A Parallel Scheme for Three-Dimensional Reconstruction in Large-Field Electron Tomography

Jingrong Zhang[1,2], Xiaohua Wan[1], Fa Zhang[1], Fei Ren[1], Xuan Wang[3], and Zhiyong Liu[1]

[1] Key Lab. of Intelligent Information Processing and Advanced Computing Research Lab., Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Yanshan University
zhangjingrong1990@gmail.com, {wanxiaohua,zhangfa,renfei,zyliu}@ict.ac.cn, wangxuan@ysu.edu.cn

Abstract. Large-field high-resolution electron tomography enables visualizing detailed mechanisms under global structure. As field enlarges, the processing time increases and the distortions in reconstruction become more critical. Adopting a nonlinear projection model instead of a linear one can compensate for curvilinear trajectories, nonlinear electron optics and sample warping. But the processing time for the reconstruction with nonlinear projection model is rather considerable. In this work, we propose a new parallel strategy for block iterative reconstruction algorithms. We also adopt a page-based data transfer in this strategy so as to dramatically reduce the processing time for data transfer and communication. We have tested this parallel strategy and it can yield speedups of approximate 40 times according to our experimental results.

Keywords: Electron tomography, Three-dimensional reconstruction, Iterative methods, Nonlinear projection model, TxBR.

1 Introduction

In electron tomography (ET), the specimen is tilted within a limited range $[-60°, 60°]$ or $[-70°, 70°]$ in small increment of $1 - 2°$ or so. During the process of taking projection images, electron beams impinge upon the specimen and penetrate it. ET can reconstruct a specimen's three-dimension (3D) internal structure from these projections. Now ET plays a crucial role in studying macromolecular assemblies. Especially, large-field high-resolution ET allows visualizing and understanding global structure such as organelles, membranes and microfiber networks extending throughout the cell and into intercellular spaces [1]. The development of hardware and techniques has made large-field high-resolution ET possible [2, 3].

Now the size of projection image has reached to 8192*8192 or larger. Then the size of final reconstruction volume will reach several GBytes [4]. In large-field ET, there are still problems to acquire high-quality reconstruction results. Projection images in ET are extremely noisy because of low signal-to-noise ratio (SNR). Furthermore, projection images are not complete at limited angle, which can lead to artifact during reconstruction. Different from traditional back-projection reconstruction algorithms, iterative methods have good performance in handling incomplete and noisy data. Many iterative methods, e.g. SIRT [5], BICAV [6] and ASART [7] have been adopted in three-dimensional (3D) reconstruction of ET. As image sizes increase, distortions in reconstructions become more pronounced because electron trajectories are helical under the influence of magnetic fields. To decrease these distortions, a global nonlinear projection model is proposed and has been already used in TxBR [8] and iterative methods [2].

However, the curvilinear projection model increases the complexity of calculation and extends the processing time. Meanwhile, iterative methods for 3D reconstruction in large-field ET are time-consuming compared with back-projection algorithms. Here to cope with the computational problem, parallel processing has been applied. Parallel strategies on clusters [2, 3, 8] have been widely used in 3D reconstruction of ET with curvilinear projection model to reduce turn-around times. These parallel strategies consider each projection map as a set of nonlinear transforms on z-sections which sum to produce the image, and the sections along Z-axis are reconstructed separately on different processors.

Graphic Processing Units (GPUs) have been widely used to accelerate scientific applications. Unlike clusters, these desktop supercomputers can obtain significant speed-ups on relatively inexpensive hardware with impressive performance-per-watt. The parallelization on GPUs is difficult for large-field data owing to the limited storage of memory in GPUs. For example, the global memory of GTX 480 graphic card is 1.5GB. However, the raw projection data and the final 3D reconstruction are approaching 50GB and 200GB respectively, if each image size is 8K*8K. TxBR adopts a parallel scheme to calculate 3D reconstruction using curvilinear projection models and backprojection algorithms on GPUs [3]. In this parallel scheme, all the volume is divided into several slabs including several Z-sections along Z-axis and each slabs are reconstructed on GPUs sequentially. But for iterative reconstruction methods, we usually need many iterations to achieve good reconstructions. Using the previous parallel schemes, we have to repeatly transfer all the sections into the memory of GPUs for each iterative step, which is rather time-consuming.

In this work, we describe a variant block-iterative version of SIRT method and implement its parallelization on GPUs. Our contribution includes two aspects. First, we propose a new Block-iterative SIRT parallel algorithm (BSIRT) with curvilinear projection model. We analyze the locality of curvilinear trajectory and then divide the data vertically according to the locality. Secondly, we adopt a page-based data transfer scheme in order to reduce the time for data transfer.

The paper is laid out as follows. Section 2 overviews iterative reconstruction algorithms, curvilinear model and previous GPU parallel strategies. Section 3 describes variant block-iterative version of SIRT methods (BSIRT) with curvilinear model. Then the implementation details of page-based data transfer scheme will be introduced in Section 4. Finally, we will show and evaluate our experimental results in Section 5.

2 Related Work

As our work focus on the iterative methods and its parallelization on GPU with curvilinear projection model. In this section, we first overview the iterative algorithms and introduce the curvilinear projection model. Then, we will review the previous parallel strategies.

2.1 Iterative Reconstruction Methods and Curvilinear Projection Model

In ET, the reconstruction problem is to obtain the internal structure of specimen by projection series. Working in real-space, the iterative methods solve the 3D reconstruction problem by formulating it as a large system of linear equations. Assuming the voxel as basis function to represent the volume, we present the result by the value of N ($N = n_{\text{width}} * n_{\text{length}} * n_{\text{height}}$) voxels. We suppose that the total number of projection pixels is M ($M = n_{\text{pro\_width}} * n_{\text{pro\_length}} * n_{\text{ang}}$), the projection procedure can be simply represented as follows:

$$p_i = \sum_{j=1}^{N} A_{ij} s_j \qquad 1 \le i \le M \tag{2.1}$$

where p_i is the value of ith projection pixel, s_j is the value of jth voxel and A_{ij} in matrix A indicates the contribution of the voxel j to the projection i. We can calculate matrix A according to projection model. We use (x, y, g) to indicate the projection point, where g is the index of orientation and $i = g * n_{\text{pro\_width}} * n_{\text{pro\_length}} + y * n_{\text{pro\_width}} + x$. We can use iterative algorithms to calculate the value of voxels $S = \{s_1, s_2 \dots s_N\}$.

Iterative methods can be generally classified into sequential, block-iterative and simultaneous methods [9]. In essence, the sequential and simultaneous algorithms are special cases of block iterative reconstruction [10]. Suppose all the equations of linear system may be subdivided into B blocks each of size T, we use a generalized version of iterative methods to describe the iterative step:

$$s_j^{k+1} = s_j^k + \lambda_k \sum_{i \in BLOCK_b} \frac{A_{ij}}{\sum_{v=1}^{N} w_v^b A_{iv}^2} (p_i - \sum_{w=1}^{N} A_{iw} s_w^k) \qquad 1 \le j \le N \tag{2.2}$$

where $b = k \bmod B$ is the index of block, i is the index of equation of system and w_v^b is the weighting factor [11]. The relaxation parameter λ_k is critical for convergence speed, usually it is found by training or experimenting [6, 12].

For the sequential iterative algorithms ($B = M, T = 1$), the equations are considered one-by-one in a circular manner. If the block number $B = 1$, the algorithm turns into simultaneous iterative algorithm. Simultaneous Iterative Reconstruction Technique (SIRT) [5] is typical simultaneous iterative algorithm. Block iterative algorithms update estimations by a subfamily of constraints in each iterative step. The main iteration can proceed sequentially from block to block and within each block in parallel. Traditional block iterative methods adopt a view-by-view strategy, the size of each block is $T = n_{\text{pro\_width}} * n_{\text{pro\_length}}$ and the block number is $B = n_{\text{ang}}$.

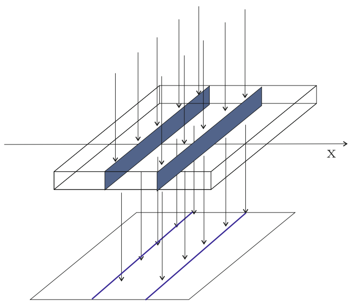
The projection step is formulated as Eq. (2.1). As mentioned, the value of matric A is decided by projection model, which describes the correspondence between projection points and voxels in object. Traditional straight-line projection model formulates projection map as a linear function. Here we adopt a quadratic curvilinear projection map. This model has been used in [2,8]. All the coefficients a_i^θ and b_i^θ are calculated by means of bundle adjustment, the details of the procedure is in [8]. The general quadratic expression is:

$$\begin{aligned} x &= a_0^\theta + a_1^\theta X + a_2^\theta Y + a_3^\theta Z + a_4^\theta X^2 + a_5^\theta XY + a_6^\theta XZ + a_7^\theta Y^2 + a_8^\theta YZ + a_9^\theta Z^2 \\ y &= b_0^\theta + b_1^\theta X + b_2^\theta Y + b_3^\theta Z + b_4^\theta X^2 + b_5^\theta XY + b_6^\theta XZ + b_7^\theta Y^2 + b_8^\theta YZ + b_9^\theta Z^2 \end{aligned} \tag{2.3}$$

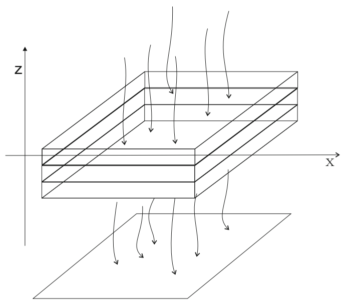
2.2 Previous Parallel Strategy on GPU

As for linear projection model, it assumes electron beams travel in straight line, so their trajectories are certainly parallel with each other and the slice perpendicular to tilting axis (in this paper we suppose the tilting axis is X) always projects in a straight line at each angle as shown in Fig. 1(a). Then 3D reconstruction problem can be decomposed into a set of independent 2D reconstruction problems [4, 13–15]. Using the reconstruction methods mentioned above, 2D slice reconstruction can be computed from a set of 1D projections (so-called sinogram [4]). But this strategy is not adaptable for curvilinear projection models because the curvilinear trajectories are not in parallel.

We need to divide data and reconstruct each part sequentially because of limited global memory on GPU and large data. TxBR [8] has proposed a GPU parallel strategy for direct reconstruction with curvilinear projection model. By regarding each projection map as a set of nonlinear transform on z-sections which sum to produce the image, it divide the object along the z-axis (see Fig. 1(b)) and reconstruct each z-section separately. Iterative algorithm ASART has implemented the parallelization on clusters [2], using similar strategy. When we use iterative algorithm, we need all the volume in one iterative step. Because of the limited storage of memory in GPUs, we need to transfer all the sections into the memory of GPUs for each iterative step, which is very time-consuming.



(a) Parallel strategy for straight-line projection model



(b) Parallel strategy for curvilinear projection model

Fig. 1. Parallel strategy for 3D reconstruction in electron tomography

3 Block-Iterative SIRT Algorithm with Curvilinear Model

Since previous algorithms are not suitable for iterative reconstruction with curvilinear model on GPUs, we need to find a new parallel algorithm. As GPU's global memory is limited for large-field reconstruction, we can consider a scheme for data partition. We analyze the locality of curvilinear trajectory to divide data. Then by modifying traditional SIRT algorithm and using a new data-divided scheme, we can effectively reduce the time for data transfer.

3.1 Locality of Curvilinear Trajectory

As discussed above, the projection of a slice is a straight line at each view if we adopt a straight-line model. However, with a curvilinear projection model, we can get different curves as the projections of a slice at each view. Conversely, for a straight line as a projection, its corresponding voxels are formed into a curved surface (see Fig. 2), which varies with the orientation. The curved surface covers a few slices. To a certain extent, it still owns locality. We can consider to analyze the locality of the curved surface and then calculate the precise scope of X coordinates of the curved surface.

Suppose that x is constant (i.e. $x = c$, c is constant) in Eq. (2.3), we can get the expression of the curved surface mentioned above:

$$c = a_0^\theta + a_1^\theta X + a_2^\theta Y + a_3^\theta Z + a_4^\theta X^2 + a_5^\theta XY + a_6^\theta XZ + a_7^\theta Y^2 + a_8^\theta YZ + a_9^\theta Z^2 \quad (3.1)$$

For these voxels on this curved surface, the x coordinate of their projection points is c. Here we can get the range of X by calculating the global maximum and minimum of curved surface within a certain domain ($D = \{(X, Y, Z)|X \in [x_{min}, x_{max}], Y \in [y_{min}, y_{max}], Z \in [z_{min}, z_{max}]\}$). $x_{min}, x_{max} \ldots z_{max}$ are constant.

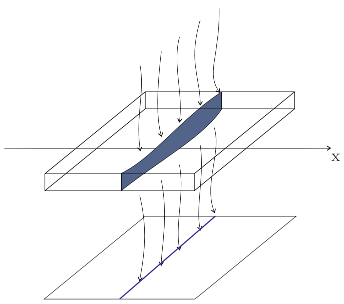


Fig. 2. Curvilinear projection model

Eq. (3.1) is an implicit function. X is dependent variable, Y and Z are independent variables. The global maximum and minimum should be a local maximum and minimum in the interior of the domain or a point on the boundary of the domain. So we can calculate these values and then choose the minimum and maximum among these values as the scope of X.

Theorem 1. *For a continuously differentiable function of several real variables, a point P is critical if all of the partial derivatives of the function are 0 at P.*

According the Theorem 1, we can use the implicit differentiation to get the partial derivatives and get the critical points in the interior of the domain.

$$\begin{cases} c = a_0^\theta + a_1^\theta X + a_2^\theta Y + a_3^\theta Z + a_4^\theta X^2 + a_5^\theta XY + a_6^\theta XZ + a_7^\theta Y^2 + a_8^\theta YZ + a_9^\theta Z^2 \\ \frac{\partial X}{\partial Z} = 0 \\ \frac{\partial X}{\partial Y} = 0 \end{cases} \tag{3.2}$$

Firstly, we calculate the critical points of intersecting lines. The curved surface may intersect with plane $Z = z_{min}$, $Z = z_{max}$,$X = x_{min}$, $X = x_{max}$, $Y = y_{min}$ and $Y = y_{max}$. Secondly, we consider the terminal point of the intersecting using the similar method. For example, the critical point of intersecting line (with plane $Z = z_{max}$) is calculated like:

$$\begin{cases} c = a_0^\theta + a_1^\theta X + a_2^\theta Y + a_3^\theta Z + a_4^\theta X^2 + a_5^\theta XY + a_6^\theta XZ + a_7^\theta Y^2 + a_8^\theta YZ + a_9^\theta Z^2 \\ Z = z_{max} \\ \frac{\partial X}{\partial Y} = 0 \end{cases} \tag{3.3}$$

According to this procedure, we can get the scope of X. Suppose the result is $\{MAX_c^\theta, MIN_c^\theta\}$. Absolutely, this result is different for different views. Next we extend the range of projection points to $\{(x, y, \theta)|x \in [c_1^\theta, c_2^\theta]\}$, c_1^θ and c_2^θ is constant, we can get the corresponding X coordinates of voxels by:

$$\begin{aligned} X_{min} &= min\{MIN_i^\theta, c_1^\theta \le i \le c_2^\theta\} \\ X_{max} &= max\{MAX_i^\theta, c_1^\theta \le i \le c_2^\theta\} \end{aligned} \tag{3.4}$$

Finally, if we consider a scope of orientations instead of one orientation, we can also get the scope of X for different views.

3.2 Block-Iterative SIRT (BSIRT) Algorithm

Different from SIRT, block iterative methods update estimations by a subfamily of constraints in each step. The iteration proceeds from block to block and usually the members of a block are selected sequentially (like SART, iterate view by view). In a straight-line projection model, the 3D object can be divided into several slices and each slice can be reconstructed separately. In a curvilinear projection model, we can divide the object into slabs according to the locality of curvilinear trajectory.

First of all, we equally divide the projection series along the tilting axis into B parts. Each part can be formulated as mathematical set S_t.

$$\begin{array}{l} S_t = \{(x, y, g) | x \in (c_t, c_{t+1}], y \in [1, n_{\text{pro\_length}}], g \in [1, n_{\text{ang}}]\} \\ c_t = \frac{n_{\text{pro\_width}}}{B} * t \qquad t = 0, 1, 2 \ldots B-1 \end{array} \tag{3.5}$$

We can calculate the scope of X in each corresponding 3D sub-volume D_t for each S_t according to the scheme described in Section 3.1. So we can divide 3D volume into B parts. The equations for S_t is:

$$p_i = \sum_{j \in D_t} A_{ij} x_j \qquad i \in S_t \tag{3.6}$$

In each iterative step, we only need the data in sub-volume D_t instead of all the volume. As shown in Fig. 3, in the first step, we reconstruct the first sub-volume D_1 from the first part S_1 of projection images. This strategy only need to transfer data one time for one update step. The size of each sub-volume is decided by the range of projection points and the number of orientations.

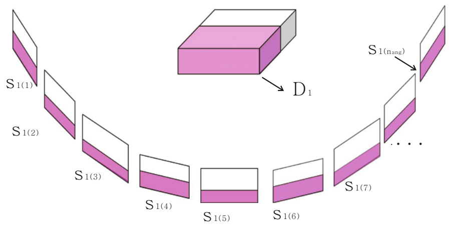


Fig. 3. Block-iterative SIRT algorithm

4 Page-Based Data Transfer Scheme

We also notice another problem: there is overlap between adjacent sub-volumes. If each sub-volume D_t is reconstructed and then transferred in each update step, the overlapping data will be transferred repeatedly. We propose a page-based data transfer scheme to eliminate the redundant data transfer. In this scheme, the overlapping data will not be transferred from the memory in the iterative steps until it is not reconstructed. Every time, we transfer useless data out of GPU and new data in GPU.

In case that the size of incoming data is bigger than the outgoing, we divide the global memory into several pages. Since the size of sub-volume D_t is different, we should calculate the size of outgoing and incoming data according to the distribution of sub-volumes. According to this, we can define the biggest number as page size.

An example is listed in Table 1. The coordinate scope of reconstruction object is $\{(X, Y, Z)|X \in [-87, 563], Y \in [-31, 652], Z \in [-62, 38]\}$. The tilting axis is X and the size of projection is $512 * 512$. We divide the projection series into 8 groups ($B = 8$). As we partition the object vertically according to the value of X, Table 1 shows that the size of X for each slab. For the first slab, after the first iterative step, we need to move out the part whose scope of X is from -78 to -12. The size of the transferred data is $(-12-(-78)+1)*651*101 = 67*651*101$. We use 67 to indicate the size of data moved out in column "out" of Table 1. We calculated the sizes of data moved out and in for each slab and found that these values are almost the same. So we can choose the maximum $(69 * 651 * 101)$ as the size of page.

Table 1. The distribution of slabs

	start position of X	end position of X	overlap range of X	out	in	total size
1st slab	-78	193	205	67	0	272
2nd slab	-11	261	206	67	68	273
3rd slab	56	329	205	69	68	274
4th slab	125	398	206	68	69	274
5th slab	193	467	207	68	69	275
6th slab	261	536	208	68	69	276
7th slab	329	563	168	67	27	235
8th slab	396	563	0	0	0	168

Next, we repartition the voxels according to the start point of each slab (shown in Table 2, the page 1 to 7). After the start points are considered, the rest of data is partitioned according to the page size (page 8,9,10).

Table 2. The distribution of pages

	start position of X	end position of X	range of X
1st page	-78	-12	67
2nd page	-11	55	67
3rd page	56	124	69
4th page	125	192	68
5th page	193	260	68
6th page	261	328	68
7th page	329	395	67
8th page	396	464	69
9th page	465	533	69
10th page	534	563	30

The whole procedure is described as follows. In the first iterative step, the first slab including several pages (from page 1 to 5) is moved in the memory of GPU. In the following iterative step, we need to move out the previous unnecessary page of data and move in the next page of data. Basically, the size of data in the memory of GPU is a little larger than that of a slab, which could ensure all the data required can be stored in GPU. When the end of data is in GPU, we can stop transfer data out until this round finish.

5 Result

In this section, we outline the experimental results. As we proposed a variant version of SIRT (called BSIRT) and studied its parallel scheme. First of all, we compare the performance of BSIRT and SIRT, show their reconstruction result. Next, we report the timing performance of SIRT and BSIRT.

The benchmark used in this section has been adopted by paper [2] and its partition has been discussed in Section 4. The thickness of sample is 350nm, the micrographs were taken in a 300kV FEI Titan TEM with a 37k magnification. The tilt series are composed of 121 micrographs, taken from $-60°$ to $+60°$. The size of each micrograph is $512 * 512$ and the size of reconstruction result is $651 * 684 * 101$.

All the experiments are carried out on machine running the Ubuntu 12.04 operating system 64-bit and the GPU card we use is NVIDIA GTX480, which owns 480 SPs and 1536M global memory.

5.1 Reconstruction Result

All the experiments are performed using quadratic projection model. Here we have considered three conditions: the first one updates the data using traditional SIRT, the second one updates the data using BSIRT parallel algorithm with 4 slabs, the third one uses BSIRT with 8 slabs. Note that BSIRT with one slab is identical to SIRT, which has been testified by experiment. In our experiment, a combination of two factors, relaxation parameter and number of iterations, need to be considered. We define the term “one iteration” as one whole sweep through all equations of the system. The relaxation parameter λ keeps constant throughout the iterations. Since the convergence of SIRT is guaranteed [16] as long as $0 < \lambda < 2$. Here we set the relaxation parameter $\lambda = 0.1$.

In our study, we applied projection error to compare the quality of the reconstructed images, which is based on the discrepancy between the experimental images and the images calculated by reprojection. Specifically, we use the mean absolute error (MAE) to calculate the projection error:

$$MAE = \frac{1}{M} \sum_{i=1}^{M} |p_i - p_i'| \tag{5.1}$$

where p_i is experimental projection images and p_i' is the calculated projection images. The line plots given in Fig.4 show these measures versus the number

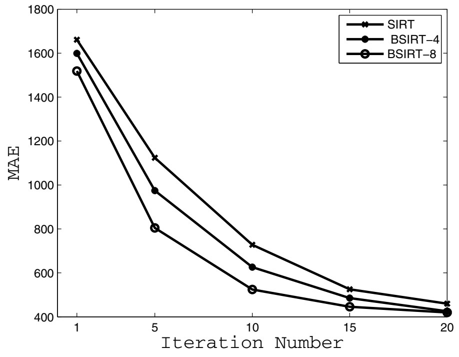
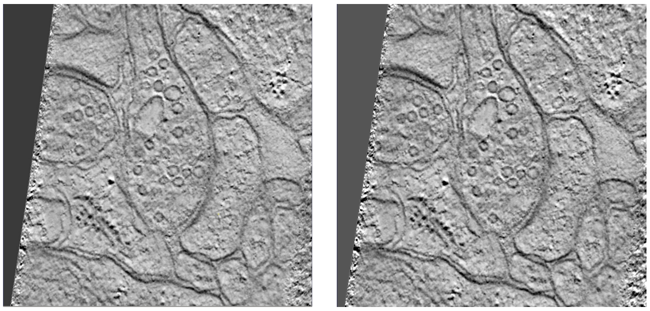


Fig. 4. Projection error

of iterations when the relaxation parameter $\lambda = 0.1$. We can see the projection error of BSIRT (8 slabs) is smaller than that of BSIRT (4 slabs) in the same iterations and the projection error of BSIRT with 4 (and 8) slabs is smaller than SIRT, especially in the first 10 iterations. It proves that BSIRT enjoys a faster convergence speed.

Fig. 5 shows the images produced after 5 iterations by SIRT and BSIRT (8 slabs). The result of BSIRT is similar to SIRT. As our algorithm is intrinsically a block-iterative algorithm, with a proper relaxation ratio, our strategy will acquire good result.



(a) SIRT + quadratic map (b) BSIRT + quadratic map

Fig. 5. Reconstruction result comparison

5.2 Performance of Parallelization

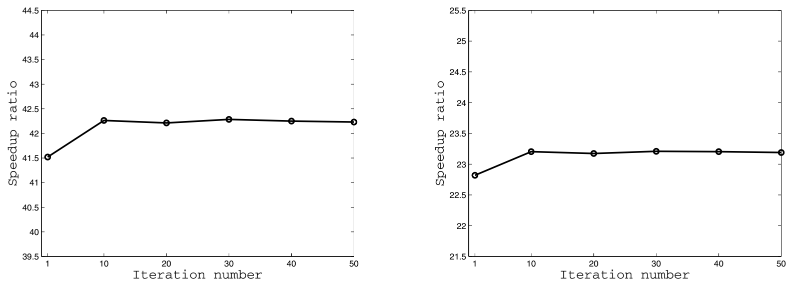
We have timed the BSIRT code on GTX 480 using CUDA 5.0. BSIRT is test by dividing the object into 1 (SIRT), 4 and 8 slabs. We use two-dimensional blocks to accomplish parallelization. The blocks in the same row are in charge of one page. The running time is shown in Table 3. We can see it is basically in proportion to iteration number. The running time of BSIRT with 4 slabs is generally less than BSIRT with 8 slabs and its growth rate is smaller. The reason

Table 3. Running time (sec)

Iteration number	1 slab (SIRT)	4 slabs	8 slabs	CPU-only
1	24.022	31.641	57.572	1313.78
10	236.291	312.907	569.932	13224.44
20	472.321	625.321	1139.072	26396.36
30	711.686	937.785	1708.491	39653.17
40	947.617	1250.752	2277.419	52843.66
50	1183.486	1563.276	2846.961	66020.13

is less slabs means greater degree of parallelism and less communication cost. So if the hardware condition permits, the less slabs the better.

Here we calculate the speedup ratio by comparing with the CPU-only program, which is shown in Fig. 6. From this figure, we can see the speedup ratio is about 23 and 41 for BSIRT (8 slabs) and BSIRT (4 slabs).



(a) 4 slabs (b) 8 slabs

Fig. 6. Speedup Ratio

6 Conclusion

In this work, we develop a variant version of SIRT (called BSIRT) and complete its parallel scheme on GPUs. With this method, 3D reconstruction for large-field electron tomography can be calculated on GPUs even though the memory of GPUs is limited. Meanwhile, we proposed a page-based data transfer scheme, which can reduce the data transfer time in implementation stage. Now our algorithm is based on SIRT, but this parallel strategy can be applied to all kinds of block-iterative algorithms. For the future work, we are considering that this parallel scheme can be extended to other parallel platforms and the influence of relaxation parameter is still to be pending to be discussed as well.

Acknowledgments. This work is supported by grants National Natural Science Foundation for China (61232001, 61202210, 61103139 and 60921002). The authors would like to thank Albert Lawrence for his instruction on the mathematical and computational problems. The authors must also thank Sébastien Phan for providing the experimental dataset.

References

1. Phan, S., Lawrence, A., Molina, T., Lanman, J., Berlanga, M., Terada, M., Kulungowski, A., Obayashi, J., Ellisman, M.: Txbr montage reconstruction for large field electron tomography. Journal of Structural Biology 180, 154–164 (2012)
2. Wan, X., Phan, S., Lawrence, A., Zhang, F., Han, R., Liu, Z., Ellisman, M.: Iterative methods in large field electron microscope tomography. SIAM Journal on Scientific Computing 35(5), S402–S419 (2013)
3. Lawrence, A., Phan, S., Singh, R.: Parallel processing and large-field electron microscope tomography. In: World Congress on Computer Science and Information Engineering, pp. 339–343 (2009)
4. Agulleiro, J.I., Fernández, J.J.: Evaluation of a multicore-optimized implementation for tomographic reconstruction. PloS One 7(11), e48261 (2012)
5. Gilbert, P.: Iterative methods for the three-dimensional reconstruction of an object from projections. Journal of Theoretical Biology 36(1), 105–117 (1972)
6. Censor, Y., Gordon, D., Gordon, R.: Bicav: A block-iterative parallel algorithm for sparse systems with pixel-related weighting. IEEE Transactions on Medical Imaging 20(10), 1050–1060 (2001)
7. Wan, X., Zhang, F., Chu, Q., Zhang, K., Sun, F., Yuan, B., Liu, Z.: Three-dimensional reconstruction using an adaptive simultaneous algebraic reconstruction technique in electron tomography. Journal of Structural Biology 175(3), 277–287 (2011)
8. Lawrence, A., Bouwer, J.C., Perkins, G., Ellisman, M.H.: Transform-based backprojection for volume reconstruction of large format electron microscope tilt series. Journal of Structural Biology 154(2), 144–167 (2006)
9. Censor, Y.: Parallel optimization: theory, algorithms and applications. Oxford University Press (1997)
10. Aharoni, R., Censor, Y.: Block-iterative projection methods for parallel computation of solutions to convex feasibility problems. Linear Algebra and Its Applications 120, 165–175 (1989)
11. Bilbao-Castro, J.R., Carazo, J.M., Fernandez, J.J., García, I.: Parallelization and comparison of 3d iterative reconstruction algorithms. In: Proceedings of the 12th Euromicro Conference on Parallel, Distributed and Network Based Processing, pp. 96–102. IEEE (2004)
12. Herman, G.T., Meyer, L.B.: Algebraic reconstruction techniques can be made computationally efficient. IEEE Transactions on Medical Imaging 12(3), 600–609 (1993)
13. Wan, X., Zhang, F., Chu, Q., Liu, Z.: High-performance blob-based iterative reconstruction of electron tomography on multi-GPUs. In: Chen, J., Wang, J., Zelikovsky, A. (eds.) ISBRA 2011. LNCS, vol. 6674, pp. 61–72. Springer, Heidelberg (2011)
14. Fernandez, J.: High performance computing in structural determination by electron cryomicroscopy. Journal of Structural Biology 164(1), 1–6 (2008)
15. Castaño Díez, D., Mueller, H., Frangakis, A.S.: Implementation and performance evaluation of reconstruction algorithms on graphics processors. Journal of Structural Biology 157(1), 288–295 (2007)
16. Van der Sluis, A., Van der Vorst, H.: Sirt-and cg-type methods for the iterative solution of sparse linear least-squares problems. Linear Algebra and its Applications 130, 257–303 (1990)