A fast reconstruction algorithm for electron microscope tomography

Kristian Sandberg,[a,*] David N. Mastronarde,[b] and Gregory Beylkin[a]

[a] *Department of Applied Mathematics, University of Colorado at Boulder, CO 80309, USA*
[b] *Boulder Laboratory for 3-D Electron Microscopy of Cells, Department of Molecular, Cellular, and Developmental Biology, University of Colorado at Boulder, CO 80309, USA*

Received 23 May 2003, and in revised form 3 September 2003

Abstract

We have implemented a Fast Fourier Summation algorithm for tomographic reconstruction of three-dimensional biological data sets obtained via transmission electron microscopy. We designed the fast algorithm to reproduce results obtained by the direct summation algorithm (also known as filtered or *R*-weighted backprojection). For two-dimensional images, the new algorithm scales as $O(N_\theta M \log M) + O(MN \log N)$ operations, where N_θ is the number of projection angles and $M \times N$ is the size of the reconstructed image. Three-dimensional reconstructions are constructed from sequences of two-dimensional reconstructions. We demonstrate the algorithm on real data sets. For typical sizes of data sets, the new algorithm is 1.5–2.5 times faster than using direct summation in the space domain. The speed advantage is even greater as the size of the data sets grows. The new algorithm allows us to use higher order spline interpoNation of the data without additional computational cost. The algorithm has been incorporated into a comder,6.7(lder,)02.4(re

is the number of projection angles and $M \times N$ is the size of the reconstructed image. This should be compared to the direct summation in the space domain (also known as the filtered or *R*-weighted backprojection algorithm) which scales as $O(N_\theta MN)$. Our algorithm has been applied to data of (current) typical sizes and is shown to be 1.5–2.5 times faster than the direct summation. Naturally, for larger data sets the improvement is even greater.

The problem of reconstructing an object from measured projections has a rich history and many applications. For example, X-ray tomography, radio as-scheme in the Fourier domain. A number of reconstruction algorithms based on interpolation in the Fourier domain are available in the literature, see for example, O'Sullivan (1985), Edholm and Herman (1987), Schomberg and Timmer (1995), Lanzavecchia and Bellon (1998), Waldén (2000), and Potts and Steidl (2001), and references therein.

Alternatively, fast hierarchical algorithms in the space domain have been proposed by Brandt et al. (2000) and by Basu and Bresler (2000). In these algorithms spatial interpolation is used at each level of subdivision which makes it di cult to control the resulting accuracy relative to the direct summation algorithm.

The interpolation techniques in the Fourier domain in O'Sullivan (1985), Schomberg and Timmer (1995), and Waldén (2000) use an approximation that also yields fast algorithms known as either non-equispaced fast Fourier transform (NFFT) (see Dutt and Rokhlin, 1993) or unequally spaced fast Fourier transform (USFFT) (see Beylkin, 1995). Compared to the interpolation techniques in the Fourier domain, the NFFT and USFFT algorithms use a (rigorously derived) nearly optimal relationship between the desired accuracy of the transform and the speed of the algorithm. In general, applications of NFFT or USFFT to problems of nondestructive evaluation are well understood (see e.g. Beylkin, 1995, p. 378), and can be v-1.19u

I_θ
t
$C_{t,\theta} = \{(x,z) \mid t = x\cos\theta + z\sin\theta\}$
z
t=0
g(x,z)
Specimen with density
N pixels

where *ds* denotes the standard Euclidean measure on the line. We note that evaluating $R(\theta, t)$ for all lines is equivalent to computing the Radon transform of $g(x,z)$.

Assuming that the measured intensity is described by $I(\theta, t) = e^{-R(\theta,t)}$, our goal is to approximate $g(x,z)$ by measuring $I(\theta, t)$. Sampling $R(\theta, t) = -\ln I(\theta, t)$ at a set of projection angles $\theta_1 < \cdots < \theta_l < \cdots < \theta_N$ and at a set of distances $t_0 < \cdots < t_k < \cdots < t_{M-1}$, yields the matrix

$$r_{kl} = R(\theta_l, t_k), \tag{3}$$

where $k = 0, 1, \ldots, M-1$ and $l = 1, 2, \ldots, N$. Each column l of the matrix in (3) contains all measurements for the angle θ_l.

The problem can now be formulated as follows: given the measurement data r_{kl}, find an approximation to $g(x_m, z_n)$, where the points (x_m, z_n) form a grid with $m = 1, 2, \ldots, M$ and $n = 1, 2, \ldots, N$. In TEM the total amount of input data is quite significant since such data are generated from measurements of a large number of two-dimensional slices of a specimen. Therefore, we not only need to find an accurate approximation of the density, but we also need to compute it in an efficient manner.

2.2. Inversion of the Radon transform

As is well known (see e.g. Deans, 1993) the two-dimensional density $g(x,z)$ can be recovered from the line integrals $R(\theta, t)$ in (2) as

$$g(x,z) = \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} (\phi * R_\theta)(t(\theta, x, z))\, d\theta, \tag{4}$$

where "*" denotes convolution and ϕ is a filter with the Fourier transform given by

$$\hat{\phi}(\omega) = |\omega|. \tag{5}$$

In practice, this filter is often modified by a bandlimiting window. For example, if we use

$$\hat{\phi}(\omega) = \begin{cases} |\omega|, & |\omega| \leqslant \omega_c, \\ \text{a smooth transition such that} & \\ \hat{\phi}(\pm\omega_c) = \omega_c \text{ and } \hat{\phi}(\pm\tfrac{1}{2}) = 0, & \omega_c < |\omega| \leqslant \tfrac{1}{2}, \\ 0, & |\omega| > \tfrac{1}{2}, \end{cases} \tag{6}$$

where $0 < \omega_c < \frac{1}{2}$ is a user specified parameter, then the density is approximated by

$$g(x,z) \approx \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \int_{-\frac{1}{2}}^{\frac{1}{2}} \hat{\phi}(\omega) \left(\int R(\theta, s)\, e^{2\pi i \omega s}\, ds \right) e^{-2\pi i \omega t}\, d\omega\, d\theta.$$

Here, t depends on θ, x, and z as in (1), but in what follows we may suppress this dependence in our notation.

2.3. Direct summation

If we assume that the electron beam is modeled by line integrals then reconstructing the density of a specimen from its projections can be viewed as the inversion of the Radon transform. Many reconstruction algorithms rely on this fact and solve the inversion problem by discretizing the inverse Radon transform (Deans, 1993; Gilbert, 1972). In this section, we describe the widely used direct summation algorithm (also known as the filtered or R-weighted back projection).

We discretize (4) and obtain

$$g(x_m,z_n) = \sum_{l=1}^{N_\theta} w_l (\rho * R_{\theta_l})(t(x_m,z_n)), \qquad (7)$$

where $t(x_m,z_n)$ is given by (1) and w_l are weights to compensate for unequally spaced angles. For measurements performed over equally spaced angles, the weights w_l are usually set to one. Since we have measurements only for a discrete set of values of t, the elements $(\rho * R_{\theta_l})(t(x_m,z_n))$ are estimated from $\{(\rho * R_{\theta_l})(t_k)\}_{k=0}^{M-1}$ by some interpolation scheme, usually piecewise linear interpolation. Let us summarize the steps for estimating the density $g(x,z)$ from the measurements of projections.

1. Filter the data to obtain $(\rho * R_{\theta_l})(t_k)$, $k = 0,1,\ldots,M-1$ by:
   1.1. applying the FFT along the columns of the matrix r_{kl} defined by (3),
   1.2. multiplying each element of the transformed matrix by the (pre-computed) filter coefficients and the weights w_l if necessary, and
   1.3. applying the inverse FFT column-wise.
2. Sum the result of the previous step to obtain the density by:
   2.1. computing $t(x_m,z_n)$ for each given (x_m,z_n),
   2.2. finding $(\rho * R_{\theta_l})(t(x_m,z_n))$ by linearly interpolating $(\rho * R_{\theta_l})(t_k)$, and
   2.3. summing the result according to (7).

Step 2 dominates the computational cost since we have to sum over N_θ terms $N \times M$ times, for the total computational cost of $O(N_\theta MN)$. Usually N_θ, M, and N are of the same order of magnitude so the above algorithm has the computational cost $O(N^3)$.

3. Inversion formula in the Fourier domain

In this section, we derive an inversion formula in the Fourier domain that is equivalent to the direct summation formula in (7). We will show that if

$$x_m = -x_s + m, \quad m = 1,2,\ldots,M_f,$$

where x_s is a shift parameter that depends on the selection of the coordinate system in the x-variable, then $g(x_m,z_n)$ can be written as

$$\sum_{l=1}^{N_\theta} w_l (\rho * R_{\theta_l})(t(x_m,z_n)) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \left(\sum_{l=1}^{N_\theta} v_l(\omega)\, e^{-2\pi i \omega z_n \tan\theta_l} \right) e^{2\pi i \omega x_m}\, d\omega,$$

where

$$v_l(\omega) = \frac{w_l}{\cos\theta_l} e^{-2\pi i x_s \frac{\omega}{\cos\theta_l}} \hat{\rho}\left(\frac{\omega}{\cos\theta_l}\right) \left(\frac{\sin\frac{\pi\omega}{\cos\theta_l}}{\frac{\pi\omega}{\cos\theta_l}} \right)^2 \times \sum_{k=0}^{M-1} R_{\theta_l}(t_k)\, e^{2\pi i k \frac{\omega}{\cos\theta_l}}. \qquad (8)$$

In Section 4.2, we will describe a numerical implementation of this formula that results in a fast $O(N_\theta M \log M + O(MN \log N))$ algorithm. We note that shifting the coordinate system in x by a constant x_s may be necessary to account for the deformation of a specimen during the data collection.

In order to obtain (8), we will discretize the z-variable but keep x

use interpolation to define $R_l(t)$ for any t. To incorporate the linear interpolation, let us introduce the ‘‘hat’’ function, or the linear spline

$$\beta(t) = \begin{cases} 1 - |t|, & -1 < t < 1, \\ 0, & \text{otherwise}, \end{cases} \tag{14}$$

with its Fourier transform given by

$$\hat{\beta}(\omega) = \left(\frac{\sin(\pi\omega)}{\pi\omega}\right)^2.$$

We express the piecewise linear interpolation of the discrete data using $\beta(t)$ by defining

$$R_l(t) = \sum_{k=0}^{M-1} \beta(t - k / x_s) r_{kl}. \tag{15}$$

It is easily verifie95359ds by

4.2. Numerical algorithm

Our first goal in designing the FFS algorithm is to match the results with those of the direct summation algorithm. We do it for two reasons. First, since the direct summation algorithm has been used for a long time in TEM and significant experience has been accumulated for interpretation of the images, we avoid the

$\hat{}_1(\omega_3)\quad \hat{}_1(\omega\ )\quad \hat{}_1(\omega\ )\quad \hat{}_1($

$$\vdots$$

Choosing M_f larger than M can be thought of as oversampling the image. The oversampling factor is given by the ratio M_f/M and it is desirable to make this factor as close to one as possible. For typical data sets, we have found that this oversampling factor ranges between 1.5 and 2. In (24), the size of M_f depends on

N tan $_{\max}$. We note that for physical reasons, large $_{\max\max}$

tilt angle, with the stretched data oversampled by a factor of two to minimize the filtering e ect of interpolating twice. The result is that a line of stretched input data can be added into a line of the tomographic slice by stepping through the input line at a fixed interval and with fixed interpolation factors. The advantages of FFS would have been considerably higher than described here without these improvements to the original code.

Further developments in the Tilt program and in the FFS algorithm were spurred by the desire to correct for

Tilt now uses FFS to compute slices perpendicular to the slice and ten slices, and the incremental time to compute nine slices were used to compare FFS and direct summation. Comparisons were done on SGI Octane computers with R10000 and R12000 processors, on a Sun Sparc Ultra-60, and on Intel-architecture computers with Pentium 3, Pentium 4, Athlon Thunderbird, and Athlon MP processors. For the SGI and Sun tests, programs were compiled with the native compilers; the Intel-architecture tests were done both with

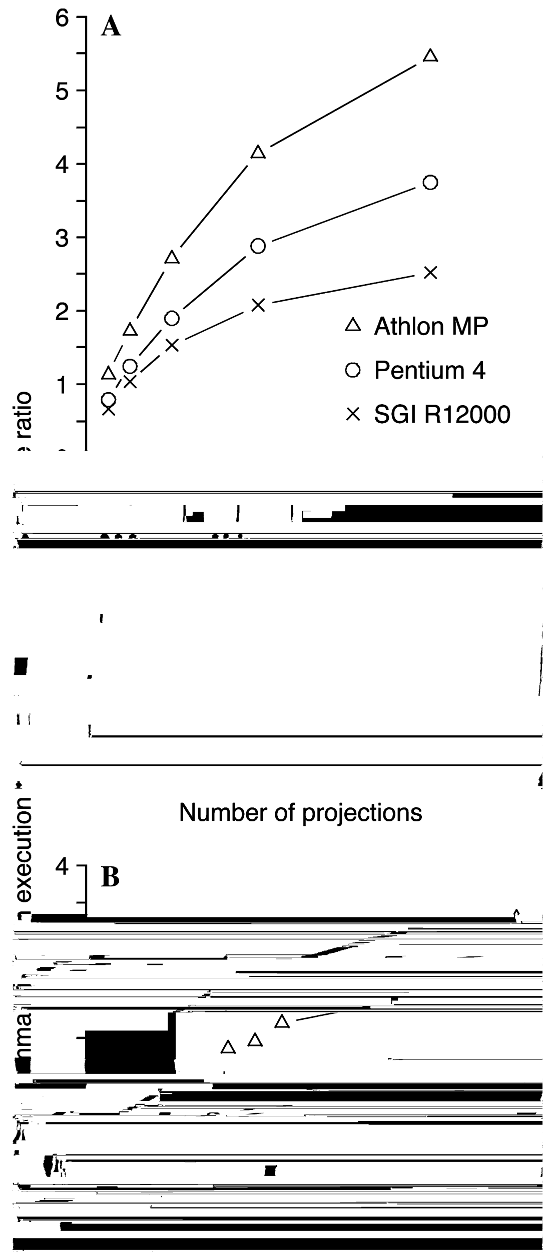

Fig. 8. Speed comparison. (Direct summation)/FFS execution time

programs compiled with GNU compilers and with Intel compilers.

The results in Fig. 8, from SGI, Pentium 4, and Athlon processors, illustrate the range of performance benefits found with FFS for typical data sizes. These graphs are not intended to demonstrate the order $O(N M \log M) + O(MN \log N)$ of the FFS algorithm since they show the dependence on each parameter separately with the other parameters fixed at typical values. The main point of Fig. 8 is to show performance gains for current typical sizes.

The strongest dependence is on the number of projections (Fig. 8A), where the speed benefit climbs 5-fold with an increase from 20 to 320 projections. This initial rise was most abrupt and pronounced with Athlon processors (e.g., Fig. 8C) and it reflects predominantly a slowing down of the direct summation per unit of computation rather than a speedup of FFS. Our interpretation is that the architecture of the Athlons is particularly favorable to the direct summation for small data sizes, but at some point a limit in cache or pipeline size is reached and the performance falls abruptly for direct summation.

The FFS is actually slower than direct summation for small data sets (Figs. 8A and B). To avoid using a slower algorithm, the Tilt program switches to direct summation when the width, thickness, or number of projections falls below a specified limit for the given computer architecture. Overall, the typical benefit from FFS is about 1.5- to 2.5-fold, with greater benefits available on some computers and with larger data sets.

Acknowledgments

This research was supported in part by DARPA Grants F49620-98-1-0491 and F30602-98-1-0154, University of Virginia Subcontract MDA-972-00-1-0016, and NSF/ITR Grant ACI-0082982 (G.B), NIH/NCRR Grant RR00592 and NIH Program Project Grant P01GM61306 to J. Richard McIntosh (D.N.M), and DARPA Grant F49620-98-1-0491, NSF/ITR Grant ACI-0082982, and NIH Program Project Grant P01GM61306 to J. Richard McIntosh (K.S.). The authors thank the anonymous reviewers and Martin Mohlenkamp for their useful and valuable comments.

Appendix A. Unequally spaced fast Fourier transform (USFFT)

As is well known, the discrete Fourier transform

$$\hat{u}_n = \sum_{k=0}^{N-1} u_k \, e^{\pm 2\pi i k \frac{n}{N}}, \quad n = 0, 1, \ldots, N-1, \qquad (A.1)$$

can be computed in $O(N \log N)$ operations using the fast Fourier transform (FFT) (Cooley and Tukey, 1965).

Since our algorithm uses the sums (23) and (19), we need a fast algorithm to compute the sums

$$\hat{u}_n = \sum_{k=1}^{M} u_k \, \mathrm{e}^{\pm 2\pi \mathrm{i} \; {}_k n}, \quad n = -\frac{N}{2}, -\frac{N}{2}+1, \ldots, \frac{N}{2}-1, \tag{A.2}$$

and

$$\hat{u}({}_k) = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} u_n \, \mathrm{e}^{\pm 2\pi \mathrm{i} n \; {}_k}, \quad k = 1, 2, \ldots, M, \tag{A.3}$$

for a given real set of points $\{{}_k\}_{k=0}^{M}$, where $|{}_k| < 1/2$ for each k. We note that M may be di erent from N. The requirement $|{}_k| < 1/2$ is not a constraint since by ap-