

# Matrix Approach for Processing of Iterative Reconstruction on Cone Beam CT

Chen Jian-lin<sup>1</sup>, Zhang Han-ming<sup>1</sup>, Yan Bin<sup>1\*</sup>, Li Lei<sup>1</sup>, Guan Ming<sup>2</sup>, Wang Lin-yuan<sup>1</sup>

1. National Digital Switching System Engineering & Technology Research Center, Zhengzhou, 450002, China

2. He Nan Province People's Hospital, Zhengzhou, 450002, China

E-mail: [tom.yan@gmail.com](mailto:tom.yan@gmail.com)

**Abstract:** Cone-beam computed tomography (CBCT) is an important technique providing new insights into the inner structure of products in industry and medicine physics. Iterative reconstruction methods have been shown to be more robust than analytical algorithm against the noise and limited angles conditions present in CT. Nevertheless, these methods are not extensively used due to their computational demands. In the iteration algorithm, the matrix of projection is massive and it is very time-consuming to calculate the forward projections and back-projections. In this work, we design a matrix approach that the coefficients of the projection matrix are pre-calculated and simultaneously stored with two compressing formats due to the different sparse structures of the matrix and its transposed matrix. And we implement the corresponding SpMV (sparse matrix-vector multiplication) based on the compressing matrices with GPU platform to realize the acceleration. Experimental results indicate that this method allows efficient implementations of reconstruction in CBCT and it can have a better performance than those with serial computing on CPU.

**Key words:** Iterative reconstruction, forward projection, back-projection, matrix approach, SpMV

## 1 INTRODUCTION

Cone-beam CT currently is a highlight in CT research. It takes advantage of high efficiency of X-ray, faster scanning speed, more flexible scanning scope and higher image resolution. Iterative reconstruction methods [15] have gathered significant interest in recent years since they can cope well with limited projection sets and noisy data, but the methods demand huge computational costs. Cone-beam CT reconstruction needs to propose big data and iteration algorithm will come across the hardware and time-consuming problem and it restricts the development of the implementation.

Traditionally, high-performance computing has been used to address such computational requirements by means of parallel computing on supercomputers, large computer clusters and multicore computers. Graphics processing unit (GPU) offers an attractive alternative platform to cope with the demands in CT in terms of the high peak performance, cost effectiveness, and the availability of user-friendly programming environments, e.g. NVIDIA CUDA [9].

At present, two main methods are presented to solve the problem of large sparse matrices processing in the iteration reconstruction. The first method is the standard implementation based on recalculation of the matrix coefficients (i.e. the projection matrix coefficients are calculated on-the-fly every time they are needed) [3]. Although computing the weight matrix on the fly is more efficient than storing the matrix in the previous GPU-based CT implementations, it could bring the redundant computations since the weighted matrix has to be computed twice at least in each iteration. When the matrix system is complex, it may obtain a poor performance on the parallel platform. The second method is to use the SpMV (Sparse Matrix-Vector

Multiplication) to compress the matrix [5] [6]. Nowadays, the memory available in modern computers allows storage of large matrices, which may improve the performance of the algorithms. The method is effective for practice reconstruction and it is a hot pot for processing big data. Nevertheless, sparse matrix data structures must be carefully devised to optimize the access to the memory hierarchies.

In this paper, we mainly study how to use the sparse nature of the projection matrix to solve the problem that a large number of space and time are consumed. We provide an appropriate method to reduce the matrix storing space and realize fast projection calculation. Furthermore, we implement all the above techniques on the GPU platforms: NVIDIA Tesla C1060. Experimental results show that the parallel strategy greatly reduces memory requirements and exhibits a significant acceleration.

## 2 ITERATIVE RECONSTRUCTION

Cone-beam CT reconstruction algorithm is to establish the algebraic equations by the relationship between the reconstructed image and the projected image established by the imaging process and the corresponding mathematical model. Fig.1 shows in CBCT the X-rays pass through the objects and the projection images are obtained by detectors. Iterative methods are based on the series expansion approach in which 3D volume is represented as a linear combination of a limited set of known and fixed basis functions, with appropriate coefficients. Coefficients of the equations are obtained by the imaging process calculation, and the projection images are obtained by the measurement. CT iterative reconstruction uses an iterative method to solve the algebraic equations for obtaining the reconstructed

image and it means that we should make continuous image discretization.

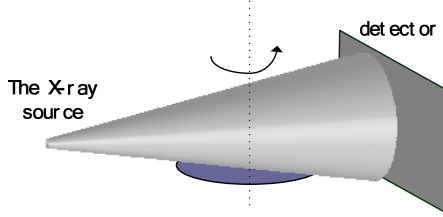
$$P_i = \sum_{j=1}^J A_{ij} x_j, 1 \leq i \leq n \quad (1)$$


Fig.1 circular trajectories in CT scanning

In the CT field, SART is a most popular iterative method that has got an excellent reputation. It is in detail described in the paper [2]. 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:

$$x_j^{(k+1)} = x_j^{(k)} + \frac{\sum_{i \in I_\theta} a_{ij} \frac{p_i - \sum_{m=1}^M a_{im} x_m}{\sum_{m=1}^M a_{im}}}{\sum_{i \in I_\theta} a_{ij}} \quad (2)$$

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

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

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

$$e^{(k)} = \frac{\frac{p_i - q_i^k}{\sum_{m=1}^M a_{im}}}{\sum_{i \in I_\theta} a_{ij}} \quad (4)$$

$$q_i^k = \sum_{j=1}^J a_{ij} x_j^{(k)}, i \in I_\theta \quad (5)$$

Equation (5) can be expressed in matrix form as:

$$q_i^k = A x_i \quad (6)$$

Equation (4) basically consists of simple component-wise vector operations. The other weights in from Equation (4) are calculated directly from the matrix  $A$  and are constant for all iterations. Equations (3)

and (6) show that one iteration of the reconstruction method requires two SpMV operations, with matrix  $A$  and its transpose  $B = A^T$ , respectively. The operations are forward projection and backprojection.

Iterative algorithms are based on the same scheme: they aim to reconstruct a 3D image from 2D projections. An estimated volume image is chosen. This volume is projected in the detector plan for each acquisition angle. Then estimated projections are compared with the acquired projections: correction coefficients are computed and applied to the estimated volume image. So the two operations are the main body in iterative algorithm.

#### A. Forward projection

This operator is the geometrical transformation from the volume space to the detector plan space. For the implementation, we need to compute the contribution of each voxel of the volume to the value of each pixel of the projection plan.

#### B. Backprojection

The backprojection is the geometrical transformation from the detector plane space to the volume space. We need to compute the contribution of each pixel of the projection plan to the value of each voxel of the volume.

### 3 MATRIX APPROACH TO ITERATIVE RECONSTRUCTION

#### 3.1 Matrix characteristic

According to the analysis above, the iterative algorithm should involve two operations. The matrix  $A$  involved is the forward projection operator while its transpose  $B = A^T$  involved is the backprojection operator. The two operations are the most computational cost in the iteration algorithm.

In our practical implementation, the matrix coefficients are computed by the classic siddon algorithm [13]. As the matrices have the sparse nature, we consider that it would be more effective to compress the matrices and design SpMV for the calculation of the forward projection and backprojection. And we think that the matrix coefficients can be pre-calculated and compressed for storage meanwhile. It would be a huge computation if we directly use the compressed matrix into the transpose operation. So during the calculation of the matrix coefficients we simultaneously store the projection matrix and its transposed matrix respectively with the compressing format.

Analysis for the case of the sparse structure of two-dimensional CBCT scanning projection matrix:

We can assume that when CBCT is scanning the x-ray source and the detector are kept fixedly, and the object is going round the rotating point while the system is collecting the projection data. And the left

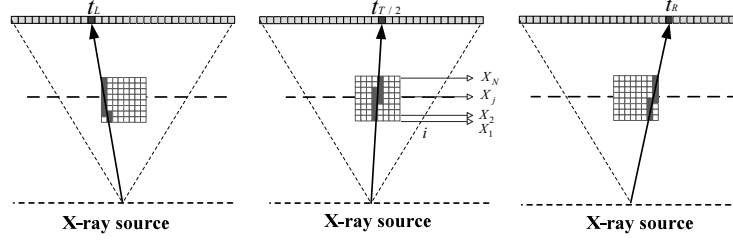


Fig 2 projection relations between rays and object in 2D CBCT scanning

probe  $t_L$  and the right probe  $t_R$  are segmented by the center line of the detector. The projection ray intersects the object as the Fig.2 shown. If the object voxels in the figure are arranged in sequence for each row  $X_j$  ( $j=1,2,...,N$ ), the distribution relationship of the projection value collected by the probe  $t_L$  corresponding to the data in the projection matrix can be shown in Fig.3, the same as the probe  $t_R$ .

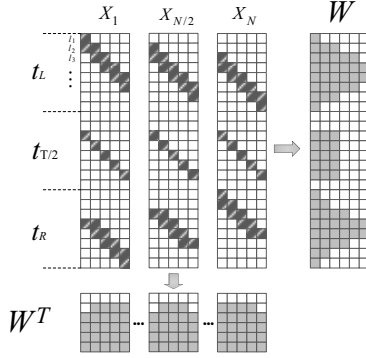


Fig.3 CBCT matrix structure of forward projection and backprojection

Nonzero value of CBCT projection matrix presents sparse diagonal distribution range as the Fig.3 shown. According to the analysis of the distribution of the nonzero value of forward projection and back projection, we can obtain the following characteristics:

- (1) The number of the nonzero value of forward projection matrix among each row varies considerably with the uneven distribution;
- (2) The number of the nonzero value of backprojection matrix among each row varies little with the mean distribution;

### 3.2 General compressing sparse matrices

Compressed row storage (CRS) [9] is the most extended format to store the sparse matrix on CPUs. It can be appropriate for storing any sparse matrix with different structures. Fig.4 shows the format of the storage structure. Subsequent non-zeros of the matrix rows are stored in contiguous memory locations and an additional integer arrays specify where each row begins. It assumes that there is no ordering among non-zero values within each row, but rows are stored in consecutive order. CRS format is fit for compressing

the forward projection matrix A.

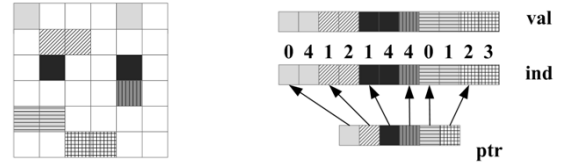


Fig.4 CRS storage formats for sparse matrices

ELL can be considered as an approach to outperform CRS. An improved approach was proposed based on ELL format in 2009 [8]. The format called ELLR-T has been proved to outperform the most efficient formats for storing the sparse matrix data structure. Fig.5 shows this storage format. ELLR consists of two arrays, *val* and *ind* of dimension  $N \times \text{Max Entries by Rows}$ , and an additional N-dimensional integer array called *rl* is included in order to store the actual number of non-zeros in each row. With the size and number of projection images increasing, the memory demand of the sparse weighted matrix rapidly increases as well. It is suitable for the matrix with average distribution of non-zeros elements in each row. We find that the matrix B is fit for compressing with this format.

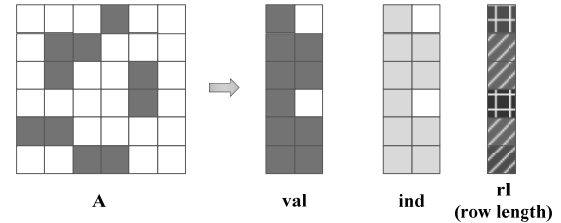


Fig 5 ELL-R storage formats for sparse matrices

There exist other compressing matrix formats, but we haven't found a format available both for projection and back projection. For example, the coordinate storage scheme (COO) to compress a sparse matrix is a direct transformation from the dense format by storing the total number of non-zero entries of the matrix. A typical implementation of COO uses three one-dimensional arrays including the matrix non-zero-value, row and column. The format matrix can be used to calculate both projection and back projection, but the parallel computing is not available especially in the back projection operation.

### 3.3 Our method for optimization computing

A. forward projection optimization based on block

### CSR format

We find that the given structure of forward matrix is not distributed equally. The number of non-zeros values in each row has varied greatly so that it would cause two problems in the GPU program.

(1) Unbalance of the threads of one warp. In NVIDIA's GPU, there are 32 threads on the hardware into a group known as a warp and the total execution consumption is decided by the longest running time in the thread of one warp. Therefore, imbalance caused by the threads of one warp will reduce the operating efficiency.

(2) No continuous memory access problems. Although the value of CSR format's data and column coordinates is stored contiguously, the access to the data of each thread is not continuous. Discontinuous visit will seriously affect the efficiency of the CUDA program.

To solve the problems above, a consolidation strategy called CSR (vector) method [9] was proposed. The method uses a warp to handle every row operation. Each thread in every warp has an independent operation and the separate calculation is accumulated. Finally each thread's calculation result of the parallel reduction is processed by the dichotomy method. It can effectively avoid the workload imbalance problem caused by thread waiting. Furthermore, since one warp processes a row of data stored continuously, as long as it is the set of the warp threads in the *thread\_id* (*thread\_id* = 0,1,2, ..., 31) to access the row of data in the *thread\_id* + 32 × *t* (*t* = 0,1,2, ...) data, it can ensure the continuity of the memory access. CSR (vector) method can effectively enhance the CSR format SpMV efficiency running on the GPU. But when the number of non-zero values in most of the matrix row is less than 32, the performance is still not satisfactory. The main reason is that if the amount of each row calculation generally is less than 32, only a minority of warp threads can be assigned to the task, resulting in a large number of threads in the idle state. A compromising approach is to use the half-warp (16 threads) combined means [14], but it can only lead to the performance improvement in a certain extent.

For the forward projection matrices in CBCT, there are a large number of rows of which the nonzero number is less than 32. Obviously, CSR (vector) method in the CBCT forward projection matrix is not very efficient. Therefore, we have designed a block CSR strategy. The original forward projection matrix with the rows containing the number of non-zero value is divided into two parts: the first part is the collection of rows of which the number of non-zero value is less than 16, and one thread is allocated to operate the data of a row; another part is the rows of which the number of non-zero value is greater than 16 and the part is treated with half-warp based on the CSR (vector) approach to treatment. Block CSR processing flow is shown in Fig.6.

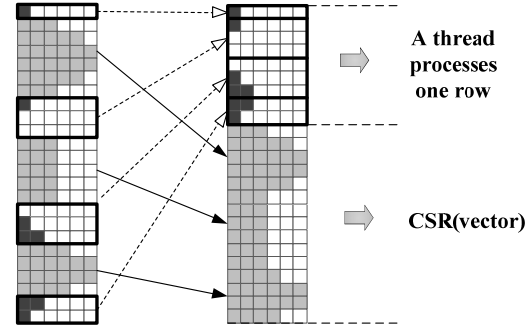


Fig.6 Block CSR format processing flow

Block CSR strategy needs to combine the rows with the same execution, but in the operation the data and column coordinates does not need to make changes. It simply changes the row offset *ptr* into two array *ptr\_old* and *ptr\_new*. According to the amount of data per row, the position of each row is classified again. The new offset information of the rows under the new order is stored as *ptr\_new*, and the original location information corresponding to each row is stored as *ptr\_old*, for completing the respective data addressing.

**B. Back-projection optimization based on ELLR-T**  
Because of the zero element supplement in the compression operation, ELL format sparse matrices exist some zero element. In order to reduce unnecessary computation cost, it usually needs to have the pre-treatment step before performing calculations of the data. It is used to judge whether the data is zero element and perform the calculation of non-zero element. However, the judgment operation performed on the GPU will cause greater overhead and affect the program's parallel performance. In order to reduce the overhead in judgment operation, the reference proposed [6] an improved ELL format called ELL-R format. Compared with ELL format, ELL-R format adds a one-dimensional vector *rl*, for storing the number of non-zero elements of each row. A small amount of storage space is increased to avoid the cost of pre-judging.

Based on the ELL-R format, the paper [10] has proposed a fast implementation on the GPU optimized called ELLR-T method. ELLR-T uses T threads in the calculation of the matrix, in order to meet the condition of coalescing accessing memory, as T is usually 16 (half-warp) multiple. In the realization, ELLR-T method and CSR (vector) method is basically similar and the goal is to take advantage of GPU hardware on a thread group (warp) on the coalescing global memory access capability so as to enhance the execution performance. Therefore, we adopts the optimization method ELLR-T on back-projection GPU parallel computing implementation, where the size of the selection for T is 16 and the size of block is 256.

### C. Memory access optimization

Two kinds of optimization methods based on the GPU storage access is adopted to realize vectorization and

texture memory bound:

(1) Vectorization refers to that the operation of the data use the vector type format as the storage format. We use float4 format to process float matrix data compressed and process the data array with int4 format in the SpMV kernel. By the operation of vectorization, a thread can access multiple data at the same time, such as one float4 data including four float data. If four arrays of data are to perform the same calculation, the data is accessed only once in float4 format and 4 float data manipulation can be performed at the same time, in order to make full use of the GPU memory bandwidth and enhance the execution efficiency.

(2) In the calculation of matrix vector multiplication  $Ax = b$ , the vector  $x$  accessing speed also affects the efficiency of the program. Due to the large accessing delay of global memory, we choose to bound the vector texture memory with the vector through the image object, in order to improve the accessing speed of vector  $x$ .

#### 4 RESULTS

This paper has presented a matrix method to solve the problem of the projection matrix coefficients calculated and storing simultaneously and the corresponding SpMV operation. In this part, we have implemented the matrix approach to SART algorithm by using the library for SpMV based on the CSR and ELLPACK-R format. It has been evaluated using several representative data sets on a CPU Intel Xeon 5520E (2.27GHz) and 24 GB SDRAM DDR3 at 1333 MHz, NVIDIA Tesla C1060, Visual Studio 2010+CUDA 4.0 platform, under Windows 7 was employed.

According to the different scale, reconstruction experiment were divided into two groups. Two data sets with one image of 256 pixels, 512 pixels were processed to volume of  $256 \times 256$  pixels, and  $512 \times 512$  pixels, respectively. And the performance of two SpMV was presented by the SART algorithm with iteration 1000 time in 180 angles. In the experiment, each group of three different methods were tested, and these methods are: A. Original CPU reconstruction; B. Based on SpMV CPU reconstruction; C. Based on SpMV GPU reconstruction. In the method A, different experimental methods were adopted for the implementation. In the group 1, the projection matrix is computed and stored in memory and it is released until the end of the iteration; In the group 2, block storage in the hard disk is adopt for projection matrix, and the matrix is read as it used.

Table 1 presents storing the data sets brings space-consuming with the compressing formats or not respectively. It shows that if the matrix is not compressed, the storage is amazingly huge. We can infer that the forward projection and backward operation will spend plenty of time because of the computation of the massive matrix without compressing. With the compressing formats, it has

reduced the most of space-consuming. When the data scale has enlarged, the compressing ratio will get very high because of the intensified sparsity. Table 2 presents the time-consuming with the three methods and the speed-up between the methods on CPU and on GPU. It shows that computing on GPU has got high speed-up than computing on both of methods on CPU. The result can prove that our method is fit for parallel computing and reduce the computing time.

Table 1 CSR&ELLR-T memory (megabytes)

	A	A with compress-ion	Com-pressing ratio	$A^T$	$A^T$ with compre-ss ion	Com-pressing ratio
$256 \times 256$	11.25GB	72.37 MB	158.1	11.25 GB	90MB	128.0
$512 \times 512$	180GB	547.84 MB	336.4	180 GB	655.36 MB	274.6

Table2 . CSR&ELLR-T time(s)

	CPU(original matrix)	CPU(SPMV)	GPU(SPMV)	Speed-up 1	Speed-up 2
$256 \times 256$	1653	159	52	31.8	3.6
$512 \times 512$	13668	960	170	80.4	5.64

Fig.7 shows the result of the  $256^2$  image reconstruction with three methods. The difference of method B and method A is between the projection matrix storage and operation, therefore two methods in the reconstruction accuracy of the results are consistent. Table 3 presents the RMSE from the reconstructed image and the contrast can be seen in the quality of the reconstructed image using the GPU and CPU. There exists the little difference of the reconstructed image quality, and the numerical accuracy of reconstruction result is maintained.

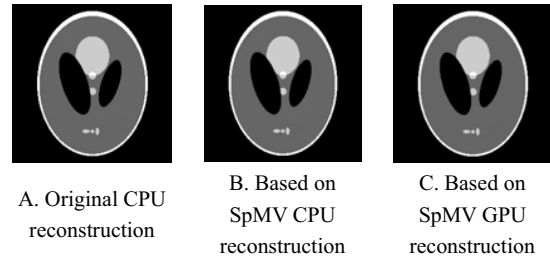


Fig 7. Reconstruction result with three methods

Table 3 The RMSE for the different methods

	A, B	C
1	7.666E-4	9.362E-4
2	5.566E-4	6.614E-4

#### 5 CONCLUSION

The reconstruction speed is an important index of the CT system, and it has important significance for the reconstruction algorithm to accelerate the research in enlarging the application of CBCT and improving the application effect of CBCT. In the paper, according to

the CBCT projection and sparse characteristics of reverse projection matrix, we use sparse matrix vector CSR format and ELL format respectively for the matrix compression and optimization of its computation. Then according to the nature of the SART iterative algorithm of each module, we achieve the optimized SpMV accelerated, increasing acceleration effect in the CPU environment for reconstruction algorithm through the use of CSR, ELL-pack optimization strategy. We apply it in the reconstruction of simulation data while the time-consuming and space-consuming can be acceptable and it take the quality of the reconstructed image compression method without loss. The compressing efficiency is higher as the data scale is bigger.

In the paper, we experiment with the basic environment of GPU on parallel computation. The method provides an approach to solve the iterative reconstruction forward projection and back projection problem with storing pre-calculated system matrix with compressing format. In the future, we will study a more effective compressing format with optimization to get better performance on GPU platform.

## REFERENCES

- [1] Herman, G.T. (2009) Image Reconstruction From Projections: The Fundamentals of Computerized Tomography (2nd edn).Springer, London.
- [2] Andersen AH . et al . Simultaneous Algebraic Reconstruction Technique(SART) :a superior implementation of the ART algoritbm. Ulwason[J]. Img.vol.01, No.06, PP 81-94, 1984.
- [3] WANG Xu, CHEN Zhi-qiang, XIONG Hua, ZHANG Li. Projection computation based on pixel in simultaneous algebraic reconstruction technique, Nuclear Electronics & Detection Technology, vol.25, No.06, 785-787,2005.
- [4] Wang J, Li T F, Xing L. Iterative image reconstruction for CBCT using edge-preserving prior[J]. Med.Phys, 36(1): 252-260.
- [5] Tilman K, Josef W, Jasmine S, et al. Parallel MLEM on Multicore Architectures[A]. International Conference on Computational Science:Part I[C], 2009, 491-500
- [6] 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.
- [7] Fernandez, J.J. (2008) High performance computing in structural determination by electron cryomicroscopy. J. Struct. Biol, 164, 1–6.
- [8] Vazquez F, Garzon E M, Martinez J A, et al. The sparse matrix vector product on GPUs[A], Proceedings of the 2009 International Conference on Computational and Mathematical Methods in Science and Engineering[C], 2009, 2: 1081-1092.
- [9] Bell N and Garland M. Efficient sparse matrix-vector multiplication on CUDA[R]. NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation, 2008.
- [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] Grimes R, Kincaid D, and Young D. ITPACK 2.0 User's Guide[R]. Technical Report CNA-150, Center for Numerical Analysis, University of Texas, 1979.
- [12] Vazquez, F., Garzon, E.M. and Fernandez, Matrix Implementation of Simultaneous Iterative Reconstruction Technique (SIRT) on GPUs, Oxford University Press on behalf of The British Computer Society.,2011
- [13]Sidden.R.L. Fast calculation of the exact radiological path for a three imensional CT array[J], Medical Physics, 1985, 12(2) 252–255.
- [14] Baskarean M M, Bordawekar R. Optimizing sparse matrix-vector multiplication on GPUs[R]. The Ohio state University, Colum-bus, USA and IBM TJ Watson Research Center Hawthorne, NY, USA, 2009.
- [15] Gilbert P. Iterative Methods for the Three-Dimensional Reconstruction of an Object from Projection [J]. Journal of Theoretical Biology, 1972, 36(1) : 105–107
- [16] Wan X, Zhang F, Liu Z: Modified simultaneous algebraic reconstruction technique and its parallelization in cryo-electron tomography. Proceeding of the International Conference on Parallel and Distributed Systems 2009, 384-390