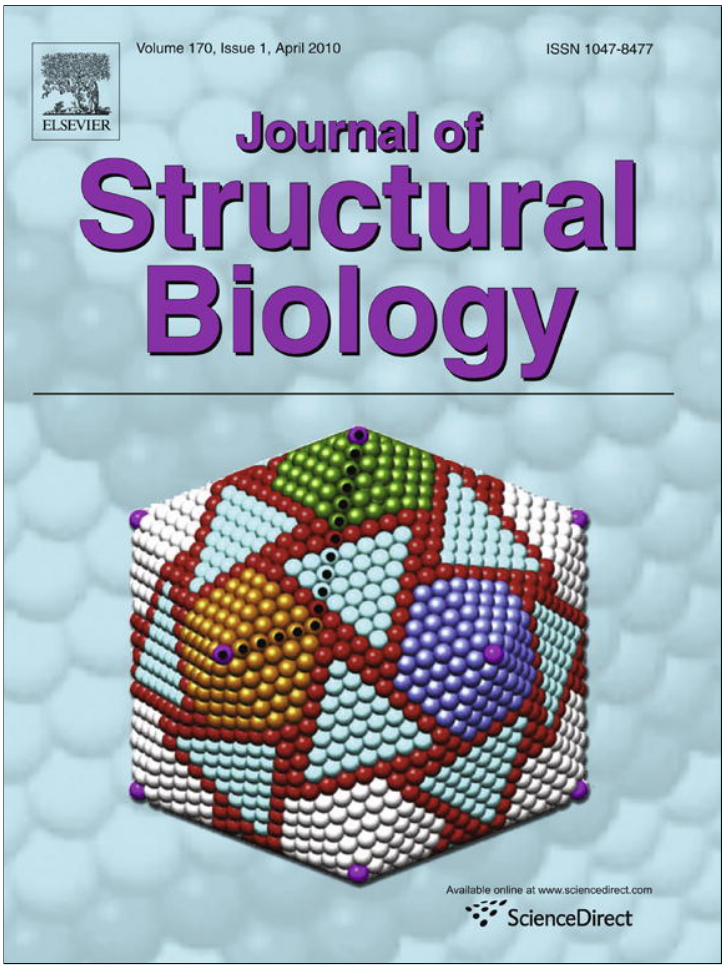

Journal of Structural Biology 170 (2010) 146–151

Contents lists available at ScienceDirect

Journal of Structural Biology

journal homepage: www.elsevier.com/locate/yjsbi

A matrix approach to tomographic reconstruction and its implementation on GPUs

F. Vázquez [a], E.M. Garzón [a], J.J. Fernández [a,b,*]

[a] Dept. Computer Architecture, University of Almería, 04120 Almería, Spain
[b] National Center for Biotechnology (CSIC), Cantoblanco, 28049 Madrid, Spain

ARTICLE INFO

Article history:
Received 23 October 2009
Received in revised form 27 January 2010
Accepted 28 January 2010
Available online 2 February 2010

Keywords:
Electron tomography
Three-dimensional reconstruction
WBP, weighted backprojection
GPU, graphics processing unit
Sparse matrix
Sparse matrix-vector product

ABSTRACT

Electron tomography allows elucidation of the molecular architecture of complex biological specimens. Weighted backprojection (WBP) is the standard reconstruction method in the field. In this work, three-dimensional reconstruction with WBP is addressed from a matrix perspective by formulating the problem as a set of sparse matrix-vector products, with the matrix being constant and shared by all the products. This matrix approach allows efficient implementations of reconstruction algorithms. Although WBP is computationally simple, the resolution requirements may turn the tomographic reconstruction into a computationally intensive problem. Parallel systems have traditionally been used to cope with such demands. Recently, graphics processor units (GPUs) have emerged as powerful platforms for scientific computing and they are getting increasing interest. In combination with GPU computing, the matrix approach for WBP exhibits a significant acceleration factor compared to the standard implementation.

1. Introduction

Electron tomography (ET) is an essential structural technique in cellular biology (Lucic et al., 2005). An individual specimen is imaged in the electron microscope, and a set of projection images is taken at different orientations. The tomographic reconstruction algorithms combine those images to obtain the three-dimensional (3D) structure of the specimen. Tomographic reconstruction can be modelled as a least square problem to be solved by matrix algorithms (Herman, 2009), where large sparse matrices are involved. In the implementations, however, the matrix coefficients are usually re-computed because of the large memory requirements. Nowadays, the memory available in modern computers allows storage of large matrices, which may improve the performance of algorithms. Nevertheless, sparse matrix data structures must be carefully devised to optimize the access to the memory hierarchy.

Large projection images (1K × 1K–4K × 4K pixels) are typically used to fulfill the resolution needs in ET. High performance computing has traditionally been used to cope with the computational requirements (Fernández, 2008). Graphics Processing Units (GPUs) are receiving great interest in 3D electron microscopy because many image processing procedures are well suited for the SIMD (single instruction, multiple data) parallelism massively exploited by GPU (Castano-Diez et al., 2007, 2008; Schmeisser et al., 2009). Programming interfaces, such as CUDA, facilitate the development of applications targeted at GPUs (Castano-Diez et al., 2008; Nickolls et al., 2008). The performance of GPUs for common steps in ET, and in particular for tomographic reconstruction, has turned out to be excellent (Castano-Diez et al., 2007, 2008).

This work introduces a matrix approach to tomographic reconstruction. The standard method, weighted backprojection (WBP), is firstly formulated as a set of sparse matrix–vector products (SpMV), where the sparse matrix is invariable. The nonzero elements of the matrix are stored into an optimized sparse matrix data structure. The power of GPU computing is then exploited to further improve the performance and yield reconstructions of biological datasets in seconds.

2. Matrix WBP

Assuming single tilt axis geometry, the 3D problem can be decomposed into a set of independent two-dimensional (2D) reconstruction subproblems corresponding to the slices perpendicular to the tilt axis (Fernández, 2008). The 3D volume is obtained by stacking the 2D slices reconstructed from the corresponding sinogram (i.e. the set of 1D projections). Now we will thus focus on the 2D reconstruction problem.

The projection process can be modelled as follows. The sinogram $\mathbf{p}$ is related to the slice $\mathbf{g}^{\star}$ by the discrete Radon Transform or projection operation:

* Corresponding author. Address: National Center for Biotechnology (CSIC), Campus Cantoblanco, C/Darwin 3, 28049 Madrid, Spain. Fax: +34 91 585 4506.
E-mail addresses: jj.fernandez@cnb.csic.es, jjfdez@ual.es (J.J. Fernández).

1047-8477/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jsb.2010.01.021

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

where $n = n_{tilts} n_{bins}$ 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.

Therefore, the projection operation can be defined as a sparse matrix–vector product, $\mathbf{p} = \mathbf{A}\mathbf{g}^{\star}$, where $\mathbf{A}$ is usually called the forward projection operator. Then, the system $\mathbf{p} = \mathbf{A}\mathbf{g}^{\star}$ must be solved to compute the unknown slice $\mathbf{g}^{\star}$. In practice, the system is ill-conditioned and a least square problem must thus be solved to compute an approximation of $\mathbf{g}^{\star}$.

WBP is the standard method to solve this problem (Radermacher, 2006), which reconstructs the specimen by uniformly distributing the specimen density present in the projection images over computed backprojection rays. Formally, the backprojection can be defined by means of the matrix backprojection operator $\mathbf{B}$ as:

$$g_j = \sum_{i=1}^{n} B_{j,i} p_i \quad 1 \leqslant j \leqslant m \tag{2}$$

where $\mathbf{B}$ is the transpose of matrix $\mathbf{A}$, and when the number of tilt angles is large enough, the vector $\mathbf{g}$ is a good estimation of the slice $\mathbf{g}^{\star}$. In WBP a high-pass filter is applied to the projections before backprojection (Radermacher, 2006), whose burden is usually negligible. In the following, we assume that the projections are already weighted.

Our matrix WBP approach then reconstructs a 3D object as a set of independent SpMV products:

$$\mathbf{g}^{\mathbf{s}} = \mathbf{B}\mathbf{p}^{\mathbf{s}} \quad 1 \leqslant s \leqslant N_{slices} \tag{3}$$

where N_{slices} is the total number of slices in the volume. Note that the matrix $\mathbf{B}$: (1) is involved in all the products, since the projections have the same geometry for all slices; and (2) is sparse and the location of nonzero coefficients (referred to as nonzeroes) exhibits some regular pattern related to its definition (i.e. $B_{j,i} = A_{i,j}$). Using the voxel-driven projection approach (Bruyant, 2002), the elements $B_{j,i}$ are located in $m_x \times n_{bins}$ blocks, which are mostly structured by bi-diagonals. Moreover, in every row of every block, there are no more than two contiguous nonzeroes. Therefore, the maximum number of nonzeroes in each row of matrix $\mathbf{B}$ is $2n_{tilts}$. Fig. 1 and Supplementary Fig. S1 illustrate the pattern of $\mathbf{B}$.

Nowadays, the memory requirements to store the sparse matrix are fulfilled in current computers, and the Compressed Row Storage (CRS) is the most extended format on CPUs (Bisseling, 2004). Let m and N_z be the number of rows and the number of nonzeroes of the matrix, respectively. The sparse data structure consists of (1) an array of N_z floating-point numbers that stores the entries; (2) an array of N_z integers that keeps their column index; and (3) an array of m integers with the pointers to the beginning of every row in the arrays (1) and (2). For GPU, however, other data structures have proven better suited for the SpMV operation (Vázquez et al., 2009).

3. Tuning matrix WBP on GPUs

The regularity and the symmetry relationships between the nonzeroes of $\mathbf{B}$ can be exploited to improve the memory access and thus accelerate the SpMV operations on GPUs. Four different implementations have been developed:

3.1. General

The simplest implementation is based on the storage of matrix $\mathbf{B}$ in a sparse matrix data structure optimal for GPUs: the ELL-R scheme (Vázquez et al., 2009). This format consists of two arrays of dimension $m \times (2n_{tilts})$, where $m = m_x m_y$ is the number of rows of $\mathbf{B}$ and $2n_{tilts}$ is the maximum number of nonzeroes in the rows. The first array, B_{sp}, stores the nonzeroes and the second, I, stores the original column index (i) in matrix $\mathbf{B}$ for each value in B_{sp}. An

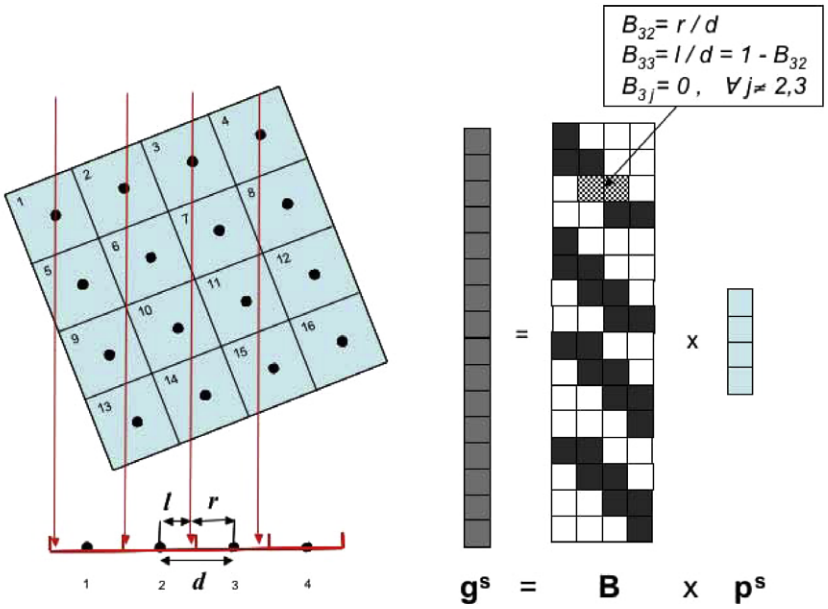

Fig. 1. Pattern of matrix $\mathbf{B}$, using only one tilt angle, with $n_{bins} = 4$, and a slice of 4×4 voxels. In general, a projected pixel contributes to two neighbour projection bins (left). The weighting factors $(l/d, r/d)$ are computed proportionally to the distance to the centre of the projection bin. For instance, the projection of the 3rd pixel contributes to the 2nd and 3rd projection bins, with the proper weights (right). This projection approach makes the matrix B exhibit a blocked bi-diagonal structure (right). Nonzero coefficients are denoted by black boxes in the sketch of matrix $\mathbf{B}$.

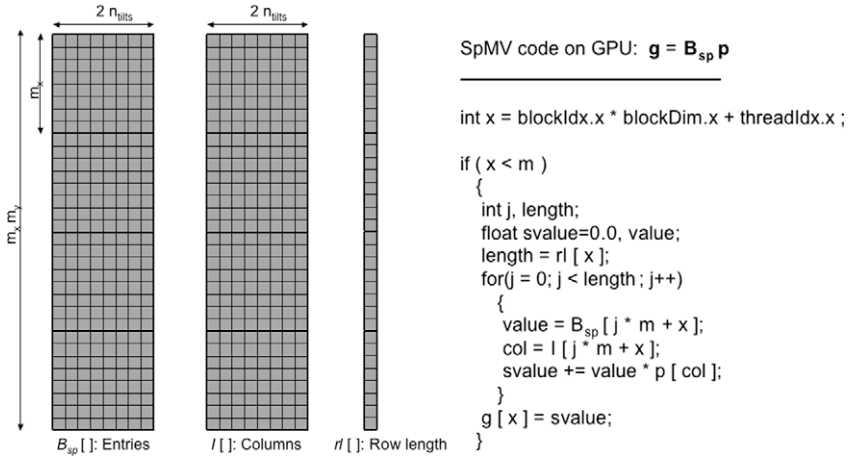

Fig. 2. Sparse data structure based on the ELL-R scheme. (Left) The nonzeroes of matrix **B** (sketched in Figure S1) are densely stored in an array B_{sp}. An auxiliary array I keeps the original column index in **B** of the components stored in B_{sp}. An additional vector rl keeps the actual number of nonzeroes in each row of the matrix. (Right) Algorithm for SpMV on GPU using the ELL-R scheme.

additional vector rl of dimension m keeps the actual number of nonzeroes in each row. The arrays B_{sp} and I store their elements in column-major order. As every thread in the GPU computes a row, this ensures optimal coalesced global memory access. Fig. 2 shows the data structures involved in the ELL-R scheme and the algorithm for the SpMV operation on the GPU.

3.2. Symmetry 1 (sym1)

The relationship between adjacent nonzeroes in **B**, $b_{j,i} \neq 0$ and $b_{j,i+1} \neq 0$, is exploited to reduce the storage size. These pair of nonzeroes comes from the projection coefficients of a given voxel and verifies $b_{j,i+1} = 1 - b_{j,i}$ (Fig. 1). In this implementation, only the first coefficient of the pair is stored in the ELL-R structure whereas the other is easily computed on the fly. Therefore, the size of the data structure is reduced at 50%.

3.3. Symmetry 2 (sym2)

This implementation also takes advantage of the symmetry existing in the projection of a slice: Assuming that the centre of the slice is (0,0), if a point (x,y) is projected to a point $r = x\cos(\theta) + y\sin(\theta)$ in the projection corresponding to the tilt angle θ, it is easy to see that the point $(-x,-y)$ of the slice is then projected to $r_s = -r$ in that projection. Therefore, for a given tilt angle θ, there is no longer need to store the projection coefficients for all the points (x,y) in the slice. This further reduces the storage space of the data structure in nearly another 50%.

3.4. Symmetry 3 (sym3)

This implementation also exploits the fact that in the tilt range typically used in ET the same tilt angles are found in the positive and negative halves (i.e., $-70, -69, \ldots, 69, 70$). Under these conditions, a projection angle of $-\theta$ makes the point $(x,-y)$ projected to the point $r = x\cos(-\theta) - y\sin(-\theta) = x\cos(\theta) + y\sin(\theta)$. Therefore, the projection coefficients are shared with the projection of the point (x,y) with angle θ. As a result, the space requirements for the sparse matrix are further reduced in nearly 50% again.

The three geometry-related symmetry properties allow a significant reduction of the memory requirements to store the matrix **B**. Supplementary Fig. S2 illustrates the symmetry relationships between the components of an example matrix.

The matrix **B** in ELL-R format is generated and stored at the GPU in order to be exploited for all the SpMV operations and avoid latencies due to transfers CPU–GPU. The sinogram being processed

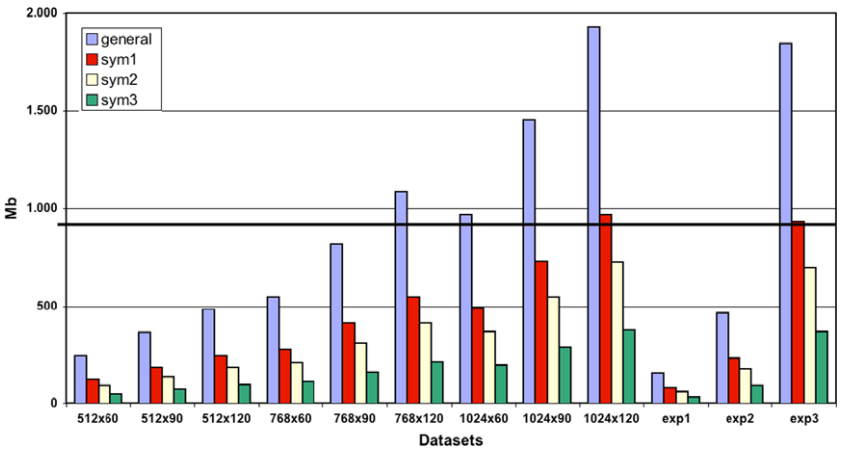

Fig. 3. Memory requirements of the different implementations for the datasets. The limit of 896 MB is imposed by the memory available in the GPU used in this work. The implementations needing higher amounts of memory than this value cannot run on this GPU. However, the use of the symmetries significantly reduces the demands, making most of the problems affordable.

Table 1
Run-times (s).

Dataset		CPU		GPU				
Size	Tilts	Standard	Matrix	Standard	General	Sym 1	Sym 2	Sym 3
512	60	94.81	23.46	2.17	1.52	0.85	0.84	0.86
512	90	142.16	35.39	2.77	2.08	1.21	1.18	1.02
512	120	189.38	47.70	3.02	2.84	1.55	1.48	1.33
768	60	319.83	78.96	6.81	4.92	2.96	2.58	2.77
768	90	478.25	116.36	8.46	7.06	4.04	3.56	3.37
768	120	637.34	155.09	9.84	–	5.20	4.72	4.06
1024	60	759.69	194.17	16.45	–	6.80	6.13	6.19
1024	90	1137.24	293.78	20.06	–	9.37	8.40	7.95
1024	120	1515.25	398.08	23.15	–	–	11.15	9.54
exp1		128.96	36.45	2.97	2.00	1.22	1.06	1.14
exp2		374.21	102.36	8.23	5.71	3.43	3.20	3.27
exp3		2992.22	795.01	64.20	–	–	25.18	26.05

Standard represents the implementation (for CPU and for GPU) based on recomputation of the coefficients; *Matrix* is the implementation of matrix WBP, using the CRS format, on the CPU. The implementation of matrix WBP for GPU using the ELL-R format is denoted by *General*, and the use of the different symmetries is represented by *SymX*. The results marked with '–' indicate unaffordable cases due to the memory demands.

is bound to the fast texture cache of the GPU (Nickolls et al., 2008). The floating-point computations are performed with single precision.

4. Experimental evaluation

The matrix WBP approach was implemented with CUDA and evaluated on a CPU based on Intel Core 2 E8400 at 3.00 GHz, 8 GB RAM and 6 MB L2 cache, under Linux, and a GPU of a NVIDIA GeForce GTX 295 card, with 30 multiprocessors of 8 cores (i.e. 240 cores) at 1.2 GHz, 896 MB of memory and compute capability 1.3. For the CPU, the standard sparse matrix format, CRS, was used. For the GPU, the four implementations based on the ELL-R format were used.

Synthetic datasets were created to carry out an extensive performance evaluation. They comprised a number of aligned projection images to yield cubic 3D reconstructions. The datasets had different image sizes (512, 768 and 1024) and number of tilt angles (60, 90 and 120). These data sizes resemble reconstructed volumes with size from $1024 \times 1024 \times 128$ to $2048 \times 2048 \times 256$ voxels, i.e. 0.5–4 GB. Three additional datasets to actually represent experimental situations were used (denoted by exp1, exp2, exp3): with 71 images of 1024×1024 pixels, 61 images of 1024×1024 pixels and 61 images of 2048×2048 pixels to yield tomograms of $1024 \times 1024 \times 140$, $1024 \times 1024 \times 480$ and $2048 \times 2048 \times 960$ voxels, respectively.

Fig. 3 shows the memory demanded by the sparse data structures. In the general implementation, the demands rapidly increase

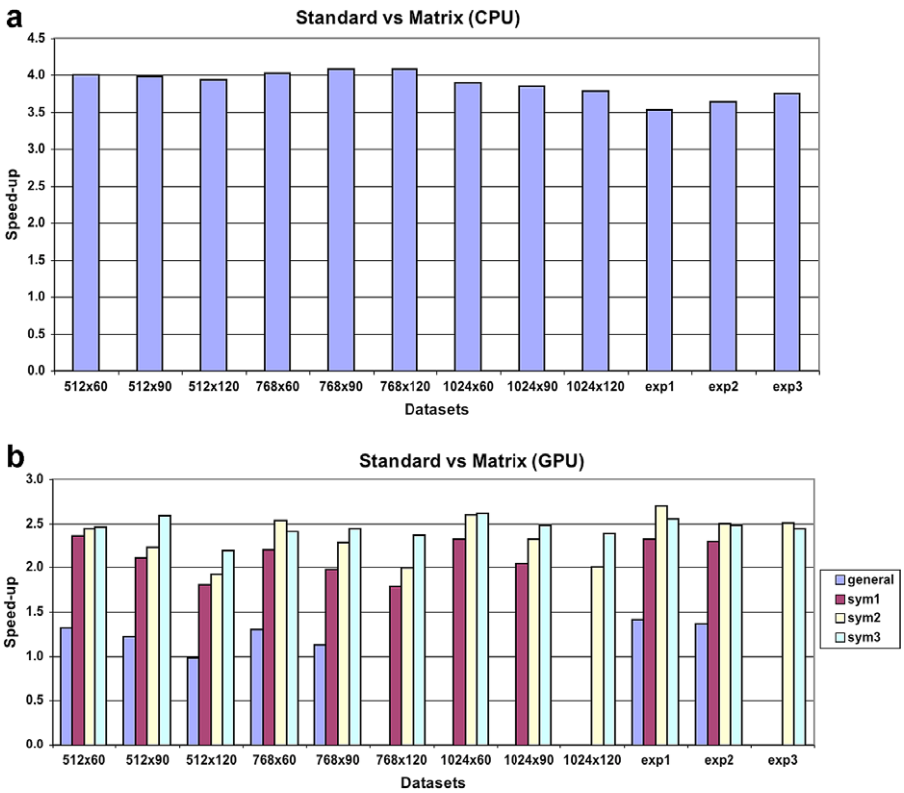

Fig. 4. Speed-up factors showed by the different matrix WBP implementations over the standard WBP based on recomputation of the coefficients. These factors were obtained on the CPU (a) and on GPU (b). For the CPU, the matrix was stored using the CRS format whereas for the GPU the four different implementations based on the ELL-R format were tested.

with the problem size, approaching 2 GB in the largest case. This amount does not turn out to be a problem in modern computers. However, the upper boundary imposed by the memory available in the GPU precludes addressing problem sizes requiring more than 896 MB of memory. Nonetheless, the symmetry relationships decrease the demands and make all problem sizes affordable on the GPU. In particular, the use of two or three symmetries makes it possible to address even the largest case, which is representative of the most expensive cases in ET.

The datasets were subjected to tomographic reconstruction with the standard WBP, based on recomputation of the coefficients, and with matrix WBP on the CPU and on the GPU. The computation times are summarized in Table 1. For a fair comparison, the time required for generating the matrix was taken into account, though it turned out to be negligible (lower than 1% of the total time). The run-time obtained on the CPU reflects the computational complexity of the algorithm, which depends linearly on m, N_{slices} and n_{tilts}. However, on the GPU the run-time increase slower than proportional because of the optimized memory access and the multi-threaded computation, which minimize latencies.

Fig. 4(a) shows that the acceleration factor thanks to the matrix approach is in the range [3.5×, 4.0×] on the CPU, regardless of the problem size. On the GPU, however, the situation is different (see Fig. 4(b)). First, the reduction of the matrix storage based on the symmetry operations is essential to obtain speed-up factors greater than 1.5. Second, for a given image size, the speed-up decreases as a function of the number of tilts. Third, in general the larger the symmetry level, the better the performance is. However, the cases corresponding to the highest level of symmetry (sym3) and small number of tilts (60) do not fulfill the two latter trends. Finally, the highest speed-up factors are around 2.5 and correspond to the highest levels of symmetry operations (sym2, sym3) with a number of angles between 60 and 90. These results demonstrate that matrix WBP succeeds in reducing the computing time required for tomographic reconstruction, either on CPU (by a factor up to 4×) or GPU (up to 2.5×).

Fig. 5(a) compares the speed-up of matrix WBP on the GPU vs. CPU. The GPU exhibits excellent acceleration factors, in general higher than 25× and, in the case of symmetry 3, reaching up to 42×. The use of some symmetry level is key to significantly increase the acceleration, otherwise the improvements only oscillate around 15×. For a given image size and symmetry level, the speed-up exhibits monotonically increasing curves as a function of the number of tilts. On the other hand, the speed-up in general increases with the symmetry level. The exception is for 60 tilts, where the second level (sym2) slightly outperforms the third one, which is clearly observed with the last three datasets.

Finally, in order to estimate the net gain by the use of matrix WBP and GPU computing, the speed-up factors against the standard WBP on the CPU were computed (Fig. 5(b)). For comparison, the acceleration factor of the GPU over the CPU on the standard recomputation-based WBP is presented too. This speed-up is in the range [45×, 65×] and thus higher than those shown in Fig. 5(a). However, in terms of computing time, the standard WBP is worse than the matrix WBP (Table 1). The other data in Fig. 5(b) correspond to the different implementations of matrix

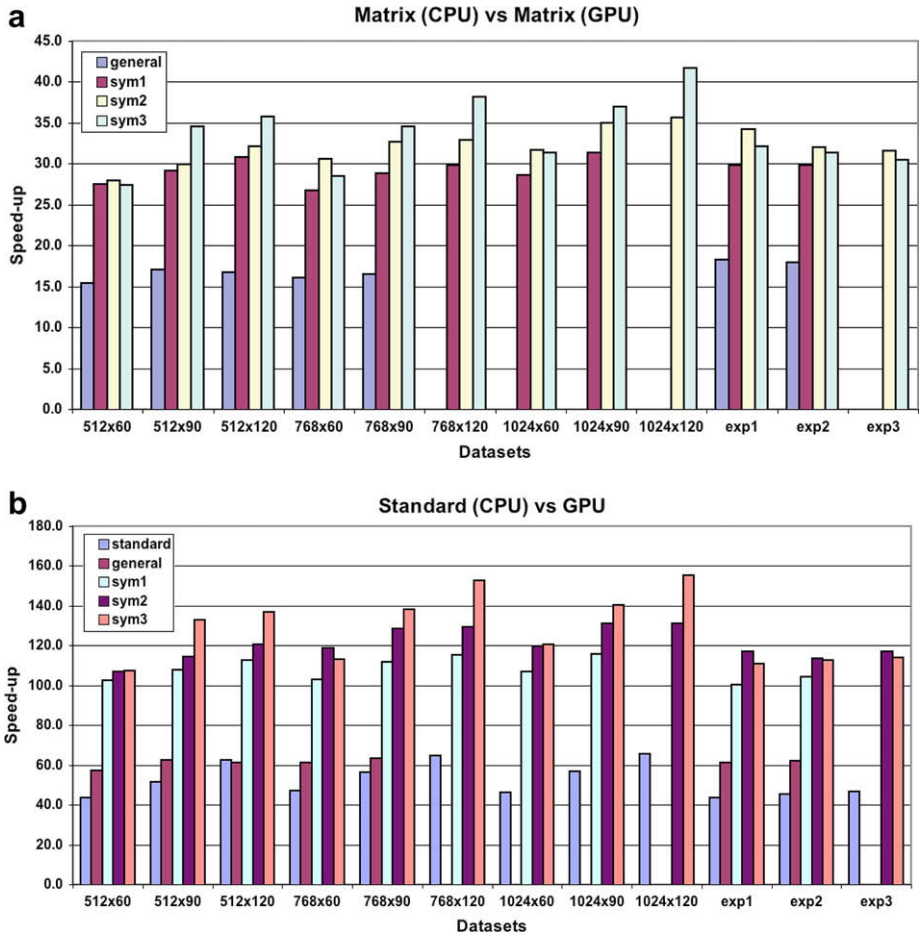

Fig. 5. (a) Speed-up factor of matrix WBP implementation on the GPU vs. CPU. (b) Effective speed-up derived from the different approaches (the standard based on recomputation and the four matrix approaches) on the GPU compared to the standard approach on the CPU.

WBP. They exhibit a trend similar – though scaled – to that shown in Fig. 5(a). And it is clearly seen that matrix WBP on the GPU yield excellent net speed-ups in the range 105×, −155×, depending on the symmetry level and the problem size. More importantly, all this acceleration factors lead to the capability of computing reconstructions of 1–4 GB in size in around 10 s, as Table 1 shows.

5. Discussion and conclusion

This work has addressed the tomographic reconstruction problem from a matrix perspective. The standard method, WBP, has been formulated as a set of sparse matrix–vector products, with the matrix being constant for all the slices in the volume. An optimized implementation for GPUs has also been developed. The results demonstrate that the matrix approach succeeds in reducing the computation time on CPUs and GPUs, with an acceleration factor of 4× and 2.5×, respectively. This difference in the acceleration rate between the two platforms comes from the extraordinary massively parallel computing capabilities of the GPUs, which also make them very fast for the standard implementation based on coefficient recalculation. The conjunction of matrix WBP and GPU computing yields significant acceleration compared to the standard implementation on CPU and is thus capable of providing tomograms in a few seconds.

Several geometry-related symmetry relationships have been exploited to reduce the matrix, which is important in GPUs due to their limited available memory. The highest level of symmetry (sym3) yields the best performance with a relatively large number of tilt angles. The level of symmetry 2 is an interesting option since (1) there is no prerequisite for the tilt angles, (2) fulfills the memory requirements of most ET problems, and (3) also provides good performance, especially for a low number of tilt angles.

This matrix approach can be easily applicable or extendible to other geometries or methods. It is suitable for double-tilt axis ET, where the same matrix would be exploited for the two axes. Iterative reconstruction methods, which are getting increasing interest in the field (Lucic et al., 2005), would benefit from this approach as well. In these methods, the projection operation can also be implemented as a sparse matrix–vector product, with the transpose of the matrix used for backprojection. This matrix approach thus opens up the possibility of implementing reconstruction methods with sparse linear algebra. Finally, the fact that tomograms can be quickly computed in the order of seconds makes this matrix approach suitable for real-time ET systems (Zheng et al., 2007; Suloway et al., 2009).

The code for matrix WBP will be available through the group's website. Although it was developed with NVIDIA CUDA, its implementation with OpenCL, the new standard for GPU programming, is straightforward. This will allow exploitation of the GPUs from different manufacturers.

Acknowledgments

The authors thank Dr. J.A. Martínez for technical help and discussions. Work supported by grants MCI-TIN2008-01117, JA-P06-TIC1426, CSIC-PIE-200920I075.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, at doi:10.1016/j.jsb.2010.01.021.

References

Bisseling, R.H., 2004. Parallel Scientific Computation. Oxford University Press.
Bruyant, P., 2002. Analytic and iterative reconstruction algorithms in SPECT. J. Nucl. Med. 43, 1343–1358.
Castano-Diez, D., Mueller, H., Frangakis, A.S., 2007. Implementation and performance evaluation of reconstruction algorithms on graphics processors. J. Struct. Biol. 157, 288–295.
Castano-Diez, D., Moser, D., Schoenegger, A., Pruggnaller, S., Frangakis, A.S., 2008. Performance evaluation of image processing algorithms on the GPU. J. Struct. Biol. 164, 153–160.
Fernández, J.J., 2008. High performance computing in structural determination by electron cryomicroscopy. J. Struct. Biol. 164, 1–6.
Herman, G.T., 2009. Image Reconstruction From Projections: The Fundamentals of Computerized Tomography, second ed. Springer, London.
Lucic, V., Foerster, F., Baumeister, W., 2005. Structural studies by electron tomography: From cells to molecules. Annu. Rev. Biochem. 74, 833–865.
Nickolls, J., Buck, I., Garland, M., Skadron, K., 2008. Scalable parallel programming with CUDA. ACM Queue 6, 40–53.
Radermacher, M., 2006. Weighted back-projection methods. In: Frank, J. (Ed.), Electron Tomography: Methods for Three-Dimensional Visualization of Structures in the Cell, second ed. Springer, pp. 245–273.
Schmeisser, M., Heisen, B.C., Luettich, M., Busche, B., Hauer, F., Koske, T., Knauber, K.H., Stark, H., 2009. Parallel, distributed and GPU computing technologies in single-particle electron microscopy. Acta Crystallogr. D 65, 659–671.
Suloway, C., Shi, J., Cheng, A., Pulokas, J., Carragher, B., Potter, C.S., Zheng, S.Q., Agard, D.A., Jensen, G.J., 2009. Fully automated, sequential tilt-series acquisition with leginon. J. Struct. Biol. 167, 11–18.
Vázquez, F., Garzón, E.M., Martínez, J.A., Fernández, J.J., 2009. Accelerating sparse matrix-vector product with GPUs. In: Proceedings of the International Conference of Computational and Mathematical Methods in Science and Engineering. CMMSE, pp. 1081–1092.
Zheng, S.Q., Keszthelyi, B., Branlund, E., Lyle, J.M., Braunfeld, M.B., Sedat, J.W., Agard, D.A., 2007. UCSF tomography: an integrated software suite for real-time electron microscopic tomographic data collection, alignment, and reconstruction. J. Struct. Biol. 157, 138–147.