Advance Access publication on 20 April 2011
doi:10.1093/comjnl/bxr033

Matrix Implementation of Simultaneous Iterative Reconstruction Technique (SIRT) on GPUs

FRANCISCO VÁZQUEZ[1], ESTER M. GARZÓN[1] AND JOSÉ JESÚS FERNÁNDEZ[2,*]

[1]*Department of Computer Architecture, University of Almeria, 04120 Almeria, Spain*
[2]*National Centre for Biotechnology, National Research Council (CSIC), Cantoblanco, 28049 Madrid, Spain*
[*]*Corresponding author: JJ.Fernandez@cnb.csic.es*

Electron tomography (ET) is an important technique in biosciences that is providing new insights into the cellular ultrastructure. Iterative reconstruction methods have been shown to be robust against the noise and limited-tilt range conditions present in ET. Nevertheless, these methods are not extensively used due to their computational demands. Instead, the simpler method weighted backprojection (WBP) remains prevalent. Recently, we have demonstrated that a matrix approach to WBP allows a significant reduction in processing time both on central processing units and on graphics processing units (GPUs). In this work, we extend that matrix approach to one of the most common iterative methods in ET, simultaneous iterative reconstruction technique (SIRT). We show that it is possible to implement this method targeted at GPU directly, using sparse algebra. We also analyse this approach on different GPU platforms and confirm that these implementations exhibit high performance. This may thus help to the widespread use of SIRT.

Keywords: tomographic reconstruction; iterative reconstruction; simultaneous iterative reconstruction technique (SIRT); graphics processing unit (GPU); high performance computing; electron tomography

Received 14 November 2010; revised 9 February 2011
Handling editor: Hong Li

1. INTRODUCTION

Tomographic reconstruction allows the elucidation of the three-dimensional (3D) structure of an object from a set of projection images taken from it by means of some imaging technique. Tomographic reconstruction is essential in many disciplines related to medicine and biology [1]. In cellular biology, in particular, electron tomography (ET) is making it possible to visualize cellular environments at an unprecedented level of detail [2, 3]. Actually, major breakthroughs have been achieved thanks to this technique [4]. To fulfil the resolution requirements, large images (typically in the range 1024×1024–4096×4096) are used and thus huge tomograms (0.5–8 gigavoxels) are computed.

Weighted backprojection (WBP) is currently the standard method in ET. However, under the noise and limited-tilt range conditions found in this field, iterative methods have turned out to be far superior [5]. Nonetheless, their computational demands have prevented their extensive use, even though high-performance computing strategies have been developed [6]. In the last few years, graphics processing units (GPUs) have revolutioned the field of ET and have played an important role to substantially accelerate these iterative methods [7, 8]. Sophisticated GPU strategies have been devised which succeed in reconstructing huge tomograms with these methods at about 1 min iteration pace [9]. However, tricky features and low-level GPU programming have been necessary to get this excellent performance.

Tomographic reconstruction can be modelled as a least square problem to be solved by matrix algorithms [1], where large sparse matrices are involved. Not long ago, the memory requirements precluded storage of the matrix coefficients, so they were recomputed on the fly. However, modern computing platforms now ship with significant amount of memory. Recently, we have presented a fresh implementation of WBP that exploits the matrix coefficients kept in memory, which remarkably accelerates the reconstruction process, both in central processing units (CPUs) and in GPUs [10]. In addition to the high speedup, one main advantage of this matrix approach is

its simplicity, in the sense that the tomographic reconstruction method is simply formulated as a set of sparse matrix vector (SpMV) products. Another advantage is the versatility, since no special low-level feature of the underlying computing platform has to be exploited.

In this work, we extend the matrix approach to iterative methods and evaluate it on different GPU platforms. Our goal is to demonstrate that it is possible to implement these methods directly using sparse algebra and that these implementations are indeed efficient and exhibit high performance. Therefore, this may contribute to extend the use of iterative reconstruction methods.

This work has been structured as follows: Section 2 presents an introduction to ET and describes the standard reconstruction method, WBP. Section 3 introduces the iterative method addressed in this work, simultaneous iterative reconstruction technique (SIRT) and our matrix approach to SIRT is then described in Section 4. The performance evaluation is carried out in Section 5.

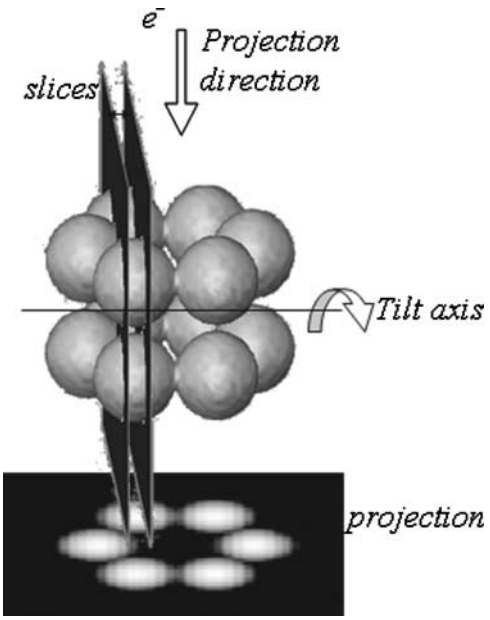

FIGURE 1. Single-tilt axis data acquisition geometry. The specimen is imaged in the microscope by tilting it over a range typically ±60° or 70° in small tilt increments. As a result, a set projection images needed to structure determination is collected. Note that the structure to be reconstructed can be considered as made up of slices perpendicular to the tilt axis, as sketched.

2. ELECTRON TOMOGRAPHY

ET is an imaging technique that is playing a major role in cellular biology [2, 11]. It allows visualization of the architecture of organelles, cells and complex viruses at close-to-molecular resolution. The level of resolution currently attainable by ET allows the identification of macromolecular assemblies and their interactions in the native cellular context. ET has made possible major breakthroughs in life sciences [4, 12–17].

In ET, the so-called single-axis tilting is the most common data collection geometry used to record the set of projection images required to compute the 3D reconstruction. For the collection of a single-axis tilt series, a single specimen is tilted over a range typically ±60° or 70° in small tilt increments (1–2°), and an image of the same object area is recorded at each tilt angle via, typically, charge-coupled device cameras (see sketch in Fig. 1). Sometimes, for better angular coverage, another tilt series is taken with the specimen rotated by 90° (the so-called double-axis tilting geometry) [18, 19]. Typical electron tomographic data sets thus have a number of images between 60 and 280. Due to the resolution requirements, the image size typically ranges from 1024 × 1024 to 4096 × 4096 pixels.

The computation of a distortion-free 3D reconstruction from a single-axis tilt series would require that a data set from the full tilt range (±90°) be available. Due to physical limitations of microscopes, the angular tilt range is limited and, as a result, tomographic tilt series have a wedge of missing data corresponding to the uncovered angular range. This limitation, which is not present in medical tomography, causes distortions in 3D reconstructions. The structural features in the 3D reconstruction become elongated and blurred along the Z direction (i.e. there is a significant loss of resolution in the Z-direction, which is the electron beam direction) [5, 20]. On the other hand, the signal-to-noise ratio in the acquired images is extremely low (in the order of 0.1) due to the use of low electron doses to reduce the radiation damage of the specimen during imaging.

Therefore, ET requires a method of '3D reconstruction from projections' to be able to deal with limited angle conditions and extremely low signal-to-noise ratios of the projection images. Currently, the standard method in the field is the well-known WBP. It assumes that projection images represent the amount of mass density encountered by imaging rays. The method simply distributes the known specimen mass present in projection images evenly over computed backprojection rays. In this way, specimen mass is projected back into a reconstruction volume (i.e. backprojected). 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 from a series of projection images. The backprojection process involves an implicit low-pass filtering that makes reconstructed volumes strongly blurred. In practice, in order to compensate for the transfer function of the backprojection process, a previous high-pass filter (i.e. weighting) is applied to the projection images, hence the term 'WBP'. For a more in-depth description of the method, refer to [21].

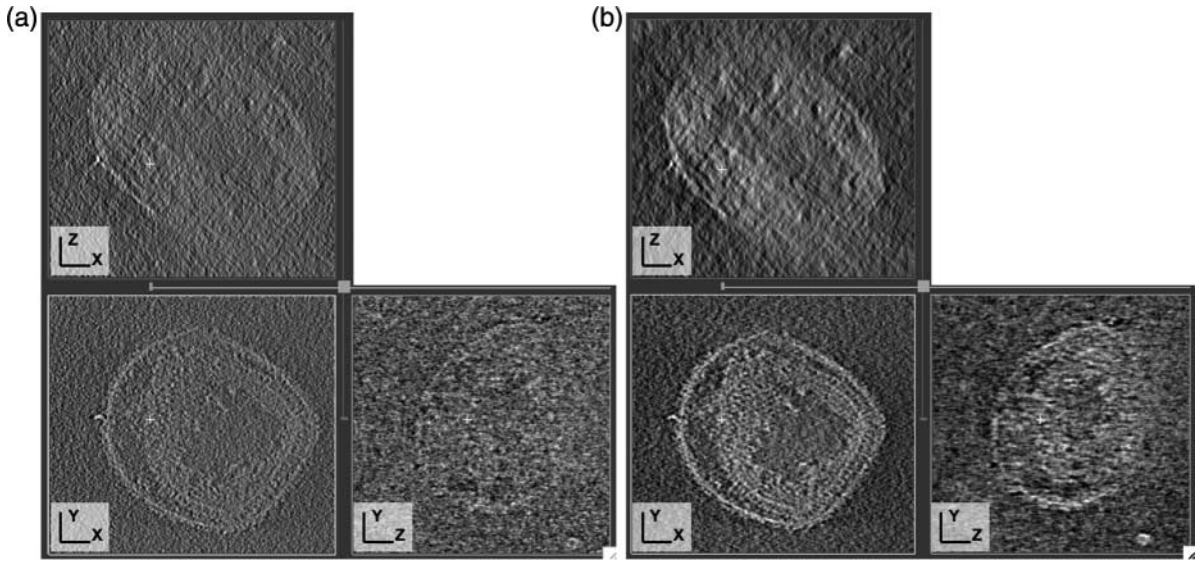

FIGURE 2. 3D reconstruction of Vaccinia virus [13]. Left: Result with WBP. Right: 100 iterations of the iterative method called SIRT. The XZ, XY and ZY planes are shown at top-left, bottom-left and bottom-right panels, respectively. The planes containing the Z-axis clearly show the artefacts due to the limited-tilt range conditions present in ET, in particular blurring and fading-out of features. Furthermore, it is clearly seen that the contrast in the SIRT reconstruction is much better.

The relevance of WBP in ET mainly stems from the linearity and the computational simplicity of the method (O($N^3 \times M$), where N^3 is the number of voxels of the volume and M is the number of projection images). The main disadvantages of WBP are that (i) the results may be strongly affected by limited tilt angle data obtained with the microscope and (ii) WBP does not implicitly take into account the transfer function of the electron microscope or the noise conditions. As a consequence of the latter, *a posteriori* regularization techniques (such as low-pass filtering) may be needed to attenuate the effects of the noise.

Several GPU implementations to accelerate WBP and make it suitable for real-time reconstruction have been reported in ET [7, 8]. Recently, a brand new GPU approach has been introduced that addresses WBP from a matrix perspective by formulating the reconstruction problem as a set of SpMV products, with the matrix being constant and shared by all the products [10]. In conjunction with the use of optimal matrix storage formats on GPUs, this approach succeeds in further accelerating WBP without dealing with low-level GPU programming features.

On the other side of the arena, there exists another family of reconstruction methods, the series expansion methods. These ones model the tomographic reconstruction problem as a large system of linear equations. This system is typically solved by means of iterative methods, which leads to the iterative reconstruction methods (see next section). These methods have turned out to be more robust against the conditions (limited angle data, noise) found in ET [5, 22, 23]. However, they have not been used extensively in this field because of their computational demands. SIRT is one of such methods [24, 25].

Figure 2 illustrates the behaviour of WBP and SIRT in the 3D reconstruction of Vaccinia virus [13]. On the left-hand side of the figure, the reconstruction obtained using WBP is shown, where the noise and artefacts due to the limited-tilt range conditions in ET are apparent. On the right, the reconstruction resulting from 100 iterations of SIRT is presented, which shows better contrast and fewer artefacts. Nonetheless, SIRT took two orders of magnitude longer than WBP. Then, high-performance computing should be applied in order to reduce the computation time of SIRT. Our approach to accelerate it combines GPU computing and the matrix implementation of SIRT.

3. ITERATIVE RECONSTRUCTION: THEORETICAL BACKGROUND

Assuming voxels as basis functions to represent the volume, the 3D problem can then be decomposed into a set of independent two-dimensional (2D) reconstruction subproblems corresponding to the 2D slices perpendicular to the tilt axis [6] (see Fig. 1 where a slice is sketched). The 3D volume is obtained by stacking the 2D slices reconstructed from the corresponding sinogram (i.e. the set of 1D projections). In the following, we thus focus on the 2D reconstruction problem.

Iterative methods are based on an image formation model where the projection measurements (i.e. the components of the sinogram $\mathbf{p}$) depend linearly on the slice $\mathbf{g}^\star$ in such a way that

$$p_i = \sum_{j=1}^{m} A_{i,j} g_j^\star \quad 1 \leq i \leq n, \tag{1}$$

where $n = n_{\text{tilts}} n_{\text{bins}}$ is the dimension of $\mathbf{p}$, with n_{tilts} being the number of projection angles and n_{bins} the number of projection values obtained for every projection angle; $m = m_x m_y$ is the dimension of $\mathbf{g}^\star$, i.e. the total number of voxels in every slice, with m_x and m_y being the number of voxels in the x and y dimensions, respectively and $A_{i,j}$ is a weighting factor representing the contribution of the voxel j to the projection value i, and its value only depends on the geometry of the projections. The set of weighting factors defines the $n \times m$ matrix $\mathbf{A}$. This matrix is sparse, i.e. many coefficients are zero, since the contribution of every voxel is associated with a small subset of projection values. Equation (1) can then be expressed as an SpMV product, $\mathbf{p} = \mathbf{A}\mathbf{g}^\star$, where $\mathbf{A}$ is usually called the forward projection operator.

Under those assumptions, the reconstruction of the slice $\mathbf{g}^\star$ can be modelled as the inverse problem of estimating the $g_j^\star$'s by solving the system of linear equations given by Equation (1). In practice, the system is ill-conditioned due to a variety of reasons (e.g. noise, discretization process, etc.) and a least square problem must thus be solved to compute an approximation $\mathbf{g}$ of $\mathbf{g}^\star$. As a consequence, iterative methods to solve the system are used, which lead to the iterative tomographic reconstruction methods.

There exist numerous iterative approaches for the reconstruction problem. In the ET field, SIRT is an iterative method that has got an excellent reputation. Conceptually speaking, in each iteration, this algorithm computes the error between the projection measurements and the projections calculated from the reconstruction at the current iteration. Then, it refines the reconstruction by backprojecting the average error. In mathematical terms, the algorithm considers all the orthogonal projections of the current iterate (k) onto the hyperplanes defined by the equations of the system (Equation (1)), and takes the average of these projections, as expressed by the following equation:

$$g_j^{k+1} = g_j^k + \sum_{i=1}^{n} \frac{p_i - \sum_{l=1}^{m} A_{i,l} g_l^k}{\sum_{l=1}^{m} A_{i,l}^2} A_{i,j} \quad \text{for } 1 \le j \le m. \quad (2)$$

Equation (2) can be expressed in matrix form as:

$$\mathbf{g^{k+1}} = \mathbf{g^k} + \mathbf{\Delta g^{k+1}} = \mathbf{g^k} + \mathbf{B e^k}, \quad (3)$$

where $\mathbf{B} = \mathbf{A}^{\mathrm{T}}$ is the backward projection operator, and the components of vector $\mathbf{e^k}$ are:

$$e_i^k = \frac{p_i - q_i^k}{w_i} \quad \text{for } 1 \le i \le n \quad (4)$$

with

$$w_i = \sum_{l=1}^{m} A_{i,l}^2, \quad (5)$$

and

$$q_i^k = \sum_{l=1}^{m} A_{i,l} g_l^k \quad \text{for } 1 \le i \le n \quad (6)$$

or, in matrix form:

$$\mathbf{q^k} = \mathbf{A g^k}. \quad (7)$$

Equation (4) basically consists of simple component-wise vector operations. The weights in Equation (5) are calculated directly from the matrix $\mathbf{A}$ and are constant for all iterations. Equations (7) and (3) show that one iteration of the reconstruction method requires two SpMV operations, with matrix $\mathbf{A}$ and its transpose $\mathbf{B} = \mathbf{A}^{\mathrm{T}}$, respectively.

Iterative 3D reconstruction can then be expressed as:

$$\mathbf{g_s^{k+1}} = \mathbf{g_s^k} + \mathbf{B e_s^k} \quad 1 \le s \le N_{\text{slices}}, \quad (8)$$

where N_{slices} is the total number of slices in the 3D volume and s denotes the index of the slice to be computed from the corresponding sinogram. As a consequence, the reconstruction of the volume requires $2 \times K \times N_{\text{slices}}$ SpMV products, where K is the number of iterations. In half of the products, the matrix involved is $\mathbf{A}$ (forward projection operator) whereas its transpose $\mathbf{B} = \mathbf{A}^{\mathrm{T}}$ (backprojection operator) is involved in the other half. Those matrices are common for all slices because the projections have the same geometry for all slices. In those matrices, the location of non-zero coefficients (referred to as non-zeroes) exhibits a regular pattern related to its definition. Using the voxel-driven projection approach [26], the elements of $\mathbf{B} = \mathbf{A}^{\mathrm{T}}$ are located in $m_x \times n_{\text{bins}}$ blocks, which are mostly structured by bi-diagonals [10]. Moreover, in every row of every block, there are no more than two contiguous non-zeroes. Therefore, the maximum number of non-zeroes in each row of that matrix is $2\,n_{\text{tilts}}$. The same features are applicable to matrix $\mathbf{A}$, with the appropriate modifications. The reader is referred to [10] for an in-depth description of the structure of the matrix.

4. MATRIX APPROACH TO ITERATIVE RECONSTRUCTION

Our matrix approach consists in implementing the reconstruction method as a set of SpMV products, along with some basic vector operations, as described in the previous section. Algorithm 1 describes the matrix implementation of SIRT and highlights the two SpMV operations in each iteration, the operations performed at the GPU (i.e. the GPU kernels) marked with the symbol $\star$, and also the CPU–GPU transfers.

The size of the matrices $\mathbf{A}$ and $\mathbf{B}$ are reduced significantly by exploiting of the symmetry relationships among the projection coefficients [10]. To store $\mathbf{A}$ and $\mathbf{B}$, we make use of a recently devised scheme to represent sparse matrix data on GPUs and that have turned out to be key for an efficient SpMV computation [27]. This scheme is known as ELLPACK-R and is based on a well-known format proposed some time ago [28, 29]. Given a sparse matrix $\mathbf{M}$, ELLPACK-R consists of two arrays with a number of rows equal to that of the original matrix and a number of columns equal to the maximum number of non-zeroes in the rows. The first array, M_{sp}, stores the non-zeroes

and the second, I, stores the original column index in matrix **M** for each value in M_{sp}. An additional vector rl keeps the actual number of non-zeroes in each row. The reader is referred to [27] for a detailed description of the ELLPACK-R format.

The SpMV operation using the ELLPACK-R scheme is optimally executed on the GPU by launching multiple threads that, independently, compute the product of a row of the matrix [27]. Despite that the matrices **A** and $\mathbf{B} = \mathbf{A}^{\mathrm{T}}$ are stored using the same format and that the actual number of operations is the same, the performance is not expected to be similar due to their different characteristics, as explained in the following. In ET, the number of tilt angles is much lower than the dimension of the images. As a consequence, the matrices **A** and **B** are not square at all, which leads to a different structure when translated into the ELLPACK-R format. Figure 3 presents an illustrative example with $m_x m_y = 2 \times 6$ and $n_{\text{tilts}} n_{\text{bins}} = 2 \times 4$ where the arrays A_{sp} and B_{sp} of the ELLPACK-R format are shown. It turns out that **B** has a higher number of rows ($m = m_x m_y$) that are more evenly filled, with a maximum number of non-zeroes per row of $2\,n_{\text{tilts}}$. On the contrary, **A** has a lower amount of rows that are less evenly filled. Moreover, the length of the rows in **A** may be significantly higher than in **B**. These structures have a strong impact on the exploitation of GPU, and so the product $\mathbf{Be^k}$ is accomplished by more concurrent threads with more balanced workload, which ensures better GPU multiprocessor occupancy and parallelism than the computation of $\mathbf{Ag_s^k}$ in Algorithm 1. Finally, another important difference between **A** and **B** is the fact that the ELLPACK-R structure of the latter is well defined *a priori* (its dimensions are $m \times (2\,n_{\text{tilts}})$), which allows its creation using the parallel capabilities of the GPU, whereas it does not hold for the former at all.

Algorithm 1 Matrix SIRT algorithm on GPU.

Compute **A** and copy it to GPU device memory
⋆ Compute **B** matrix
⋆ Compute the weights **w** (Eq. 5)
For $s = 1$ to $s = N_{slices}$
  Copy $\mathbf{p_s}$ to GPU device memory
  ⋆ $\mathbf{g_s^0} = 0$
  ⋆ For $k = 0$ to $K - 1$
    ⋆ $\mathbf{q^k} = \mathbf{Ag_s^k}$; ← *SpMV product*
    ⋆ Compute $\mathbf{e_s^k}$ from $\mathbf{p_s}$, $\mathbf{q^k}$, **w** (Eq. 4)
    ⋆ $\mathbf{g_s^{k+1}} = \mathbf{g_s^k} + \mathbf{Be_s^k}$; ← *SpMV product*
  Copy $\mathbf{g_s^K}$ to CPU main memory

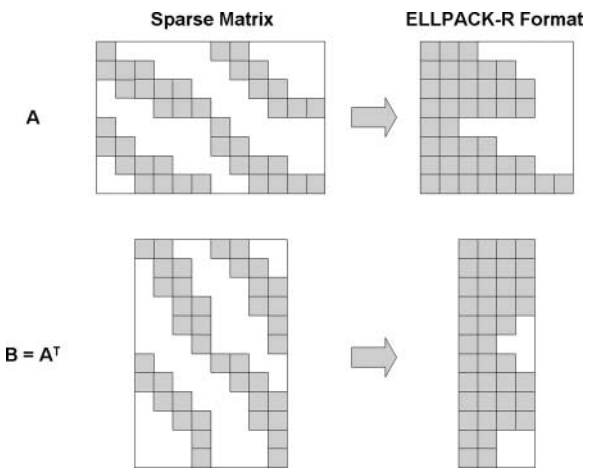

FIGURE 3. ELLPACK-R format of **A** and $\mathbf{B} = \mathbf{A}^{\mathrm{T}}$ matrices.

5. RESULTS

The matrix approach to SIRT has been implemented using the library for SpMV based on the ELLPACK-R format developed by the authors [27, 30]. It has been evaluated using several representative data sets on three different state-of-the-art GPUs (NVIDIA C1060, GTX285 and C2050) that were installed in different computers. Compute unified device architecture was the choice for GPU code development. The characteristics of the GPUs are shown in Table 1. Note that C2050 is based on the cutting-edge 'Fermi' NVIDIA technology. For comparison with the CPU implementations, a computer with a CPU Intel Xeon Quad-core 5520 at 2.27 GHz and 24 GB SDRAM DDR3 at 1333 MHz, under Linux was employed. Two CPU implementations have been tested. The first one (denoted by 'Recalc.' in Table 1) is the standard implementation based on recalculation of the matrix coefficients (i.e. the coefficients $A_{i,j}$, $B_{j,i}$ are calculated on-the-fly every time they are needed). The second one [denoted by compressed row storage (CRS)] is a matrix implementation that makes uses of the most extended sparse matrix storage format on CPUs, the CRS, as already done in our previous work [10]. In both cases, one core of the CPU was used.

Three data sets (denoted by data 1–3) with 71 images of 1024×1024, 61 images of 1024×1024 and 60 images of 2048×2048 were processed to yield tomograms of $1024 \times 1024 \times 140$, $1024 \times 1024 \times 480$ and $2048 \times 2048 \times 256$, respectively. Two more data sets were tested in order to

TABLE 1. Characteristics of the GPUs.

	C1060	GTX280[a]	GTX285	C2050
Peak GFlops	933	933	1062	1030
Bandwidth (GB/s)	102	141	159	144
Clock (GHz)	1.3	1.3	1.4	1.15
Mem. clock (MHz)	800	1107	1242	1500
Memory (GB)	4	1	2	2.6
Cores	240	240	240	448

[a]This GPU was not used in this work. It is included for comparison with the work [9].

TABLE 2. Run-time results.

			Total run-time (s)—CPU		Total run-time (s)—GPU		
	Tilt-series, thickness	Mem (MB)	Recalc.	CRS	C1060	GTX285	C2050
Data 1	71 × 1024 × 140	251.93	10 539.92	3028.19	130.05	95.89	87.77
Data 2	61 × 1024 × 1024, 480	628.84	30 302.82	10 766.78	562.53	403.96	296.58
Data 3	60 × 2048 × 2048, 256	1587.93	64 176.81	20 463.08	1193.74	881.17	665.84
Data 4	61 × 712 × 1012, 296	268.13	12 771.30	3454.33	143.98	116.85	110.93
Data 5	61 × 1424 × 2024, 591	1070.78	138 853.00	38 886.52	1538.39	1269.63	1137.38

compare with the most advanced GPU solution thus far [9]. That work assessed their implementation on a GPU NVIDIA GTX280, whose characteristics are also shown in Table 1. These data sets (denoted by data 4–5) had 61 images of 712 × 1012 and 1424 × 2024 to produce tomograms of 712 × 1012 × 296 and 1424 × 2024 × 591, respectively. A representative amount of 30 iterations of SIRT were run for all experiments.

The results, including memory consumption and total processing time on the CPU and GPUs, are summarized in Table 2. The total time includes the computation of the matrices, which turned out to be around 1–2% in all cases. We also measured the time spent in the data transference between the CPU and the GPU, and it turned out to be in the order of 1% of the total time. The results clearly show that the requirements fit the amount of memory shipped in standard GPUs, with 1.5 GB in the largest case. Taking into account that standard GPUs have 4 GB or more, it is expected that this matrix approach will readily accommodate the demands needed for even larger tomographic reconstructions. Table 2 also shows that the matrix CPU implementation using the CRS format involves an acceleration factor around 3.0×–3.5× when compared with the approach based on coefficient recalculation, thereby confirming the behaviour previously observed with WBP [10]. As far as GPU processing times are concerned, the results confirm that 30 iterations of SIRT for huge tomographic reconstructions (1.0–1.7 Gvoxel), such as data 3 or data 5, can be computed in only 11–19 min, which is striking.

Table 3 and Fig. 4 show that all GPU implementations achieve a significant acceleration factor compared with the CPU ones. In the case of coefficient recalculation, the GPU proves to outperform the CPU in a factor even higher than 100×. Nevertheless, the gain obtained by the GPU compared with the CRS-based CPU implementation is reduced by the factor of 3.0×–3.5× mentioned above, thus reaching 36× in the best case. With regard to the particular GPUs, the Fermi architecture remarkably exhibits the best performance with an acceleration factor >90× and >30×, for the two CPU implementations, respectively. The C1060 reaches a very good speedup factor ([53×, 90×] and [17×, 25×], respectively), but it is consistently lower than the other GPUs. The behaviour of GTX285 is in-between. The speedup factors achieved by

TABLE 3. Speedup factors. GPU vs. CPU.

	C1060	GTX285	C2050
CPU—Recalc.			
Data 1	81.05	109.92	120.08
Data 2	53.87	75.01	102.18
Data 3	53.76	72.83	96.38
Data 4	88.70	109.29	115.13
Data 5	90.26	109.37	122.08
CPU—CRS			
Data 1	23.29	31.58	34.50
Data 2	19.14	26.65	36.30
Data 3	17.14	23.22	30.73
Data 4	23.99	29.56	31.14
Data 5	25.28	30.63	34.19

the GPUs for matrix SIRT prove to be lower than those previously obtained for matrix WBP [10], which were in the range [100×, 130×] and [28×, 36×] for the two CPU implementations, respectively. As explained in Section 3, this is caused by the fact that the SpMV operation with the matrix $\mathbf{A}$ is not as efficient as that with $\mathbf{B} = \mathbf{A}^{\mathrm{T}}$.

Finally, our matrix approach was compared with the leading GPU solution in the ET field [9], which relies on low-level programming and exploitation of tricky features of GPUs. Table 4 reproduces the results (time per iteration) obtained by that work and also presents those obtained with our matrix approach on the different platforms. The fact that the GPU used in that work (GTX280) is different from some of the GPUs tested here (GTX285 and C2050) makes the results not directly comparable. However, our GPU C1060 essentially has the same features as the GTX280, though with a slower memory and worse bandwidth. Therefore, the assessment of the goodness of the matrix approach based on the comparison between the GTX280 and C1060 is thus reliable. Table 4 indeed shows that the matrix implementation performs better. An analysis of the speedup of C1060 (calculated from the time spent at the GTX280) with the different data sets reveals that the gain decreases with the data set size. For medium-large sizes (data 4), the gain is higher than 2× whereas for huge data sets

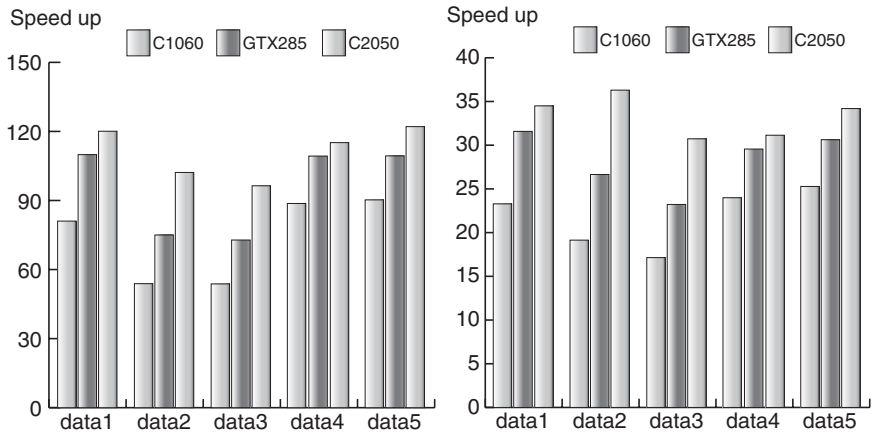

FIGURE 4. Speedup factors obtained with matrix SIRT on the different GPU when compared with the CPU implementations (left) based on coefficient recalculation and (right) matrix-based using the CRS format.

TABLE 4. Comparison with Xu *et al.* [9].

	Data 4	Data 5
Average iteration time (s)		
GPU GTX280 [9]	10.67	58.47
GPU C1060	4.80	51.28
GPU GTX285	3.90	42.32
GPU C2050	3.70	37.91
Speedup		
GPU C1060	2.22	1.14
GPU GTX285	2.74	1.38
GPU C2050	2.88	1.54

(data 5) the matrix implementation is slightly better (1.14×). This behaviour is also shared by the other two GPUs, GTX285 and C2050. Therefore, these results draw the conclusion that the matrix approach succeeds in providing reconstructions at a speed better than sophisticated GPU strategies. Finally, Table 4 also confirms the Fermi GPU is the best platform for medium to huge data sets.

6. CONCLUSION

In this work, we have presented a new approach to 3D reconstruction by means of SIRT. Although we have focused on ET, the approach is readily extendible and applicable to other tomography fields. It relies on the formulation of the iterative method as a set of SpMV products. The matrices with the projection coefficients are kept in memory during the whole process by means of optimized sparse data structures. They are thus exploited in the calculation of all the slices of the volume. The results undoubtedly demonstrate that the matrix approach outperforms the standard approach based on coefficient recalculation. When combined with the power of GPUs, the new approach achieves striking speedup factors. Moreover, the approach proves to provide large reconstructions at a pace better than sophisticated GPU strategies. In addition to high performance, the major advantages of this approach are the simplicity to be implemented and the versatility to be run under different GPU architectures.

FUNDING

This work was supported by the Spanish Ministry of Science [TIN2008-01117], National Research Council (CSIC) [PIE-200920I075] and Junta de Andalucia [P10-TIC-6002].

REFERENCES

[1] Herman, G.T. (2009) *Image Reconstruction From Projections: The Fundamentals of Computerized Tomography* (2nd edn). Springer, London.
[2] Lucic, V., Foerster, F. and Baumeister, W. (2005) Structural studies by electron tomography: from cells to molecules. *Annu. Rev. Biochem.*, **74**, 833–865.
[3] Fernandez, J.J., Sorzano, C.O.S., Marabini, R. and Carazo, J.M. (2006) Image processing and 3D reconstruction in electron microscopy. *IEEE Signal Process. Mag.*, **23**, 84–94.
[4] Medalia, O., Weber, I., Frangakis, A., Nicastro, D., Gerisch, G. and Baumeister, W. (2002) Macromolecular architecture in eukaryotic cells visualized by cryoelectron tomography. *Science*, **298**, 1209–1213.
[5] Fernandez, J.J., Lawrence, A.F., Roca, J., García, I., Ellisman, M.H. and Carazo, J.M. (2002) High performance electron tomography of complex biological specimens. *J. Struct. Biol.*, **138**, 6–20.
[6] Fernandez, J.J. (2008) High performance computing in structural determination by electron cryomicroscopy. *J. Struct. Biol.*, **164**, 1–6.

[7] Castano-Diez, D., Mueller, H. and Frangakis, A.S. (2007) Implementation and performance evaluation of reconstruction algorithms on graphics processors. *J. Struct. Biol.*, **157**, 288–295.

[8] Castano-Diez, D., Moser, D., Schoenegger, A., Pruggnaller, S. and Frangakis, A.S. (2008) Performance evaluation of image processing algorithms on the GPU. *J. Struct. Biol.*, **164**, 153–160.

[9] Xu, W., Xu, F., Jones, M., Keszthelyi, B., Sedat, J., Agard, D. and Mueller, K. (2010) High-performance iterative electron tomography reconstruction with long-object compensation using graphics processing units (GPUs). *J. Struct. Biol.*, **171**, 142–153.

[10] Vazquez, F., Garzon, E.M. and Fernandez, J.J. (2010) A matrix approach to tomographic reconstruction and its implementation on GPUs. *J. Struct. Biol.*, **170**, 146–151.

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

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

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

[14] Nicastro, D., Schwartz, C., Pierson, J., Gaudette, R., Porter, M.E. and McIntosh, J.R. (2006) The molecular architecture of axonemes revealed by cryoelectron tomography. *Science*, **313**, 944–948.

[15] Al-Amoudi, A., Diez, D.C., Betts, M.J. and Frangakis, A.S. (2007) The molecular architecture of cadherins in native epidermal desmosomes. *Nature*, **450**, 832–837.

[16] Brandt, F., Etchells, S.A., Ortiz, J.O., Elcock, A.H., Hartl, F.U. and Baumeister, W. (2009) The native 3D organization of bacterial polysomes. *Cell*, **136**, 261–271.

[17] Brandt, F., Carlson, L.A., Hartl, F.U., Baumeister, W. and Grunewald, K. (2010) The three-dimensional organization of polyribosomes in intact human cells. *Mol. Cell*, **39**, 560–569.

[18] Penczek, P., Marko, M., Buttle, K. and Frank, J. (1995) Double-tilt electron tomography. *Ultramicroscopy*, **60**, 393–410.

[19] Mastronarde, D.N. (1997) Dual-axis tomography: an approach with alignment methods that preserve resolution. *J. Struct. Biol.*, **120**, 343–352.

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

[21] Rademacher, M. (1992) Weighted Back-Projection Methods. In Frank, J. (ed.), *Electron Tomography. Three-Dimensional Imaging with the Transmission Electron Microscope*. Plenum Press, New York.

[22] Marabini, R., Herman, G. and Carazo, J. (1998) 3D reconstruction in electron microscopy using ART with smooth spherically symmetric volume elements (blobs). *Ultramicroscopy*, **72**, 53–56.

[23] Fernandez, J.J., Gordon, D. and Gordon, R. (2008) Efficient parallel implementation of iterative reconstruction algorithms for electron tomography. *J. Parallel Distrib. Comput.*, **68**, 626–640.

[24] Kak, A.C. and Slaney, M. (2001) *Principles of Computerized Tomographic Imaging*. SIAM, Philadelphia.

[25] Gilbert, P. (1972) Iterative methods for the 3D reconstruction of an object from projections. *J. Theor. Biol.*, **76**, 105–117.

[26] Bruyant, P. (2002) Analytic and iterative reconstruction algorithms in SPECT. *J. Nucl. Med.*, **43**, 1343–1358.

[27] Vazquez, F., Fernandez, J.J. and Garzon, E.M. (2011) A new approach for sparse matrix vector product on NVIDIA GPUs. *Concurrency Comput. Pract. Exp.*, doi:10.1002/cpe.1658.

[28] Kincaid, D.R. and Young, D.M. (1984) The ITPACK Project: Past, Present And Future. In Birkhoff, G. and Schoernstadt, A. (eds), *Elliptic Problem Solvers II*. Academic Press, New York.

[29] Rice, J.R. and Boisvert, R.F. (eds) (1985) *Solving Elliptic Problems using ELLPACK*. Springer, New York.

[30] Vazquez, F., Ortega, G., Fernandez, J.J. and Garzon, E.M. (2010) Improving the Performance of the Sparse Matrix Vector Product with GPUs. *Proc. IEEE Int. Conf. Computer and Information Technology (CIT 2010)*, Bradford, UK, June 29–July 1, pp. 1146–1151. IEEE Press.