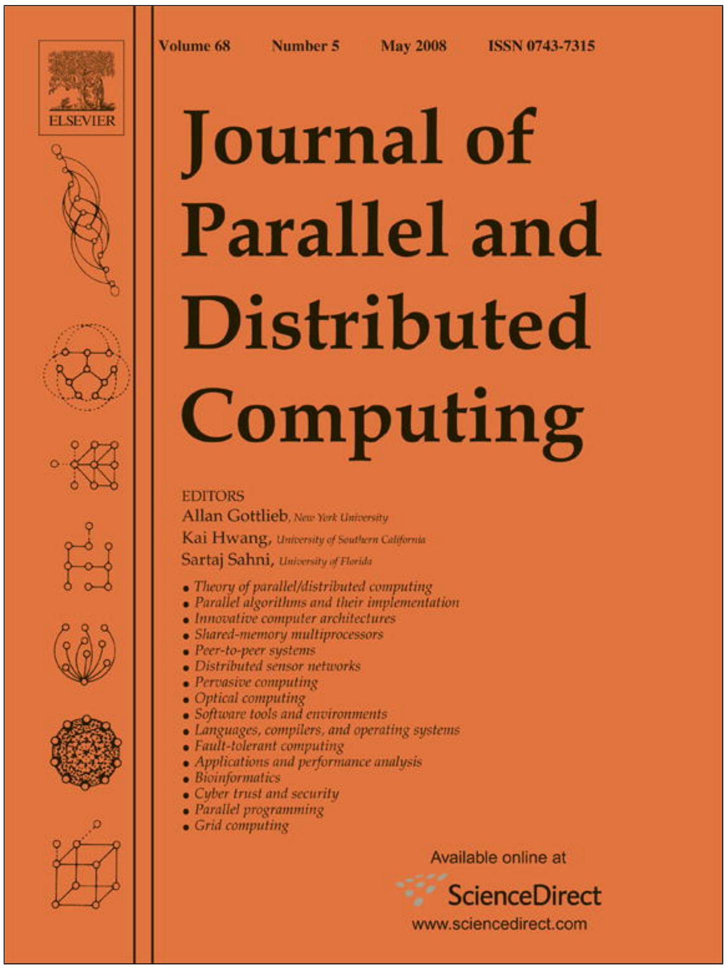
Volume 68 Number 5 May 2008 ISSN 0743-7315
ELSEVIER
Journal of Parallel and Distributed Computing
EDITORS
Allan Gottlieb, New York University
Kai Hwang, University of Southern California
Sartaj Sahni, University of Florida
• Theory of parallel/distributed computing
• Parallel algorithms and their implementation
• Innovative computer architectures
• Shared-memory multiprocessors
• Peer-to-peer systems
• Distributed sensor networks
• Pervasive computing
• Optical computing
• Software tools and environments
• Languages, compilers, and operating systems
• Fault-tolerant computing
• Applications and performance analysis
• Bioinformatics
• Cyber trust and security
• Parallel programming
• Grid computing
Available online at
ScienceDirect
www.sciencedirect.com

Available online at www.sciencedirect.com
ScienceDirect
J. Parallel Distrib. Comput. 68 (2008) 626–640
Journal of Parallel and Distributed Computing
www.elsevier.com/locate/jpdc

Efficient parallel implementation of iterative reconstruction algorithms for electron tomography

José-Jesús Fernández[a,*], Dan Gordon[b], Rachel Gordon[c]

[a]*Department of Computer Architecture and Electronics, University of Almeria, Almeria 04120, Spain*
[b]*Department of Computer Science, University of Haifa, Israel*
[c]*Department of Aerospace Engineering, The Technion, Haifa, Israel*

Received 21 August 2006; received in revised form 1 August 2007; accepted 12 September 2007
Available online 22 September 2007

Abstract

Electron tomography (ET) combines electron microscopy and the principles of tomographic imaging in order to reconstruct the three-dimensional structure of complex biological specimens at molecular resolution. Weighted back-projection (WBP) has long been the method of choice since the reconstructions are very fast. It is well known that iterative methods produce better images, but at a very costly time penalty. In this work, it is shown that efficient parallel implementations of iterative methods, based primarily on data decomposition, can speed up such methods to an extent that they become viable alternatives to WBP. Precomputation of the coefficient matrix has also turned out to be important to substantially improve the performance regardless of the number of processors used. Matrix precomputation has made it possible to speed up the block-iterative component averaging (BICAV) algorithm, which has been studied before in the context of computerized tomography (CT) and ET, by a factor of more than 3.7. Component-averaged row projections (CARP) is a recently introduced block-parallel algorithm, which was shown to be a robust method for solving sparse systems arising from partial differential equations. It is shown that this algorithm is also suitable for single-axis ET, and is advantageous over BICAV both in terms of runtime and image quality. The experiments were carried out on several datasets of ET of various sizes, using the blob model for representing the reconstructed object.
© 2007 Elsevier Inc. All rights reserved.

Keywords: BICAV; CARP; CAV; Component-averaging; Electron microscopy; Electron tomography; Image reconstruction; Parallel processing; WBP; Weighted back-projection

1. Introduction

Electron tomography (ET) combines electron microscopy and the principles of tomographic imaging to elucidate the three-dimensional (3D) structure of complex biological specimens at molecular resolution [11,23,26]. This structural information is of paramount importance to understand the cellular function [11,33]. ET has already made possible the visualization of the structure of large complex viruses, organelles and even whole cells [6,14,26,29]. Recently, ET has also been applied in physical sciences for a full 3D analysis of nanomaterials [30,37].

In ET, a set of images from a single individual specimen is acquired at different orientations by following the so-called single-axis tilt geometry. Here, the specimen is tilted over a limited range at small tilt increments, and an image of the same object area is then recorded at each tilt angle (Fig. 1). Those images represent projections, or in other words, "radiographs", of the specimen. To prevent radiation damage, images are taken at reduced electron doses, which makes signal-to-noise ratio (SNR) extremely low [23,26]. From those projection images, a 3D reconstruction can be obtained by means of tomographic reconstruction algorithms.

Proper interpretation of the structural features derived by ET require that image reconstruction introduce as little noise and artifact as possible at the spatial scales of interest. Therefore, sophisticated 3D reconstruction algorithms are needed in order to deal with the particularities of limited angle data and

* Corresponding author. Fax: +34 950 015 486.
E-mail addresses: jose@ace.ual.es (J.-J. Fernández), gordon@cs.haifa.ac.il (D. Gordon), rgordon@tx.technion.ac.il (R. Gordon).

0743-7315/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2007.09.003

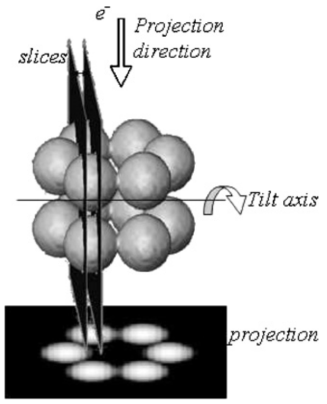

Fig. 1. Single-tilt axis data acquisition geometry. The specimen is imaged in the microscope by tilting it over a typical range of $[-60°, +60°]$ or $[-70°, +70°]$ in small tilt increments. The specimen can be considered as being composed of slices perpendicular to the tilt axis, as sketched. Reproduced from [7] with permission from Elsevier.

extremely low SNR present in ET. Despite its high sensitivity to these problems [8,27], weighted back-projection (WBP) [32] is the standard algorithm in the field mainly due to its computational simplicity. However, series expansion reconstruction methods [17] are better suited for the conditions found in ET because of the implicit regularization mechanism provided by their iterative nature and the possibility of incorporating spatial constraints [8,27]. The use of spherically symmetric volume elements (blobs) [24,25,28] makes them even more appropriate for the noise conditions in ET [8]. These methods have not been extensively used in ET due to their high computational costs, although they are lately receiving an increasing interest [23,34,37].

The need for high resolution makes ET of complex biological specimens use large projection images [7,8], which yields large reconstructed volumes after an extensive use of computational resources and considerable processing time. High performance computing has turned out to be the key to address such computational requirements and, in particular, to make the use of iterative methods affordable in ET [7,8].

This work concentrates on the efficient parallel implementation of some iterative methods using blobs as basis functions for its application in ET. We focus on parallel iterative algorithms that have recently arisen for solving large and sparse systems of linear equations, such as BICAV [3]. BICAV is the block-iterative version of CAV (component averaging) [4]; both algorithms were originally introduced for parallel image reconstruction in computerized tomography (CT), and BICAV was later also used for ET [7,8]. Another tested algorithm is the recently introduced block-parallel component-averaged row projections (CARP) [12], which was shown to provide a robust solution method for sparse systems arising from partial differential equations (PDEs). It is shown that this algorithm is also suitable for single-axis ET, and is advantageous over BICAV both in terms of runtime and image quality. These two methods are compared with the standard WBP. We present these algorithms in terms of computational performance (runtimes and speedups), as well as image quality. Results from synthetic and experimental datasets are shown and analyzed. Note that the problem domains of PDEs and ET are quite different: In PDEs, the system matrix is very sparse, with a small fixed number of elements in each row, while in ET, the number of nonzero elements is proportional to the square root of the number of variables.

In previous studies of iterative methods in ET [7,8], it was found that a single iteration of BICAV took about 10 times as long as the computation by WBP. We present two main findings. The first finding is that after efficient implementation, a single iteration of CARP takes only about 2.5 times as long as WBP when both algorithms are run on the same number of processors (regardless of the actual number of processors). Thus, the time difference between these two approaches has narrowed down considerably. Our second important finding is that it takes fewer than about 5–10 iterations to obtain reasonable-quality images with CARP, and even after one iteration, it is already better than WBP (in terms of the measured vs. estimated projection correlation). In terms of the quality of reconstruction, the iterative methods are far superior to WBP, especially under adverse conditions. CARP was found to be somewhat superior to BICAV both in terms of runtime and image quality. The consequences are that in some situations, some iterative methods—specifically CARP and BICAV—can be considered as viable alternatives to WBP. The extra time required by these methods might be compensated for by adding more processors, whose price is very minor compared to that of an electron microscope.

The rest of this paper is organized as follows. Section 2 explains the problem of image reconstruction in ET and provides an overview of the common approaches to this problem. Section 3 explains the BICAV and CARP algorithms used in our reconstructions. Section 4 provides details of the various methods of implementing BICAV and CARP for ET reconstruction. Section 5 gives the results of our experiments and a discussion, and Section 6 concludes with a summary and some future research directions.

2. Image reconstruction in electron tomography

For data acquisition in single-tilt axis ET, a specimen is placed in the electron microscope and a beam of electrons is shot towards the specimen in a direction perpendicular to an axis. Scattered and unscattered electrons emerging from the specimen are then collected by magnetic lenses and focused to form an interference pattern, which constitutes the projection image that is recorded by CCD cameras [11,15]. The specimen is then tilted around the axis, and another beam of electrons is shot. This is sketched in Fig. 1. Typically, the process is repeated over a range of $[-60°, +60°]$, or $[-70°, +70°]$, in small tilt increments of 1–2°. As a result, a so-called single-axis tilt series is obtained, which is made up of all the images acquired from the specimen at different orientations.

The reconstruction problem in ET is to obtain the 3D structure of the specimen from the set of projection images. WBP [32] is currently the standard algorithm in ET. Briefly, the method distributes the measured projection values over computed backprojection rays, i.e., the specimen mass is projected back into a reconstruction volume. When this process is repeated for a series of projection images recorded from different tilt angles, backprojection rays from the different images intersect and reinforce each other at the points where mass is found in the original structure. Therefore, the 3D mass of the specimen is reconstructed. Parallelization of WBP is straightforward because the 3D problem can be decomposed into a set of independent 2D reconstruction problems corresponding to the slices perpendicular to the tilt axis (see Fig. 1).

Series expansion reconstruction methods assume that the 3D object or function f to be reconstructed can be approximated by a linear combination of a finite set of known and fixed basis functions g_j

$$f(r, \phi_1, \phi_2) = \sum_{j=1}^{n} x_j g_j(r, \phi_1, \phi_2), \qquad (1)$$

where (r, ϕ_1, ϕ_2) are spherical coordinates, and n is the total number of the unknown variables x_j.

Our aim is to estimate the x_j's. These methods are based on an image formation model where the measurements depend linearly on the object in such a way that:

$$b_i = \sum_{j=1}^{n} a_{ij} x_j, \qquad (2)$$

where b_i denotes the ith measurement of f and a_{ij} the value of the ith projection of the jth basis function.

Under those assumptions, the image reconstruction problem can be modeled as the inverse problem of estimating the x_js by attempting to solve the system of linear equations given by Eq. (2). The problem is that (2) is inevitably not a consistent system due to a variety of reasons, such as the noise and the discretization process. Hence, there is no exact solution, and the problem has given rise to several approaches. One approach is to apply some iterative algorithm to the system, and to evaluate the sequence of iterates according to some convergence measures, and also according to the visual results of the reconstruction. One of the earliest and most successful of such approaches, introduced for CT, is widely known as algebraic reconstruction technique (ART) [16] (see Kaczmarz's algorithm in next section). ART with small relaxation parameters, has yielded excellent results. See also [17]. More recent methods were aimed towards parallel computing environments, such as the component-averaging methods CAV [4] and its block-iterative version BICAV [3], which have already been successfully tested in ET [7,8].

Another approach to dealing with the inconsistent system (2) is to seek a solution that satisfies some optimization criterion. Quadratic optimization is one such method; it seeks to minimize the L_2-norm of the residual; see [1,13,18]. Other optimization approaches are based on statistical considerations, such as maximum likelihood (ML) and expectation maximization (EM). These methods also require long reconstruction times, and this has prompted several studies on parallelization; see for example [20,22] and the references therein.

Series expansion methods have the flexibility to use basis functions to represent the volume elements g_j, which greatly influences the result of the reconstruction algorithm [24]. Spherically symmetric volume elements (blobs) with smooth transition to zero were thoroughly investigated [24,28] as alternatives to voxels for image representation, concluding that blobs are more appropriate for representing natural structures [8,27]. Blobs are defined as the generalized Kaiser-Bessel window functions [25]. They are spatially limited and, because of the analytical restrictions on the density and its derivatives, can also be considered band-limited. The shape of the blob and its spectral features are controlled by, among other parameters, the blob radius. This parameter is chosen on the basis of a tradeoff between resolution (narrower blobs) and noise suppression (wider blobs). For a detailed description of blobs, refer to [24].

The use of blob basis functions provides the series expansion methods with a further regularization mechanism, which produces inherently smooth (effectively band-limited) reconstructions. This feature makes reconstruction algorithms incorporating blobs well suited for noisy conditions, as already shown in ET [8,27] and medicine [24,28]. However, blobs involve significant additional difficulties in the parallelization of reconstruction methods in ET due to their overlapping nature [7]. In particular, the use of blobs precludes decomposing the 3D reconstruction problem into independent 2D problems, thereby needing substantial communications among the processors [7].

3. The CAV, BICAV and CARP algorithms

3.1. Background

Consider the following system of m linear equations in n variables:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad \text{for } 1 \leqslant i \leqslant m, \text{ or, in matrix form: } Ax = b. \qquad (3)$$

Throughout the rest of the paper we assume that every column of A is nonzero. Such a column, if it existed, can be removed as a preliminary step since it corresponds in effect to coefficients of a "fictitious'' variable, i.e., one whose value can be arbitrary. For $1 \leqslant i \leqslant m$, we denote by a_{i*} the ith row vector of the matrix A in Eq. (3), and by $\langle u, v \rangle$ the dot product of two vectors u, v. $\|u\|$ denotes the L_2-norm of u.

In Kaczmarz's algorithm [21] (also known as ART), which we shall denote by KACZ, each iterate is projected onto a hyperplane (defined by one of the equations) in a cyclic manner. KACZ has been studied very extensively, both theoretically and experimentally. For results related to its convergence, see [2,19,35,36].

KACZ can be described as follows: Starting from an arbitrary point x^0 in the n-dimensional Euclidean space, the kth iterate x^k is projected (orthogonally) towards the next hyperplane, and the hyperplanes are chosen in cyclic order. The following operator, called KSWP (Kaczmarz sweep), will be useful for describing KACZ and CARP (in Section 3.4).

Definition 1. Let B be a set of equations such as (3) and $\Lambda = (\lambda_1, \dots, \lambda_m)$ a vector of relaxation parameters (one for each equation). For $x \in \mathbb{R}^n$, we define the operator $\mathrm{KSWP}(B, \Lambda, x) = y^m$, where $y^0, \dots, y^m \in \mathbb{R}^n$ are obtained as follows:

```
    set $y^0 = x$.
    for $i = 1, 2, \dots, m$ do
        $y^i = y^{i-1} + \lambda_i \frac{b_i - \langle a_{i*}, y^{i-1} \rangle}{\|a_{i*}\|^2} a_{i*}$    (4)
    enddo
```

If $\lambda_1 = \cdots = \lambda_m = \lambda$ then we shall simply write λ instead of Λ in KSWP.

Algorithm 1 (*KACZ*).

```
    set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
    for $k = 0, 1, 2, \dots$ until convergence do
        $x^{k+1} = \mathrm{KSWP}(B, \Lambda, x^k)$
    enddo
end algorithm
```

Another well-known projection method is the Cimmino algorithm [5]. In contrast to the Kaczmarz algorithm, the Cimmino algorithm considers all the orthogonal projections of the current iterate, and takes the average of these projections. Assuming a fixed relaxation parameter λ, we get:

Algorithm 2 (*Cimmino*).

```
    set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
    for $k = 0, 1, 2, \dots,$ until convergence do
        $x^{k+1} = x^k + \frac{\lambda}{m} \sum_{i=1}^{m} \frac{b_i - \langle a_{i*}, x^k \rangle}{\|a_{i*}\|^2} a_{i*}$    (5)
    enddo
end algorithm
```

3.2. The CAV algorithm

Cimmino's algorithm was the starting motivational step leading to the CAV algorithm [4]. Consider the jth component in Eq. (5), and rewrite the Cimmino iterative step with respect to this component

$$x_j^{k+1} = x_j^k + \frac{\lambda}{m} \sum_{i=1}^{m} \frac{b_i - \langle a_{i*}, x^k \rangle}{\|a_{i*}\|^2} a_{ij}. \tag{6}$$

It was observed in [4] that when the system (3) is sparse, then in the sum of Eq. (6) only a "small'' number of elements a_{ij} are nonzero but the sum is divided by the "large'' m. This led to an attempt to replace the m of Eq. (6) by the number of nonzero elements in the sum, which we denote by s_j. Note that s_j is simply the number of nonzero elements of the jth column of A, and it is always positive according to our assumption on A. This led to an algorithm whose iterative step is given by the following equation

$$x_j^{k+1} = x_j^k + \frac{\lambda}{s_j} \sum_{i=1}^{m} \frac{b_i - \langle a_{i*}, x^k \rangle}{\|a_{i*}\|^2} a_{ij}, \tag{7}$$

which is identical to [4, Eq. (1.9)].

In the Cimmino algorithm, the kth iterate is an average of orthogonal projections toward the hyperplanes. However, it was shown in [12] that the replacement of m by s_j, i.e., Eq. (7), resulted in projections that are not orthogonal (unless all the s_j's are equal). The CAV algorithm [4] also made use of the s_j's, but in such a manner as to make the projection directions orthogonal. For $1 \leqslant i \leqslant m$, we define a *sparsity weight*

$$w_i = \frac{1}{\sum_{j=1}^{n} s_j (a_{ij})^2}. \tag{8}$$

The iterative step of CAV is given by

$$x^{k+1} = x^k + \lambda \sum_{i=1}^{m} w_i \left(b_i - \langle a_{i*}, x^k \rangle \right) a_{i*}. \tag{9}$$

We can see that the difference between the iterative steps of CAV and Cimmino (Eqs. (9) and (5)) lies in the different weights given to the orthogonal projections. Note that Eqs. (9) and (7) coincide when all the s_j's are equal.

We introduce the following notation in order to formally describe the CAV algorithm. This notation will also serve us in the formal description of BICAV in the next subsection.

Definition 2. Let B denote a set of linear equations, such as those of Eq. (3), w_i as defined in Eq. (8), and λ a relaxation parameter. For $x \in \mathbb{R}^n$, we define

$$\mathrm{CAV}(B, \lambda, x) = x + \lambda \sum_{i=1}^{m} w_i \left(b_i - \langle a_{i*}, x \rangle \right) a_{i*}. \tag{10}$$

Algorithm 3 (*CAV*).

```
    set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
    for $k = 0, 1, 2, \dots$ until convergence do
        $x^{k+1} = \mathrm{CAV}(B, \lambda, x^k)$
    enddo
end algorithm
```

All Cimmino-type algorithms, including CAV, can be easily implemented in parallel as follows: the equations of the system (3) are partitioned into (disjoint) blocks, and one block is assigned to every available processor. Every processor now computes the partial sum formed from the equations of its assigned block. The partial sums are then distributed and summed up, and the result is used to update the current iterate.

3.3. The BICAV algorithm

BICAV is a block-sequential (or "block-iterative'') generalization of CAV, performed as follows. The equations of the linear system (3) are divided into blocks, $B_1, \ldots, B_t$, which are not necessarily disjoint. BICAV cycles over the blocks sequentially, and within each block of equations, one step of the CAV algorithm is performed with respect only to the equations belonging to that block, meaning that the s_j's and the sparsity weights which are used within the block are local to the block. Using our notation, BICAV can be expressed as follows:

Algorithm 4 (*BICAV*).

```
    set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
    for $k = 0, 1, 2, \ldots$ until convergence do
        set $y^0 = x^k$.
        for $\ell = 1, 2, \ldots, t$ do
            $y^\ell = \mathrm{CAV}(B_\ell, \lambda, y^{\ell-1})$
        enddo
        set $x^{k+1} = y^t$.
    enddo
end algorithm
```

A detailed parallel implementation of BICAV, specific to the problem of ET, is presented in Appendix A.

3.4. The CARP algorithm

The CARP algorithm was introduced by Gordon and Gordon [12] as a method of parallelizing KACZ for partial differential equations using domain-decomposition, but without using block-projections with their inherent limitations. This led to a block-parallel method in which the domain is divided into subdomains (possibly overlapping), which can be assigned to different processors.

Starting from some initial estimate, the following two main steps are repeated until convergence: In the first step, consecutively executed row projections are performed in every block (as in KACZ) on all the equations of the block; in fact, more than one sweep can be carried out. In the second step, the results of the separate blocks are "merged'' together to form the next iterate by a "component-averaging'' operation: every variable shared by two or more blocks is replaced by the average of its values in the separate blocks. Thus, only variables bordering a neighboring domain are averaged. CARP can be applied to 3D reconstructions by assigning to each processor a "slab'' of consecutive slices. This way, in the second step, only data at the boundaries of the slabs need to be exchanged.

Formally, we proceed as follows. The equations of the linear system (3) are divided into blocks, $B_1, \ldots, B_t$, which are not necessarily disjoint. Denote $\mathscr{B} = \{B_1, \ldots, B_t\}$. For each $1 \leqslant j \leqslant n$, denote by L_j the index set of the blocks which contain an equation with a nonzero coefficient of x_j, i.e., $L_j = \{1 \leqslant \ell \leqslant t \mid x_j$ has a nonzero coefficient in some equation of $B_\ell\}$. Let $s_j = |L_j|$ (the size of L_j). In other words, s_j is the number of blocks which contain at least one equation with a nonzero coefficient of x_j. This differs from the s_j of CAV; the two definitions coincide only when each equation is taken as a (single) block. Note that since A contains no column of zeros, all the s_j's are positive. The following definition formalizes the component-averaging operation of CARP.

Definition 3. Let $\mathscr{B} = \{B_1, \ldots, B_t\}$ be as above. The component-averaging operator relative to $\mathscr{B}$ is a mapping $\mathrm{CA}_{\mathscr{B}} : (\mathbb{R}^n)^t \to (\mathbb{R}^n)$, defined as follows: Let $y^1, \ldots, y^t \in \mathbb{R}^n$. Then $\mathrm{CA}_{\mathscr{B}}(y^1, \ldots, y^t)$ is the point in $\mathbb{R}^n$ whose *j*th component is given by

$$\mathrm{CA}_{\mathscr{B}}(y^1, \ldots, y^t)_j = \frac{1}{s_j} \sum_{\ell \in L_j} y_j^\ell, \tag{11}$$

where y_j^ℓ is the *j*th component of y^ℓ, for $1 \leqslant \ell \leqslant t$.

Let $\mathscr{B}$ be as above and λ a fixed relaxation parameter. Denote by d the number of times that KSWP is repeated in each block before the averaging operations are done. CARP is the following.

Algorithm 5 (*CARP*).

```
    set $x^0 \in \mathbb{R}^n$ to an arbitrary value.
    for $k = 0, 1, 2, \ldots$ until convergence do
        for $1 \leqslant \ell \leqslant t$ parallel do
            $y^\ell = \mathrm{KSWP}^d(B_\ell, \lambda, x^k)$
        enddo
        set $x^{k+1} = \mathrm{CA}_{\mathscr{B}}(y^1, \ldots, y^t)$.
    enddo
end algorithm
```

A detailed parallel implementation of CARP is given in [12, Appendix A]. A special case of CARP, called CARP1 in [12], occurs when every equation is taken as a separate block. As mentioned, each s_j in this case is the same as in CAV, i.e., it is the number of equations in which x_j has a nonzero coefficient. It is easy to see that the iterative step of CARP1 is given by Eq. (7). CARP1 can be implemented on a multi-processor system in block-parallel mode, though the implementation-blocks are not necessarily single equations—they are simply (disjoint) blocks of equations assigned to different processors for the sake of runtime efficiency. Furthermore, the implementation-blocks of CARP1 are not intrinsic to the algorithm as in the general CARP. This implementation is detailed in [12, Appendix B].

In [4], CARP1 was examined on phantom test cases of image reconstruction from projections in transmission CT. Its initial rate of convergence was identical to that of CAV, but the images obtained were somewhat inferior to those obtained with CAV. Similar comparative results were obtained in [3] with BICAV and a block-sequential version of CARP1.

4. Implementation details

The single-tilt axis data acquisition geometry used in ET allows the use of the *Single-Program Multiple-Data* (SPMD) computational model for parallel computing. Here, all the nodes in the parallel system apply the same program to its data subdomain. The single-tilt axis geometry allows data decomposition into slabs of slices orthogonal to the tilt axis, and each node then reconstructs its own slab.

Those slabs of slices would be independent if voxel basis functions were used. However, due to their overlapping nature, the use of blobs as basis functions makes the slices, and consequently the slabs, inter-dependent. Therefore, the nodes in the parallel computer have to receive a slab composed of its corresponding subdomain together with additional redundant slices from the neighbor nodes. The number of redundant slices depends on the blob extension. Fig. 2 shows a scheme of the data decomposition, where the 2D slices are sketched by bars and the circle represents the blob extension. This data decomposition approach is used in our implementation of WBP, BICAV and CARP.

Previous implementations of BICAV in [7,8] recalculated the matrix coefficients (i.e. a_{ij} in Eq. (2)) at every iteration, for every angle and slice. The computation of the matrix coefficients is usually carried out by a footprint-based procedure, as described in [28]. Our main improvement to this implementation was to precalculate the coefficients just once and store them in memory. We refer to this mode of computation as the "large memory model'', or LM for short. Thus, BICAV-LM and CARP-LM refer to the BICAV and CARP algorithms, respectively, implemented in the LM mode.

Recall from Section 3 that BICAV partitions the set of equations into blocks, and then performs the CAV algorithm, in sequence, on each block. For this operation, we chose each block as consisting of all the equations corresponding to a fixed angle. Assuming, for example, that the angles ranged from $-60°$ to $+60°$, with an angle increment of 2°, BICAV proceeds as follows:

Step 1: Perform CAV on all the equations of $-60°$.

Step 2: Perform CAV on all the equations of $-58°$.

$\cdots\cdots$

Step 61: Perform CAV on all the equations of $+60°$.

The above sequence of steps repeats itself until convergence—see Appendix A.

One problem that was encountered with the above approach is the lack of core memory (for the matrix) with some of the very large datasets. This led us to implement an intermediate approach, in which the matrix is not precomputed in its entirety as above. Instead, before each of the above steps of BICAV, the coefficients corresponding to all the equations of a single angle are computed, and then they are exploited for all the slices in the slab during the CAV step. After the CAV step, the coefficients for the next angle are computed and stored in place of the coefficients of the completed step. We call this implementation the "small memory model'', or SM for short. BICAV-SM and CARP-SM refer to the BICAV and CARP

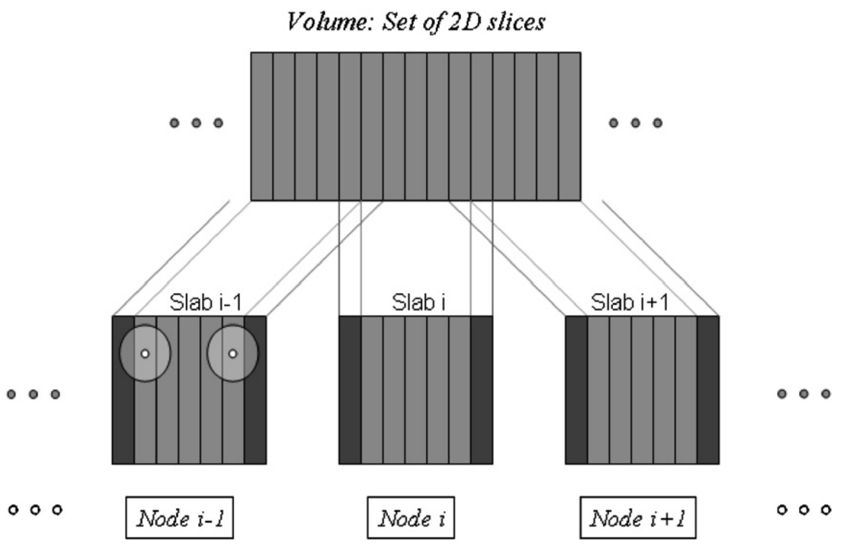

Fig. 2. Data decomposition. Every node in the parallel system receives a slab including unique slices (light gray), and additional redundant slices (dark gray) according to the blob extension. Reproduced from [7] with permission from Elsevier.

algorithms, respectively, implemented in the SM mode. As will be shown in the next section, this approach is not as efficient as the LM approach, but it is still significantly better than the original implementation of BICAV.

As described in Section 3, CARP is a block-parallel approach, meaning that the nodes of the computing system operate in parallel on the different blocks. CARP uses the same partitioning of the slices into slabs as BICAV. In each step of CARP, each node operates on its allotted slab, and then the nodes perform a component-averaging operation on the data common to two adjacent slabs. Within each slab, each computing node performs a Kaczmarz sweep on the equations; the order of the sweep depends on whether the CARP-LM or the CARP-SM mode is in effect. In the LM mode, we implemented the sweep so that it traversed the slices of the relevant slab in the outermost loop, within each slice it traversed all the angles, and for every angle, all the equations corresponding to that angle. Other orders are also possible. In the SM mode, the outer loop should be on the angles in order to utilize the available coefficients for each relevant angle.

Besides precomputation of matrix coefficients, we have also used symmetry to save time in the SM and LM modes, and also to save space in the LM mode. If the angles are distributed evenly about 0°, then there is a symmetry around the *Y*-axis. Thus, the equation coefficients for a given angle α are the same as those for $-\alpha$, but with an exchange of the slice's *X*-coordinates with $-X$. This saves computation time in both modes, and also about 50% of the memory required for the equation coefficients in the LM mode. For a given memory size, the use of this symmetry increased the scope of problems which can be handled in the LM mode.

There is another symmetry which we have not utilized in the present study, and that is the symmetry about the $X = Y$ line; i.e., for $0 \leqslant \alpha < 45°$, the coefficients for $45° + \alpha$ are the same as those for $45° - \alpha$, but with an exchange of the *X* and *Y* coordinates. Using this symmetry would save about 25% of the memory when the maximal angle is 60°, and about 36% when the maximal angle is 70°.

Finally, the reconstruction algorithms have been provided with the ability to impose spatial constraints, in the sense that a priori information about the object or the image formation process may be used to control the solution. In this work, the algorithms have the so-called non-negativity constraint incorporated. This constraint forces any negative density in the reconstruction of the current iteration to be zero.

5. Results and discussion

Tests were run on a PC cluster running under Linux. The PC cluster consists of 16 Pentium IV 2.8 GHz processors with 1 GB memory each, except for the master processor which has 3 GB memory. The processors are connected with a dedicated 1 Gb/s. Ethernet switch. The algorithms were parallelized using the standard MPI (Message Passing Interface) routines. The problems were run on the PC cluster with 1, 2, 4, 8, 12 and 16 processors. The runtimes of the algorithms were measured by running them in stand alone mode, to ensure exclusive use of the communication and processors CPU and memory.

Two classes of experiments were carried out. The first class dealt with runtime and speedup comparisons of the algorithms, which are described in Sections 5.1 and 5.2. The second class of experiments is related to the quality of reconstruction, which is described in Sections 5.3 and 5.4. For the former, synthetic datasets resembling mitochondrion data were used, as designed in [8]. For the latter, experimental datasets were used.

5.1. Runtime results

To obtain the runtimes per iteration and speedups of the algorithms, three cases of reconstruction problems with different sizes were studied

- *Problem* 1: $128 \times 128 \times 128$ blobs from 70 images of 128×128 pixels (70 projections of 128 rays and 128 slices). This problem has $m = 1,146,880$ equations and $n = 2,097,152$ variables.
- *Problem* 2: $256 \times 256 \times 256$ blobs from 70 images of 256×256 pixels. This problem has $m = 4,587,520$ equations and $n = 16,777,216$ variables.
- *Problem* 3: $480 \times 480 \times 480$ blobs from 70 images of 480×480 pixels. This problem has $m = 16,128,000$ equations and $n = 110,592,000$ variables.

In all three problems, the set of tilt angles ranged between $-70°$ and $+68°$, with 2° increment, which resembles typical experimental problems in ET. Note that the runtimes per iteration and speedups of the algorithms are independent of the specific dataset; they depend only on the data size.

Six different reconstruction algorithms/implementations were tested: WBP, CARP-LM, CARP-SM, BICAV-LM, BICAV-SM, and the original implementation of BICAV as applied to electron microscopy in [7]. For problems 1, 2 and 3, BICAV was run with 70 blocks, i.e., with block sizes of 128×128, 256×256 and 480×480 equations, respectively.

The value of the relaxation parameter was taken as $\lambda = 0.15$ for CARP and $\lambda = 1.0$ for BICAV. These values were found in preliminary optimization studies to be near optimal. It should be noted that the value of λ does not influence the runtime and speedup of the parallel implementation. It affects only the correlation and 3D image quality for a given number of iterations. The calculations were carried out for the standard blob radius $= 2.0$ [8,24,28], which involved a single neighboring slice, at both sides of each slice, affecting the calculation. All other blob parameters were taken from previous studies [7] and do not influence the performance of the parallel implementation. All our experiments were initiated with $x^0 = 0$.

Tables 1–3 present a summary of the runtimes (computation and communication runtimes) of the different algorithms and their implementations on 1, 2, 4, 8, 12 and 16 processors, for problems 1–3, respectively.

Table 1
Problem 1 (128 × 128 × 128): runtimes (in s) for one iteration

Method	1 proc.	2 proc.	4 proc.	8 proc.	12 proc.	16 proc.
CARP-LM	14.2	7.2	3.7	1.9	1.3	0.9
CARP-SM	15.9	8.5	4.6	2.6	1.9	1.6
BICAV-LM	17.7	12.8	8.5	4.1	2.7	2.1
BICAV-SM	20.7	15.8	14.6	7.6	6.1	5.6
BICAV-orig.	87.0	64.6	32.7	16.5	11.1	8.4
WBP	7.69	3.86	1.97	0.96	0.66	0.49

Table 2
Problem 2 (256 × 256 × 256): runtimes (in s) for one iteration

Method	1 proc.	2 proc.	4 proc.	8 proc.	12 proc.	16 proc.
CARP-LM	128.7	66.8	33.7	16.6	11.7	8.7
CARP-SM	150.2	80.8	41.5	23.1	15.6	12.4
BICAV-LM	156.2	82.9	43.1	22.3	15.2	11.8
BICAV-SM	168.4	90.6	50.4	32.5	26.0	21.9
BICAV-orig.	630.9	210.8	160.5	79.7	54.9	40.2
WBP	57.37	28.58	14.33	7.16	4.92	3.57

Table 3
Problem 3 (480 × 480 × 480): runtimes (in s) for one iteration

Method	1 proc.	2 proc.	4 proc.	8 proc.	12 proc.	16 proc.
CARP-LM	904.3	523.3	235.6	118.0	79.5	60.0
CARP-SM	1121.4	570.0	286.8	147.6	99.9	74.5
BICAV-LM	1097.6	544.8	277.8	140.9	95.3	76.3
BICAV-SM	1170.8	612.0	322.1	200.0	134.4	112.1
BICAV-orig.	4104.9	2060.4	1028.3	515.4	344.8	259.6
WBP	370.8	186.3	92.6	46.4	30.8	23.3

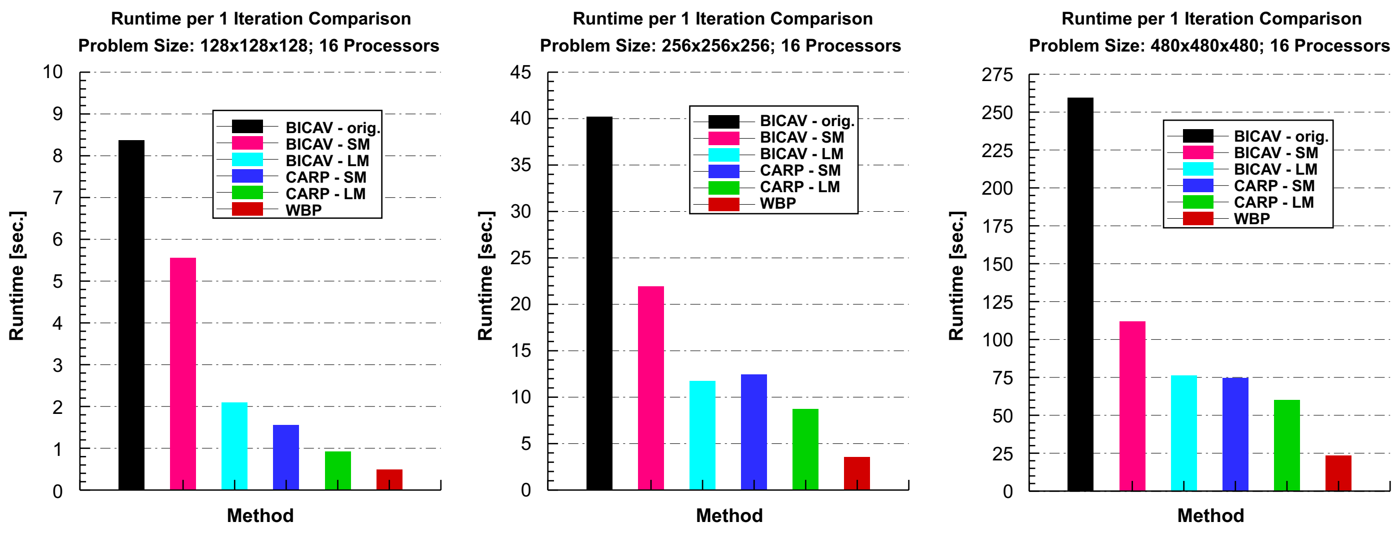

Fig. 3. Runtime for one iteration on 16 processors, for problems 1–3.

In order to help visualize the runtime distinctions between the various algorithms, Fig. 3 presents bar plots of the runtimes for one iteration of all the algorithms, for problems 1–3, on 16 processors. In addition, Fig. 4 shows how the runtimes improve as the number of processors increases, for problem 3 (without the SM algorithms).

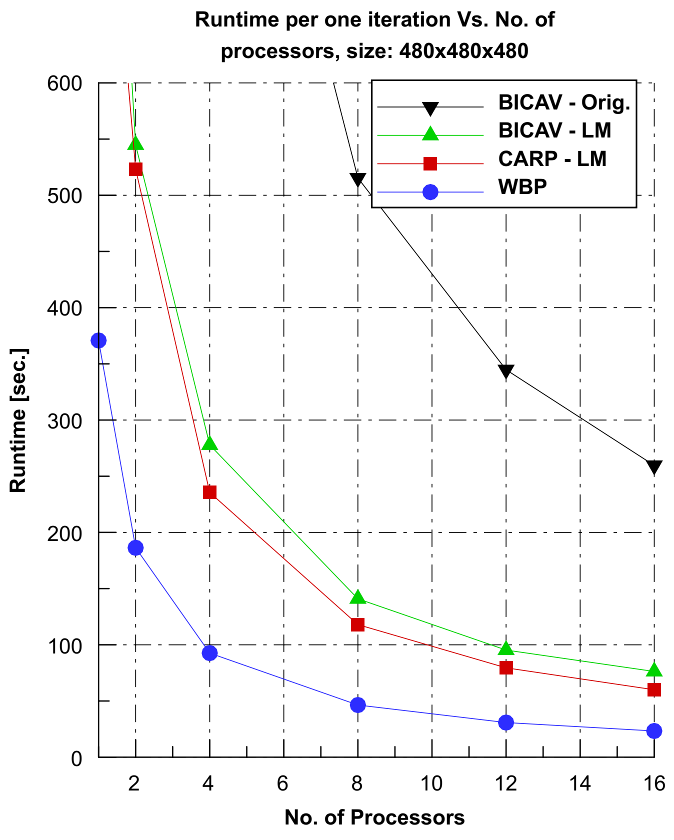

Fig. 4. Problem 3: Runtimes (per iteration) as a function of the number of processors.

The following major conclusions can be drawn from Tables 1–3 and from Figs. 3 and 4:

1. Our implementations of BICAV have decreased its runtime (per iteration) by over 70%.
2. On 16 processors, the runtime (per iteration) ratios of CARP-LM over BICAV-LM are approximately 0.6, 0.7 and 0.8, for problems 1, 2 and 3, respectively.
3. Perhaps for the first time, an iterative algorithm (CARP) can be considered as a reasonable alternative to WBP in terms of runtime, especially in view of the fact that a much superior image can be obtained—see Sections 5.3 and 5.4. A single iteration of CARP takes only about 2.5 times as long as WBP regardless of the number of processors.
4. Of the iterative algorithms, CARP-LM is generally the most efficient method on any number of processors, though in two cases it is only slightly better or comparable to CARP-SM. If the LM model cannot be implemented, then the next best choice is CARP-SM.
5. Fig. 4 shows the extent to which BICAV-LM and CARP-LM improve on the original BICAV. More importantly, it shows that as the number of processors increases, the difference between one iteration of the LM algorithms and WBP diminishes.

5.2. Speedup results

All our speedup results are based on the average runtime for one iteration. Fig. 5 presents the speedup graphs of the different methods for problems 1–3. These graphs show that WBP exhibits a linear speedup behavior, and that its speedup is almost optimal (i.e., the graph's slope is close to 1). The speedup behavior of the original BICAV implementation and of CARP-LM is almost linear in all three cases. The original BICAV is close to optimal in cases 2 and 3, while CARP-LM is close to optimal in all cases. BICAV-LM is similar to CARP-LM for the medium and large problems (2 and 3), and it is somewhat worse for the small problem (1). BICAV-SM exhibits a rather poor speedup, especially for the small size problem. We can also see that for both BICAV and CARP, the large memory version has better speedup behavior than the small memory version, on all three problems.

When comparing between various algorithms, speedup results should always be considered together with the overall runtime, because they measure the ratio of the time for one processor over the time for p processors. Even if one algorithm is much less efficient than another, its speedup ratio could be much better. The original BICAV is a case in point: its runtime is much worse than the others, but its speedup on problems 2 and 3 is very good.

5.3. Correlation comparisons

Two datasets were used in the present study to evaluate the performance of the different algorithms in terms of computation times, speedups, correlation and general 2D/3D image quality, when applied to experimental data. The two datasets are well known in the cell-biology field and their structure was solved at molecular resolution by electron microscope tomography. Different techniques were employed in their preparation, and these yielded different noise levels.

The first dataset is from Vaccinia virus (VV) (the virus that was used as a vaccine against the smallpox virus) obtained with cryo-electron tomography; see [6]. Here, the specimen is frozen in vitreous ice to preserve it during the exposure. The contrast in this dataset is poor and the SNR is low. The advantage of this method is that the specimen is imaged in its "native state'', without any deformation/artifacts.

The second dataset is that of a mitochondrion prepared with "negative staining'', see [31]. With this technique the specimen is covered with a stain to preserve it from the electron dose. Here, the contrast and the SNR are relatively good. The disadvantage of this method is the "deformations'' that the stain may introduce into the specimen.

The above datasets were used to obtain the following two additional problems:

- *Problem* 4: The VV, with a dataset of size $340 \times 340 \times 250$ blobs, with 61 images of 340×250 pixels. This problem consists of 5,185,000 equations in 28,900,000 variables. The tilt angles varied from -60° to $+60^\circ$ in increments of 2°.
- *Problem* 5: The mitochondrion (Mito) dataset of size $380 \times 380 \times 450$ blobs, with 70 images of 380×450 pixels. This problem consists of 11,970,000 equations in 64,980,000 variables. The tilt angles varied from -70° to $+68^\circ$ in increments of 2°.

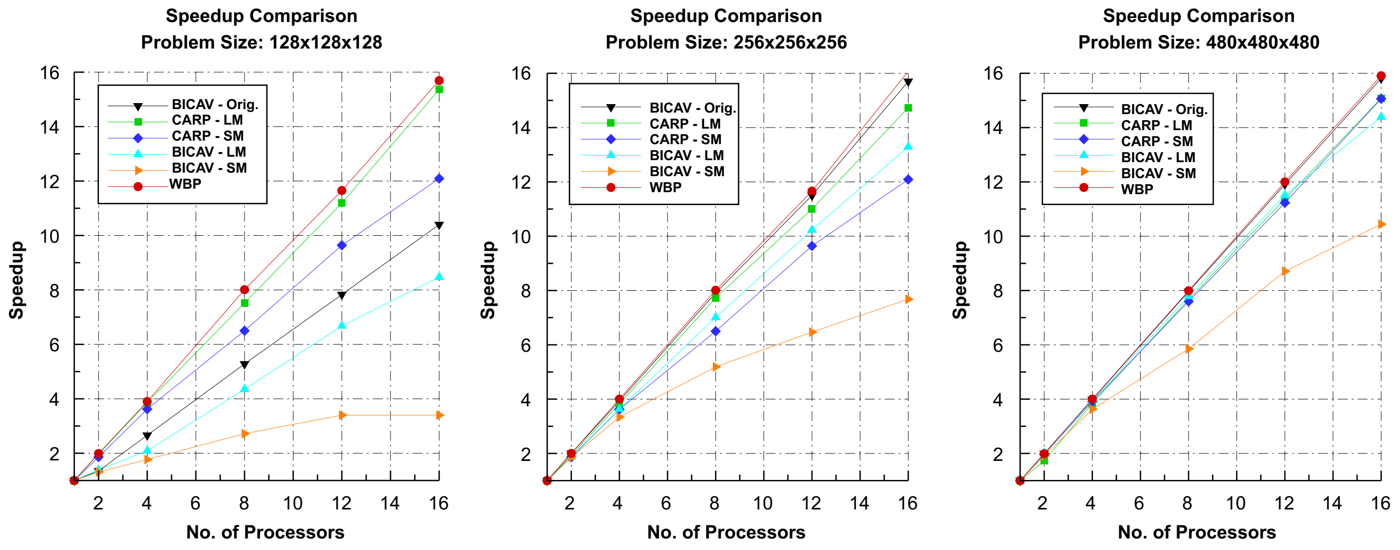

Fig. 5. Speedup graphs for problems 1–3. Results are based on the runtime for one iteration.

Table 4
Correlations achieved for the VV case

Iter	4 processors			16 processors		
	WBP	CARP-LM	BICAV-LM	WBP	CARP-LM	BICAV-LM
1	0.5394	0.5693	0.5058	0.5394	0.5777	0.5058
2	–	0.7712	0.7266	–	0.7687	0.7266
3	–	0.8128	0.7809	–	0.8110	0.7809
4	–	0.8469	0.8057	–	0.8430	0.8057
5	–	0.8625	0.8224	–	0.8582	0.8224

Table 5
Correlations achieved for the Mito case

Iter	4 processors			16 processors		
	WBP	CARP-LM	BICAV-LM	WBP	CARP-LM	BICAV-LM
1	0.6723	0.7229	0.7198	0.6723	0.7251	0.7198
2	–	0.7520	0.7567	–	0.7537	0.7567
3	–	0.7748	0.7702	–	0.7752	0.7702
4	–	0.7817	0.7765	–	0.7816	0.7765
5	–	0.7872	0.7715	–	0.7869	0.7715

To assess the quality of the reconstructions quantitatively we use the correlation measure. The correlation is computed between the projections calculated from the reconstruction and the experimental projection images taken with the microscope for the specific experimental specimen (stained specimen in the Mito case/frozen specimen for the VV case). Following the notation in Eq. (2), the correlation is computed between the set of experimental measures b_i and the calculated projections $\sum_{j=1}^{n} a_{ij}x_j$, where x_j represents the final reconstruction iterate. Here, any deformations or artifacts due to the specimen preparation or due to the imaging system are included in the experimental images to which they are compared.

Tables 4 and 5 present a comparison of the correlations of WBP, CARP-LM and BICAV-LM versus iteration number (up to 5 iterations), on 4 and 16 processors, for problems 4 and 5, respectively. Note that the iteration numbers are only relevant for CARP and BICAV. These tables show that in all cases, CARP achieves a better correlation than BICAV. Also, even after one iteration, CARP is better than WBP on both problems 4 and 5, while BICAV is worse than WBP on problem 4 and better on problem 5. The runtimes ratios of WBP over one CARP iteration, for problem 4, are 0.37 and 0.36 for 4 and 16 processors, respectively. For problem 5, the respective ratios are 0.39 and 0.42.

It can be noticed that CARP achieves slightly different correlation values when the number of processors is different. This is because CARP is actually KACZ in some superspace, and it is well known that KACZ behaves

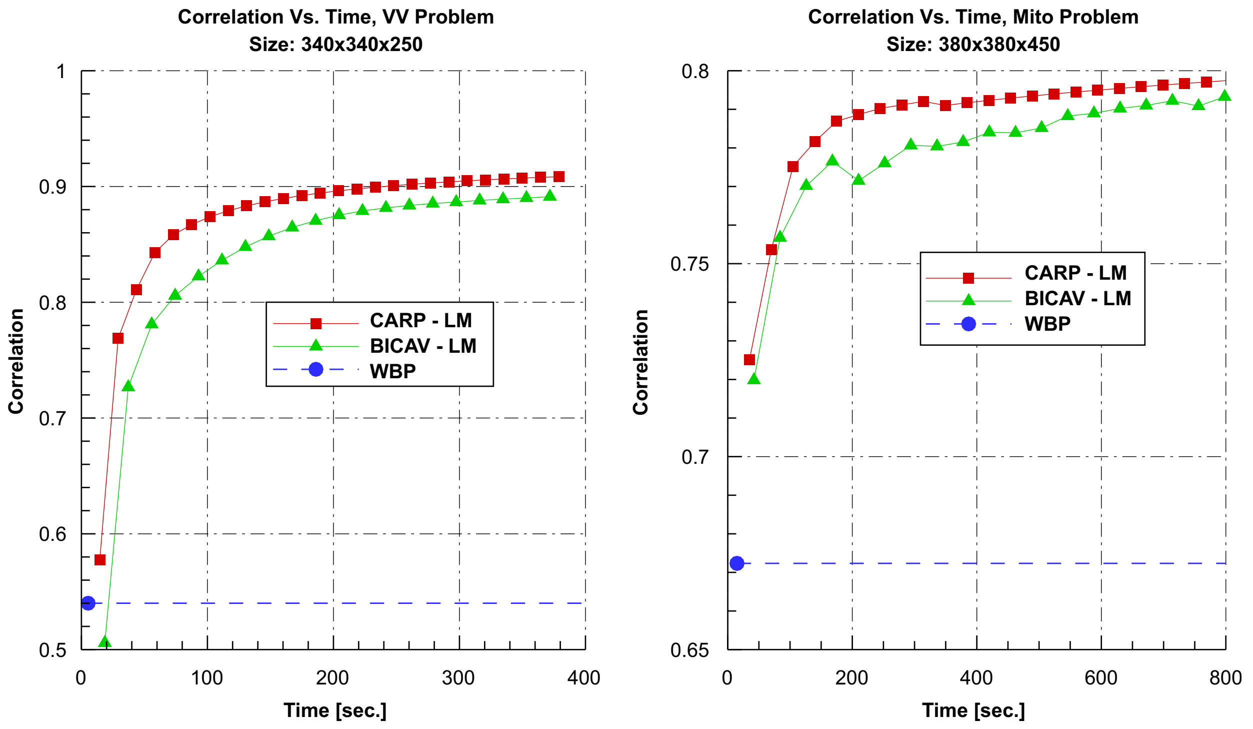

Fig. 6. Correlation vs. time on 16 processors for problems 4 (VV case) and 5 (Mito case).

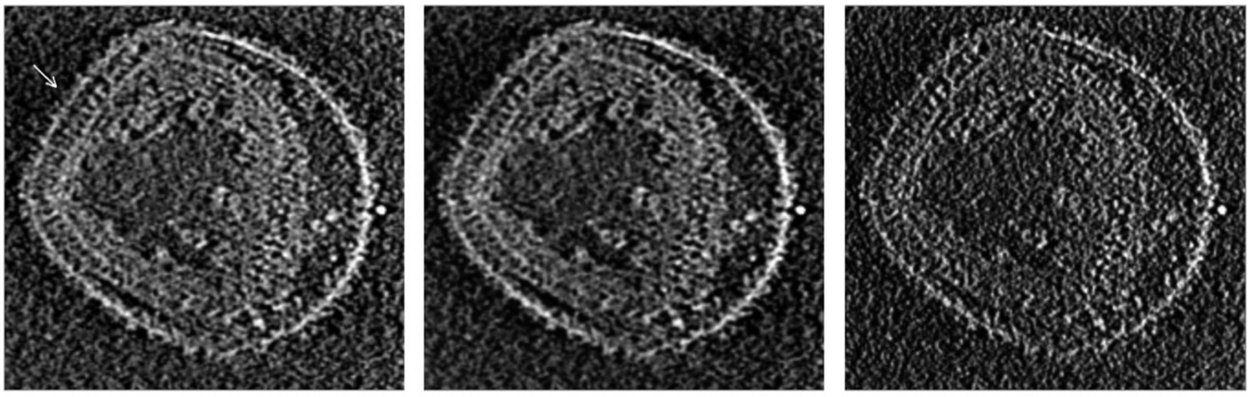

Fig. 7. An axial slice of the reconstructed VV dataset.

differently when the equations are handled in different orders.

Tables 4 and 5 do not present the actual computation time required to reach a certain correlation value. This information is shown in Fig. 6, which presents the correlation of CARP-LM, BICAV-LM and WBP versus runtime, for problems 4 and 5, on 16 processors. Just one data point is shown for WBP because it is not iterative. Both iterative methods achieve significantly better results than WBP, with CARP having a clear advantage over BICAV. This figure shows that with the VV case, if we wish to achieve a correlation value of 0.85 or higher, then BICAV needs at least twice the computation time as CARP. Similar consequences can also be drawn with the Mito case.

5.4. Visual comparisons

Figs. 7 and 8 show visual results derived from the application of CARP, BICAV and WBP to the VV and Mito datasets, respectively. In those figures, an axial slice extracted from the 3D reconstruction is shown. The results from CARP and BICAV shown here correspond to the reconstructions obtained after 5 iterations using 16 processors and the LM approaches. For the VV dataset, the correlation obtained for those reconstructions was 0.8582 for CARP, 0.8224 for BICAV and 0.5394 for WBP. The computation times that those reconstructions required were 72.9, 92.9 and 5.35 s, respectively. For the Mito dataset, the correlation was 0.7869 for CARP, 0.7715 for

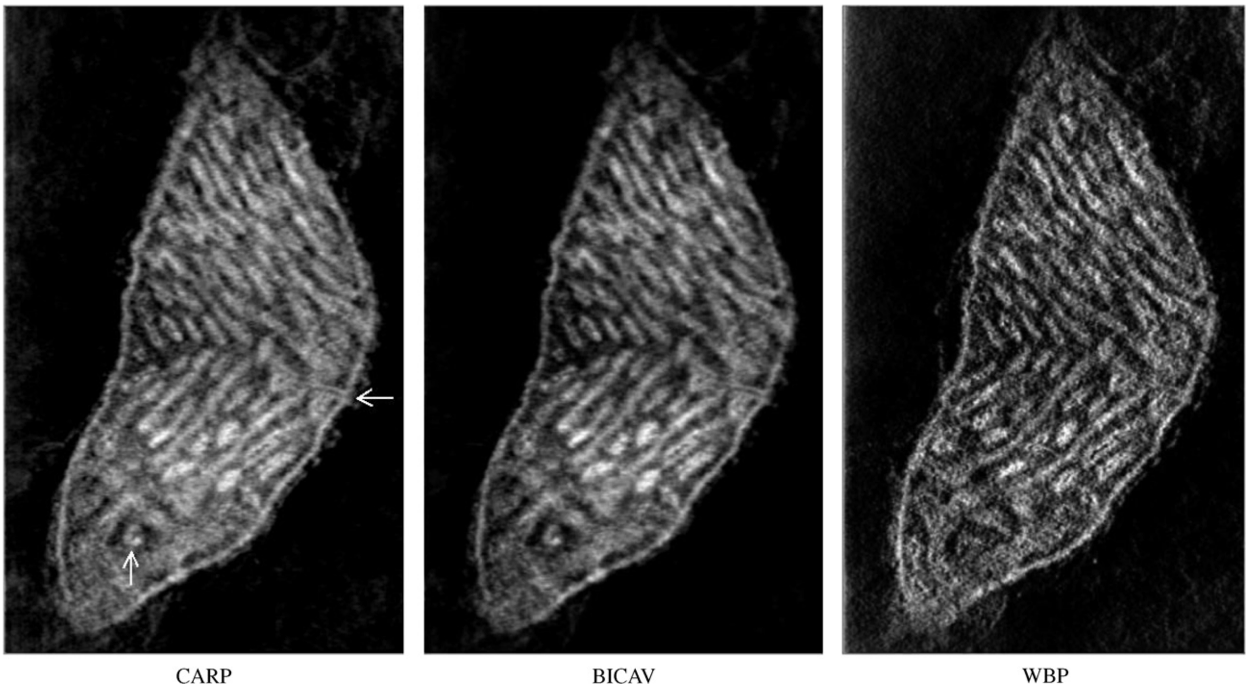

Fig. 8. An axial slice of the reconstructed Mito dataset.

BICAV and 0.6723 for WBP, and the computation times were 174.7, 210.0 and 14.8 s, respectively.

The results obtained with WBP are noisier and with lower contrast compared to those from CARP and BICAV. Some structural features appear to be smeared out by the noise and the low contrast (for instance, compare with the areas marked with arrows in the reconstruction from CARP). This effect is particularly important in the VV case due to the extremely noisy conditions derived from cryomicroscopy techniques.

CARP and BICAV are superior to WBP, yielding results that are smoother and with higher contrast. Both iterative methods obtain results with significantly better preservation of the interesting structural features, such as better continuity of the membranes both in the VV and Mito cases; the spiky inner membrane of VV and its potential embedded pores are highlighted (see arrow in Fig. 7); the junctions between the cristae and the membrane in the mitochondrion stand out (see horizontal arrow in Fig. 8); some other prominent features become visible (see vertical arrow in Fig. 8). These structural details are of paramount importance from the biological point of view. However, they are not apparent in the WBP reconstructions. Visually, the differences between BICAV and CARP are hard to discern as far as the preservation of the interesting structural features is concerned.

In these experiments, we saw that BICAV requires more iterations to achieve results of the same visual quality as CARP. These additional iterations make BICAV require at least twice the computation time as CARP. For instance, in the VV case, 9 iterations of BICAV (which involved a computation time of 167.3 s) yielded a reconstruction with approximately the same sharpness as that obtained with 5 iterations of CARP. In that example, the runtime ratio of BICAV over CARP (to yield similar results) was 2.3. There were similar experiences with the Mito case.

These experiments have also shown that only a relatively few iterations of the iterative methods are needed to obtain a good quality reconstruction. Around 5–10 iterations of CARP seem to be an appropriate number of cycles. Further iterations only involve marginal improvements in the reconstructions. For BICAV, 10–20 iterations appear to be an adequate number of iterations.

These visual results turn out to be consistent with the correlation measures shown above. CARP exhibits the best results, followed by BICAV, and finally WBP. The difference among the three methods is substantially more evident for extremely noisy conditions, such as those found in cryo-electron tomography (VV case). These results then point out that the benefits from iterative methods are better exploited under the adverse circumstances of low contrast and low signal-to-noise ratio.

Note that, in the case of CARP, thin artefactual lines may appear at the boundaries between slabs in the first iterations, but they rapidly vanish as iterations evolve. They go unnoticed or disappear completely after approximately 5 CARP iterations. An initialization of the process with backprojection makes those lines disappear earlier.

6. Conclusions and further research

This paper presented efficient implementations of two parallel reconstruction algorithms as applied to single-axis electron tomography—BICAV and CARP. The data model was based on blobs, which are superior in terms of quality to regular voxels for unregularized image reconstruction from projections. Experiments were carried out on several datasets, demonstrating that parallel iterative methods can be considered as reasonable alternatives to weighted back-projection (WBP) in terms of computational efficiency. It is already well known that iterative methods are far superior to WBP in terms of image quality.

The primary reason for improved runtime efficiency is the computation of the matrix coefficients as a preprocessing step, instead of the usual method of computing the coefficients on the fly for each equation. On our test data, this produced runtime improvements by factors ranging from 3.74 to 4.9. An intermediate approach, in which coefficients are precomputed only for every specific angle, was less efficient. Since core memory is very cheap as compared to the price of an electron microscope, the best approach is clearly the precomputation of the entire matrix.

Of the two tested iterative methods, CARP proved to be better than BICAV both in terms of computational efficiency and in terms of image quality. Depending on the problem size, CARP was from 25% to 67% faster than BICAV (per single iteration) when run on 16 processors. Furthermore, CARP achieved better correlation values with each iteration. The result of these two advantages is that (starting with iteration 4 and onward) BICAV needed at least twice the time as CARP to reach the same correlation as CARP.

There are two main reasons why CARP is more efficient than BICAV. The first one is basically similar to the reason why KACZ is more efficient than Cimmino: in KACZ, the projected values are used immediately for the next projection, while in Cimmino, in every step, all the projections are taken from the current iterate. This fundamental difference also exists between CARP and BICAV, which are derived from KACZ and Cimmino, respectively. A second reason is that in the block-parallel mode (CARP), data is exchanged only after a complete sweep of all the equations, while in the block-sequential mode (BICAV), the processors exchange information after every block. Thus, CARP requires less communication time.

Future research in this direction should concentrate on the following topics:

- Testing BICAV and CARP on raw data from electron microscopy in biological and physical sciences.
- Testing and evaluating efficient implementations of other iterative algorithms for image reconstruction in electron tomography.
- Quantitative analysis of the influence of blob parameters and relaxation parameter on BICAV and CARP.

The parallelization strategy described in this article is applicable to other problems too, namely, to those based on the solution of a PDE [12], or those related to multidimensional image processing [9,10]. In general, it is applicable to any problem where domain decomposition with neighboring relationships between immediate neighbors can be used.

Acknowledgments

Thanks are due to the Biocomputing Unit (BCU) of the Centro Nacional de Biotecnología (CNB), Madrid, Spain, for supporting the visit of the second and third authors in October 2005, when this research was initiated. Special thanks are due to Prof José-María Carazo, head of the BCU, and the staff of BCU, for many instructive discussions. The authors also wish to thank Dr J.L. Carrascosa and Dr G. Perkins for kindly providing the datasets from the Vaccinia virus and mitochondrion, respectively. This work has been partially supported by the Spanish MEC (Grant TIN2005-00447), Junta de Andalucia (Grant P06-TIC-01426) and EU (Contract LSHG-CT-2004-502828). The authors are indebted to the anonymous reviewers whose many detailed comments have helped to improve the presentation of our results.

Appendix A. Parallel implementation of BICAV for ET

1. Let t be the total number of angles. The primary partitioning of the equations into blocks for BICAV will be according to the angle corresponding to each equation. For every $0 \leqslant \alpha \leqslant t-1$, let B_α be the block of equations corresponding to angle α. Every block of equations will be processed in parallel by the different processors.
2. Let p be the number of available processing nodes, denoted $P_0, \ldots, P_{p-1}$. The set of slices is partitioned into p disjoint slabs, with each slab holding the same number (or almost the same number) of consecutive slices. The slabs are also numbered $0, \ldots, p-1$, and for $0 \leqslant r \leqslant p-1$, P_r will handle only equations belonging to slab r. The partitioning planes between the slabs pass between blob centers, but blobs close to the boundary may extend beyond their slab.
3. For every angle $0 \leqslant \alpha \leqslant t-1$ and $0 \leqslant r \leqslant p-1$, denote:
$$I_{\alpha r} = \{1 \leqslant i \leqslant m \mid \text{ equation } i \text{ corresponds to angle } \alpha \text{ and is in slab } r\},$$
$$I_r = \{1 \leqslant i \leqslant m \mid \text{ equation } i \text{ is in slab } r\} = \bigcup_{\alpha=0}^{t-1} I_{\alpha r}.$$
4. The equations are sent to the processors.
5. For $0 \leqslant r \leqslant p-1$, two sets of column indices are defined as
$$J_r^1 = \{1 \leqslant j \leqslant n \mid x_j \text{ represents a blob of slab } r\}.$$
$$J_r^2 = \{1 \leqslant j \leqslant n \mid \text{ for some } i \in I_r, a_{ij} \neq 0\}.$$
The boundary between two adjacent slabs passes through the midpoint between blob centers. J_r^1 is the set of indices of blobs whose centers lie in slab r. J_r^2 also contains all those indices, but it may also contain indices of boundary blobs of neighboring slabs whose radius extends into slab r to such an extent that they are intersected by rays of slab r. This happens when the blob radius is > 1.
6. Every processor P_r, $0 \leqslant r \leqslant p-1$, has an assigned set of "neighboring'' processors, defined as $N_r = \{0 \leqslant q \leqslant p-1 \mid q \neq r \text{ and } J_q^2 \cap J_r^2 \neq \emptyset\}$.

7. For every $0 \leqslant r \leqslant p-1$, an "expanded'' set of row indices is defined as
$$E_r = \{1 \leqslant i \leqslant m \mid \text{ for some } j \in J_r^1, a_{ij} \neq 0\}.$$
Clearly, $I_r \subseteq E_r$. Each processor P_r now retains only the equations whose index is in E_r. Note that an equation in E_r–I_r is handled by a neighboring processor of P_r, but due to the overlapping blobs, such an equation might have a nonzero coefficient corresponding to a blob of slab r.
8. Every processor P_r computes the sparsity weights relevant to its equations. Note that for $i \in I_{\alpha r}$, the weights are specific to the block α, so they are denoted by w_i^α.
9. The initial estimate $x^0 = (x_1^0, \ldots, x_n^0)$ is sent to the processors. Throughout the computation, each processor P_r holds only the variables x_j for $j \in J_r^2$.
10. Every processor sets the iteration index $k = 0$.
11.

```
for $k = 0, 1, 2, \ldots$ until convergence do
    set $y^0 = x^k$. Note: $y^\alpha$ is the iterate during the loop on $\alpha$.
    for $\alpha = 0, 1, \ldots, t-1$ do (sequential iteration on the blocks)
        for $0 \leqslant r \leqslant p-1$ parallel do
            For every $i \in I_{\alpha r}$, $P_r$ computes all the values
                $d_i = w_i^\alpha \left(b_i - \langle a_{i*}, y^\alpha \rangle\right)$.
                Note: $P_r$ has all the required information for this computation.
            For every $q \in N_r$, and $i \in I_{\alpha r}$, if $i \in E_r$, $P_r$ sends $d_i$ to $P_q$.
            For every $j \in J_r^1$, $P_r$ computes the jth component of the next $y$ iterate:
                $y_j^{\alpha+1} = y_j^\alpha + \lambda \sum_{i \in E_r} d_i a_{ij}$.
            For $j \in J_r^1$, if $j \in J_q^2$ for some $q \in N_r$, $P_r$ sends $y_j^{\alpha+1}$ to $P_q$.
        enddo (on $r$)
    enddo (on $\alpha$)
    set the next iterate: $x^{k+1} = y^t$.
enddo (on $k$)
```

References

[1] E. Artzy, T. Elfving, G.T. Herman, Quadratic optimization for image reconstruction, II, Comput. Graphics Image Process. 11 (1979) 242–261.

[2] Y. Censor, P.P.B. Eggermont, D. Gordon, Strong underrelaxation in Kaczmarz's method for inconsistent systems, Numerische Mathematik 41 (1983) 83–92.

[3] Y. Censor, D. Gordon, R. Gordon, BICAV: a block-iterative parallel algorithm for sparse systems with pixel-dependent weighting, IEEE Trans. Medical Imaging 20 (10) (2001) 1050–1060.

[4] Y. Censor, D. Gordon, R. Gordon, Component averaging: an efficient iterative parallel algorithm for large and sparse unstructured problems, Parallel Comput. 27 (6) (2001) 777–808.

[5] G. Cimmino, Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari, La Ricerca Scientifica XVI, Ser. II, Anno IX 1 (1938) 326–333.

[6] M. Cyrklaff, C. Risco, J.J. Fernández, M.V. Jimenez, M. Esteban, W. Baumeister, J.L. Carrascosa, Cryo-electron tomography of vaccinia virus, Proc. Nat. Acad. Sci. USA 102 (2005) 2772–2777.

[7] J.J. Fernández, J.M. Carazo, I. García, Three-dimensional reconstruction of cellular structures by electron microscope tomography and parallel computing, J. Parallel Distrib. Comput. 64 (2004) 285–300.

[8] J.J. Fernández, A.F. Lawrence, J. Roca, I. García, M.H. Ellisman, J.M. Carazo, High performance electron tomography of complex biological specimens, J. Structural Biol. 138 (2002) 6–20.

[9] J.J. Fernandez, S. Li, An improved algorithm for anisotropic nonlinear diffusion for denoising cryo-tomograms, J. Structural Biol. 144 (2003) 152–161.

[10] J.J. Fernandez, S. Li, Anisotropic nonlinear filtering of cellular structures in cryo-electron tomography, Comput. Sci. Engrg. 7 (5) (2005) 54–61.

[11] J.J. Fernández, C.O.S. Sorzano, R. Marabini, J.M. Carazo, Image processing and 3D reconstruction in electron microscopy, IEEE Signal Process. Mag. 23 (3) (2006) 84–94.

[12] D. Gordon, R. Gordon, Component-averaged row projections: a robust block-parallel scheme for sparse linear systems, SIAM J. Sci. Comput. 27 (2005) 1092–1117.

[13] D. Gordon, R. Mansour, A geometric approach to quadratic optimization: an improved method for solving strongly underdetermined systems in CT, Inverse Problems Sci. Engrg, 2007, in press.

[14] K. Grunewald, P. Desai, D.C. Winkler, J.B. Heymann, D.M. Belnap, W. Baumeister, A.C. Steven, Three-dimensional structure of herpes simplex virus from cryo-electron tomography, Science 302 (2003) 1396–1398.

[15] P. Hawkes, The electron microscope as a structure projector, in: J. Frank (Ed.), Electron Tomography, Plenum Press, New York, 1992, pp. 17–38.

[16] G.T. Herman, Image Reconstruction from Projections: The Fundamentals of Computerized Tomography, Academic Press, London, 1980.

[17] G.T. Herman, Algebraic reconstruction techniques in medical imaging, in: C. Leondes (Ed.), Medical Imaging. Systems, Techniques and Applications, Gordon and Breach Science, London, 1998, pp. 1–42.

[18] G.T. Herman, A. Lent, Quadratic optimization for image reconstruction, I, Comput. Graphics Image Process. 5 (1976) 319–332.

[19] G.T. Herman, A. Lent, P.H. Lutz, Relaxation methods for image reconstruction, Comm. ACM 21 (1978) 152–158.

[20] C.A. Johnson, A. Sofer, A primal-dual method for large-scale image reconstruction in emission tomography, SIAM J. Optimization 11 (3) (2001) 691–715.

[21] S. Kaczmarz, Angenäherte Auflösung von Systemen linearer Gleichungen, Bulletin de l'Académie Polonaise des Sciences et Lettres, A 35 (1937) 355–357.

[22] J.S. Kole, F.J. Beekman, Parallel statistical image reconstruction for cone-beam X-ray CT on a shared memory computation platform, Phys. Medicine Biol. 50 (2005) 1265–1272.

[23] A.P. Leis, M. Beck, M. Gruska, C. Best, R. Hegerl, W. Baumeister, J.W. Leis, Cryo-electron tomography of biological specimens, IEEE Signal Process. Mag. 23 (3) (2006) 95–103.

[24] R. Lewitt, Alternatives to voxels for image representation in iterative reconstruction algorithms, Phys. Medicine Biol. 37 (1992) 705–716.

[25] R.M. Lewitt, Multidimensional digital image representations using generalized Kaiser-Bessel window functions, J. Optical Soc. Amer. A 7 (10) (1990) 1834–1846.

[26] V. Lucic, F. Foerster, W. Baumeister, Structural studies by electron tomography: from cells to molecules, Annual Rev. Biochemistry 74 (2005) 833–865.

[27] R. Marabini, E. Rietzel, E. Schroeder, G.T. Herman, J.M. Carazo, Three-dimensional reconstruction from reduced sets of very noisy images acquired following a single-axis tilt schema: application of a new three-dimensional reconstruction algorithm and objective comparison with weighted backprojection, J. Structural Biol. 120 (1997) 363–371.
[28] S. Matej, R. Lewitt, Practical considerations for 3-D image reconstruction using spherically symmetric volume elements, IEEE Trans. Med. Imaging 15 (1996) 68–78.
[29] O. Medalia, I. Weber, A. Frangakis, D. Nicastro, G. Gerisch, W. Baumeister, Macromolecular architecture in eukaryotic cells visualized by cryoelectron tomography, Science 298 (2002) 1209–1213.
[30] P.A. Midgley, M. Weyland, 3D electron microscopy in the physical sciences: the development of z-contrast and EFTEM tomography, Ultramicroscopy 96 (2003) 413–431.
[31] G. Perkins, C. Renken, M. Martone, S. Young, M. Ellisman, T. Frey, Electron tomography of neuronal mitochondria: 3-D structure and organization of cristae and membrane contacts, J. Structural Biol. 119 (1997) 260–272.
[32] M. Rademacher, Weighted back-projection methods, in: J. Frank (Ed.), Electron Tomography, Plenum Press, New York, 1992, pp. 91–115.
[33] A. Sali, R. Glaeser, T. Earnest, W.Baumeister. From words to literature in structural proteomics, Nature 422 (2003) 216–225.
[34] R.H.M. Schoenmakers, R.A. Perquin, T.F. Fliervoet, W. Voorhout, H. Schirmacher, New software for high resolution, high throughput electron tomography,, Microscopy Analysis 108 (2005) 5–6.
[35] K. Tanabe, Projection method for solving a singular system of linear equations and its applications, Numerische Mathematik 17 (1971) 203–214.
[36] M.R. Trummer, Reconstructing pictures from projections: on the convergence of the ART algorithm with relaxation, Computing 26 (1981) 189–195.
[37] T.J.V. Yates, J.M. Thomas, J.J. Fernández, O. Terasaki, R. Ryoo, P.A. Midgley, Three-dimensional real-space crystallography of MCM-48 mesoporous silica revealed by scanning transmission electron tomography, Chem. Phys. Lett. 418 (2006) 540–543.

Jose-Jesus Fernandez received the M.Sc. and Ph.D. degrees in computer science from the University of Granada, Spain, in 1992 and 1997, respectively. He became an Assistant Processor in October 1997 and, subsequently, Associate Professor of Computer Architecture at the University of Almeria, Spain, in 2000. He has also been working at the National Center for BioTechnology (CNB), Spanish National Research Council (CSIC), Madrid, Spain and at the MRC Laboratory of Molecular Biology, Cambridge, UK. He is a member of the Supercomputing-Algorithms Research Group, an associated unit of the CNB–CSIC. Current research interests include high performance computing, image processing and tomographic reconstruction in electron microscopy.

Dan Gordon received the B.Sc. and M.Sc. degrees in mathematics from the Hebrew University of Jerusalem and the D.Sc. degree in mathematics from the Technion—Israel Institute of Technology. He is an Associate Professor of Computer Science at the University of Haifa, and has also taught at other universities in Israel and the USA. Current research interests include scientific computing, computer graphics, geometric computing and parallel computing.

Rachel Gordon (nee **Gelman**) received the B.Sc. and M.Sc. degrees in Applied Mathematics from Tel-Aviv University and the D.Sc. degree in Aerospace Engineering from the Technion—Israel Institute of Technology. Rachel is a Senior Research Fellow in the Department of Aerospace Engineering at the Technion, and has previously worked in research and development in several aerospace-related industries in Israel. Rachel has also taught at the Technion and at Texas A&M university.
Current research interests include computational fluid dynamics and numerical methods for linear systems arising from partial differential equations and biomedical image reconstruction.