), 91–94.

[9] 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.

[10] Crowther, R. A., L. A. Amos, J. T. Finch, D. J. DeRosier, and A. Klug, "Three Dimensional Reconstructions of Spherical Viruses by Fourier Synthesis from Electron Micrographs", Nature 226 (1970), 421–425.

[11] 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.

[12] Crowther, R. A., and L. A. Amos, "Harmonic analysis of Electron Microscope Images with Rotational Symmetry," J. Mol. Biol. 60 (1971), 123–130.

[13] Crowther, R. A., L. A. Amos, and A. Klug, "Three-Dimensional Image Reconstruction Using Functional Expansion," in *Proceedings of the Fifth European Congress on Electron Microscopy*, The Institute of Physics, London and Bristol (1972), 593–597.

[14] Crowther, R. A., and A. Klug, "Three Dimensional Image Reconstructions on an Extended Field — A fast Stable Algorithm", Nature 251 (1974), 490–492.

[15] Crowther, R. A., and A. Klug, "Structural Analysis of Macromolecular Assemblies by Image Reconstruction from Electron Micrographs," Annu. Rev. Biochem. 44 (1975), 161–182.

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

[17] De Rosier, D. J., and A. Klug, "Reconstruction of Three Dimensional Structures from Electron Micrographs", Nature 217 (1968), 130–134.

[18] DeRosier, D. J., and P. B. Moore, "Reconstruction of Three-Dimensional Images from Electron Micrographs of Structures with Helical Symmetry", J. Mol. Biol. 52 (1970), 355–369.

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

[20] Grangeat, P., and J-L Amans, edits., "Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine," Kluwer Academic Publishers, 1996.

[21] Grosse, E., "Spectral Spline Approximation", in *Approximation Theory V*, **Who??**, edits., Academic Press, 1986, 363–366

[22] Hawkins, W. G., and H. H. Barret, "A Numerically Stable Circular Harmonic Reconstruction Algorithm", SIAM J. Numer. Anal. 23 (1986), 823–890.

[23] Hoppe, Von W., "Das Endlichkeitpostulat und das Interpolationstheorem der dreidimensionalen elecktronenmikroskopischen Analyse aperiodisher Struturen", Optik 29 (1969), 617–621.

[24] 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 (1994) 550–559.

[25] *The Radon Transform*, Birkhäuser, 1980.

[26] Lohner, R., "Translation of Radon's 1917 Paper", in [16], pp. 204–217.

[27] Markoe, A., "Fourier Inversion of the Attenuated X-Ray Transform", SIAM J. Math. Anal. 15 (1984), 718–722.

[28] Nievergelt, Y., "Elementary Inversion of Radon's Transform", SIAM Review 28 (1986), 79–84.

[29] Nievergelt, Y., "Exact Reconstruction Filters to Invert Radon Transforms with Finite Elements", J. Math. Anal. Appl. 120 (1986), 288–314.

[30] Radon, J., "Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten," Berichte Sächsische Akademie der Wissenschapten. Leipzig, Math.–Phys. Kl. 69 (1917), 262–267.

[31] Ramm, A. G., *The Radon Transform and local tomography*, CRC Press, 1996

[32] Robb, R. A., *Three-Dimensional Biomedical Imaging; Principles and Practice*, VCH Publishers, 1995.

[33] Shannon, C. E., "Communication in the Presence of Noise", Proc. IRE 37 (1949), 10–21.

[34] Shepp, L. A., and J. B. Kruskal, "Computerized Tomography: The New Medical X-Ray Technology", Amer. Math. Monthly 85 (1978), 420–439.

[35] Swarztrauber, P. N., FFTPACK, a Package of Fortran Subprograms for the Fast Fourier Transform of Periodic and Other Symmetric Sequences, 1985. Obtainable by e-mail or by ftp from netlib@ornl.gov.

[36] Sweet, R. A.,, L. L. Lindgren, and R. F. Boisert, VFFTPK, A Vectorized Package of Fortran Subprograms for the Fast Fourier Transform of Multiple Real Sequences, 1990. Obtainable by e-mail or by ftp from netlib@ornl.gov.

[37] Trefethen, L. N., and D. Bau, III, *Numerical Linear Algebra*, SIAM Publications, 1997.

[38] Tuy, H. K., "An Inversion Formula for Cone-Beam Reconstruction", SIAM J. Appl. Math. 43 (1983), 546–552.

[39] Udupa, J. K., and G. T. Herman, edits., *3D Imaging in Medicine*, CRC Press, 1991

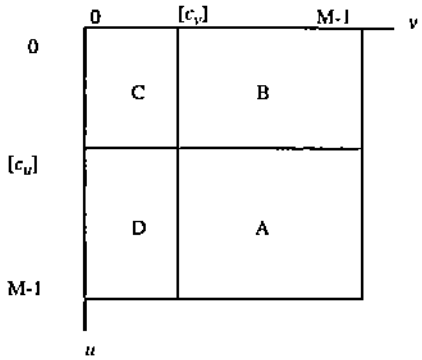

Figure 1a

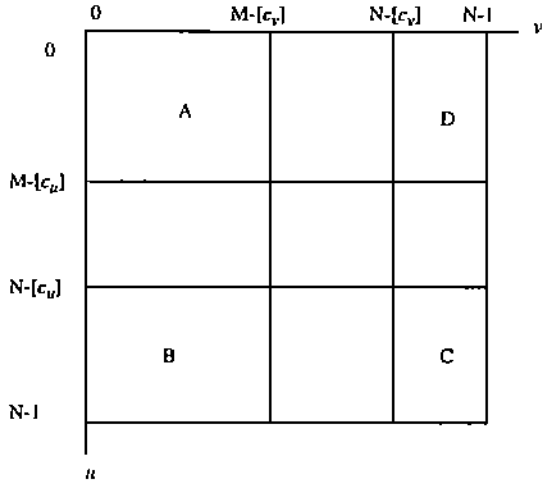

Figure 1b

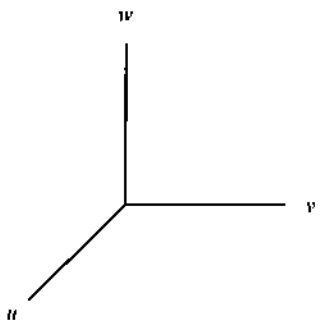

Figure 2a

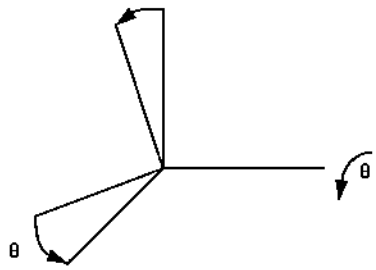

Figure 2b

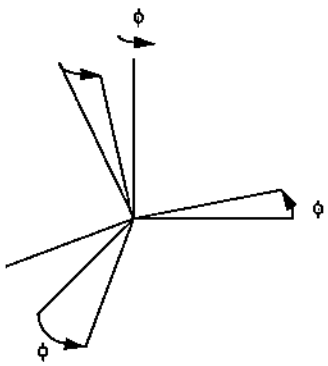

Figure 2c

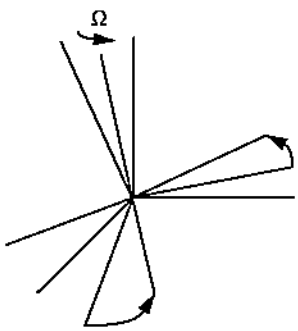

Figure 2d

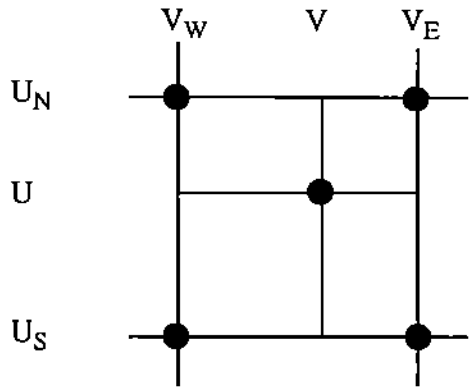

Figure 3