Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 18 (2013) 1412 – 1420

Procedia Computer Science

2013 International Conference on Computational Science

# CT Image Reconstruction Based on GPUs

Liubov A. Flores*, Vicent Vidal, Patricia Mayo, Francisco Rodenas, Gumersindo Verdú

Polytechnic University of Valencia, Camino de Vera s/n, 46022 Valencia, Spain

**Abstract**

In X-ray computed tomography (CT) iterative methods are more suitable for the reconstruction of images with high contrast and precision in noisy conditions and from a small number of projections. However, in practice, these methods are not widely used due to the high computational cost of their implementation. Nowadays technology provides the possibility to reduce effectively this drawback. It is the goal of this work to develop a fast GPU-based algorithm to reconstruct high quality images from under sampled and noisy projection data.



*Keywords*: CT image reconstruction; iterative algorithm; CUDA C; CUBLAS; CUSPARSE.

## 1. Introduction

In conventional X-ray radiography the three-dimensional structure of the body is projected onto a two-dimensional image and, this loss of the depth information may become a problem when it is necessary to resolve spatially structures along the direction of x-ray propagation. The computed tomography (CT) arises as a technique to overcome this difficulty of the conventional radiography by obtaining sectional images.

Computed tomography relies on the X-ray flux measurements (projections) from different directions (angles). An X-ray beam passes through a planar slice of the body only in directions that are parallel to the plane of the slice, and the intensities of the attenuated X-rays are measured by a detector in the opposite side. Those measures are called projections. This operation must be repeated for different directions of the X-ray beam in order to get a set of data large enough to reconstruct the inner structures on the body. The problem of converting the x-ray measurements into a digital image is called reconstruction and it is an interesting and complicated topic.

* Corresponding author. Tel.: +34-645045360
*E-mail address: liuflo@posgrado.upv.es*


doi:10.1016/j.procs.2013.05.308

The most important application of CT is in diagnostic medicine, where it allows the radiologists to view internal organs with precision and safety to the patient. More recently, there have arisen numerous non-medical imaging applications.

It is well-known that the intensity of the X-ray is attenuated when it goes through a material following a decreasing exponential law. The attenuation depends on the thickness and on the linear attenuation coefficient of the material. If the material is not uniform, the X-ray passes by zones with different attenuation coefficient. Physically, the projection measured by the scanner is the line or path integral of the attenuation coefficient. Since the attenuation coefficient depends on the density and characteristics of the material, then reconstructing the attenuation coefficient means obtaining the information about the different characteristics of the objects in the X-ray line. Therefore, it is possible to obtain an image of the body's slice by reconstructing the attenuation coefficient function from the projections and then imaging the attenuation coefficient.

Assume that a X-ray beam passes through a slice of the body and that *f(x,y)* is the two dimensional distribution of the attenuation coefficient. Each single X-ray goes though the body's section producing a measure in the detector. The projection is defined as the line integral of the attenuation coefficient where the integration path is the line followed by the X-ray from the source to the detector. By moving the source and detector, it is possible to obtain a set of projections. Thus, a single *k*-th projection at angle *r* is defined by the integral along a line *l* by

$$P_{k,r} = \int_l f(x,y)dl \tag{1}$$

The reconstruction problem consists of determining the values of the function *f(x,y)* from the set of the experimental projection data. Then, imaging the attenuation coefficient function *f(x,y)*, an image of the planar section of the body is obtained. From now on, let us identify the function *f(x,y)* with the grayscale intensity values at pixel *(x,y)* of the image grid of the planar section of the body.

Mathematically, reconstruction problem corresponds to obtain a function from its line integrals for different angles. The first mathematical foundations of this problem are based on Radon work of 1917. However, since the introduction of the first clinical scanner, tremendous advances have been made in CT technology and in reconstruction techniques. Reconstruction needs to manage a very large amount of information and, then, the advances in computer science have become a key stone for the development of reconstruction techniques that could be implemented into clinical scanners successfully. There are many books about the principles of the computed tomography (see, for instance, [1-3]) and a very large number of research works devoted to particular advances or applications of CT technology (see [4] for an outlook on CT developments). In this work, we are interested in the implementation of the algorithm in a very efficient way to reduce as much as possible the computation time. Previous work done in this direction can be found in [5-6].

There exist many different reconstruction algorithms. Most algorithms can be subdivided in two categories: filtered back-projection (*FBP*) and algebraic iterative methods. The selection of one of these algorithms must represent a compromise between accuracy and computation time required. Filtered back-projection demands fewer computational resources, while iterative methods generally produce less number of errors in the reconstruction at a higher computational cost.

Filtered back-projection was the algorithm used first by almost all commercially available CT scanners [3], however, the advance in new computer architectures is doing that algebraic methods can be considered as reconstruction method in commercial scanners. Filtered back-projection methods are based on analytical algorithms which use the inverse Fourier transform and they need a set of projections equally separated (see [2] for a review on *FBP* method).

However, in CT it is common to find under sampled set of no equally spaced projections. In these cases, images reconstructed with conventional *FBP* algorithm are highly degraded due to insufficient and noisy projections. On the other hand, iterative methods do not require complete data collection and do provide the

optimal reconstruction in noisy conditions in the image. These methods allow reconstructing images with higher contrast and precision in noisy conditions from a small number of projections than the methods based on the Fourier transform [7-9].

Although widely used in nuclear medicine (gamma-camera, single photon emission computed tomography (SPECT), positron emission tomography (PET)), iterative reconstruction has not yet penetrated in CT. The main reason for this is that data sets in CT are much larger than in nuclear medicine and iterative reconstruction then becomes computationally very intensive. Acceleration of iterative reconstruction is an active area of research. Stone *et al*. [5] describe the accelerated reconstruction algorithm on graphical processing units (GPUs) for advanced magnetic resonance imaging (MRI). They reconstruct images of $128^3$ voxels in over one minute. Johnson and Sofer [10] propose a parallel computational method for emission tomography applications that is capable of exploiting the sparsity and symmetries of the model and demonstrate that such a parallelization scheme is applicable to the majority of iterative reconstruction algorithms. The time needed for the reconstruction of thick-slices images (128x128x23 in voxels) is over 3 minutes. Pratx *et al* [11] show results of iterative reconstruction using GPU in PET. The required time on a single GPU to reconstruct an image of 1603 voxels is 8.8 second. Multi GPU implementation of tomographic reconstruction accelerates reconstruction of images 350x350x9 up to 67 seconds on a single GPU and 32 seconds on four GPUs [6].

It seems that the resolution of an image to be reconstructed remains to be a problem. In our previous work we have reported results on using Extensive Toolkit for Scientific computation (PETSc) and binary format of input data to facilitate the programming task and accelerate the whole process of reconstruction [12-13]. In this research, our aim is to take advantage of the massive computing power of GPU in order to reconstruct CT images with higher resolution without losing quality. We present a description and validation of our algorithm

The rest of the paper is organized as follows: mathematical aspects of the problem are described in section 2 and the reconstruction algorithm in section 3, GPU implementation of this algorithm is presented in section 4. The results obtained in the experiment are shown in section 5 and, finally, we summarize the conclusions.

## 2. Algebraic reconstruction method

In this work we use an algebraic method to reconstruct the function *f(x,y)* in equation (1) from the set of projections. This approach for tomography imaging consists of assuming that the cross section is a discrete squared grid and that in each cell (image pixel) the function *f(x,y)* is constant. Then, in this approach reconstruction means to solve a set of algebraic equations for the unknowns (*f(x,y)* value at each pixel) in terms of the measured projection data.

Let $x_{ij}$ denotes the constant value of *f(x,y)* in the *ij*th pixel with *i,j=1,2...n*. Fundamentally, the algebraic methods of image reconstruction from projections are schemes for solving a linear equation system:

$$Ax = P, \tag{2}$$

where the system matrix $\boldsymbol{A}$ simulates computer tomography functioning and its elements ($W_{ij}$) depend on the projection number and the angle and may not be square, $\boldsymbol{x}$ is a column matrix whose values represent intensities of the image , and the column matrix $\boldsymbol{P}$ represents projections collected by a scanner.

Assume that, for a given angle, the number of projections ranges from 1 to *m*. If there are *k* different angles, then in (2) $\boldsymbol{P}$ is a column matrix with *mxk* elements, $\boldsymbol{x}$ is a column matrix with $n^2$ elements and $\boldsymbol{A}$ is a $mkxn^2$ rectangular matrix as it follows in (3). In (3), *Wij(rq)* denotes the entries of the matrix $\boldsymbol{A}$ and sub-indices *ij* in *Wij(rq)* refer to the pixel location and indices *r,q* refer to projection and angle numbers respectively. *Wij(rq)* is the weighting factor that represents the contribution of the *ij*th pixel to the *rq*th X-ray line integral (1).

Many properties of the reconstructed image depend on the approximations when calculating the system matrix. In this work we use Siddon's algorithm to calculate elements of the matrix in a rectangular grid [14].

$$A=\begin{bmatrix} W_{11}(11) & W_{12}(11) & \dots & W_{nn}(11) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ W_{11}(m1) & W_{12}(m1) & \dots & W_{nn}(m1) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ W_{11}(mk) & W_{12}(mk) & \dots & W_{nn}(mk) \end{bmatrix}, \quad P=\left[p_{11}\dots p_{m1}\dots p_{mk}\right]^T, \quad x=\left[x_{11}\dots x_{12}\dots x_{nn}\right]^T. \qquad (3)$$

According to Siddon's method, *Wij(rq)* is equal to the length of the segment of the *rq*th X-ray going through the *ij*th image pixel. Note that most of the *Wij(rq)* elements are zero since only a small number of cells contribute to any individual X-ray. It has been found that Siddon's algorithm gives a good approximation of the system matrix [15].

## 3. Reconstruction algorithm

Many different algorithms have been proposed in the literature to solve the reconstruction problem (2) (see [1] for a review). The properties of the system matrix ***A*** are very important in order to propose an algorithm to solve the system (2). Main characteristics of the matrices used in the experiment are summarized in Table 1. Figure 1 shows a sparsity pattern of such matrices (it is shown a part of the matrix with 3500 rows and 8000 columns). In practice, ***A*** is a rectangular no symmetrical sparse matrix and therefore it is recommendable to store only nonzero elements. The appropriate storage format for such matrices is Compact Sparse Row (CSR) or Compact Sparse Column (CSC) format. The system (2) may be over determined or undetermined. Over determined systems contain more information on the image and, consequently, the reconstructed image is less noisy. The dimensions of ***A*** grow proportionally to the resolution of the image to be reconstructed and the number of projections, increasing therefore the computational cost. In the experiment, the input matrix ***A*** and the right hand side vector ***P*** have been generated previously, they can be stored in two formats: as a plain text (ASCII format) or in a binary format. We use the input data in binary format, which allows reducing the memory storage and the loading time of the input data (up to 10 times).

Table 1. The main characteristics of the system matrix

| Matrix Size (pixels) | Generation Time (sec) | Matrix Size (MB) | |
|---|---|---|---|
| | | *ASCII format* | *Binary format* |
| (256x100) x (256x256) | 11.3 | 236 | 91 |
| (256x200) x (256x256) | 22.4 | 475 | 181 |
| (256x400) x (256x256) | 45.4 | 954 | 361 |
| (512x100) x (512x512) | 72.5 | 973 | 361 |
| (512x200) x (512x512) | 203.2 | 2047 | 721 |
| (512x400) x (512x512) | 446.4 | 2148 | 755 |

We implemented the Least Square QR method (LSQR) [16] to solve the system (2) by minimizing $\min\|Ax-P\|_2$. The matrix ***A*** is normally large and sparse and is used only to compute products of the form ***Av*** and $\boldsymbol{A^T u}$ for various vectors ***v*** and ***u***. The input data is stored in binary format. Figure 2 illustrates the following main steps of the reconstruction process:

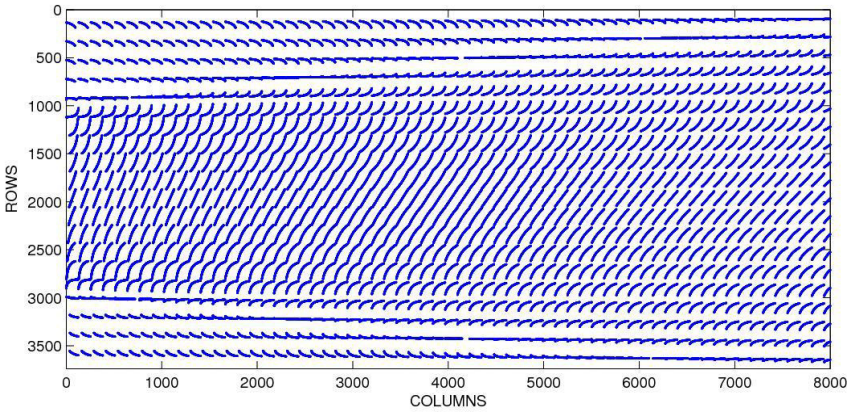


Fig. 1. Sparsity pattern of the system matrix. It is shown a part of the matrix data structure that corresponds to 3500 rows and 8000 columns. Matrix of the size (512x400) x (512x512) has 125986768 nonzero elements.

- CT projections are collected by a scanner
- The system matrix that simulates the scanning process is generated previously by Siddon algorithm
- In binary format these data are used by LSQR solver to find the solution of the system (2) that represents the reconstructed image.
- LSQR solver is implemented in CUDA parallel programming model.

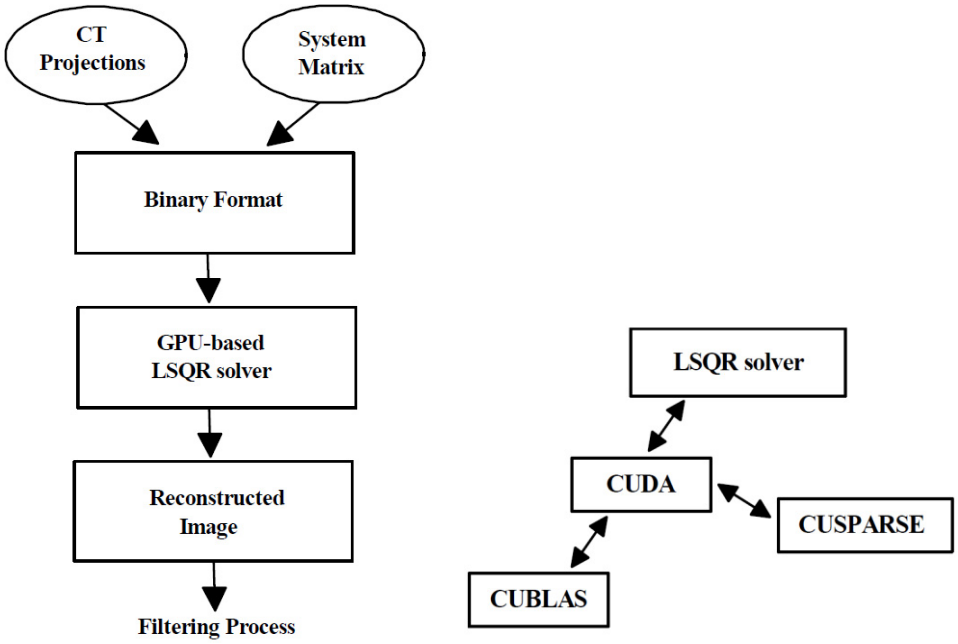


Fig. 2. a) LSQR solver uses input data in binary format to reconstruct an image; b) Libraries used for the implementation of the algorithm and their relationship.

In our previous work, we have analyzed the efficiency of the LSQR solver in parallel image reconstruction on CPU [13]. A speed up of 1.8 has been achieved to reconstruct images of 512x512 pixels. In this paper, we attempt to develop an algorithm suitable for GPU parallelization in order to take advantage of the massive computing power of GPUs.

## 4. GPU implementation

Computer graphic cards, such as the NVIDIA GeForce series and the GTX series, are conventionally used for display purpose on desktop computers. Special GPUs card dedicated for scientific computing, like the NVIDIA Tesla M2050 card is used in this paper to carry out the experiment. Such a GPU card has a total number of 448 cuda cores with 3GB ECC memory, shared by all processor cores. Utilizing such a GPU card with tremendous parallel computing ability can considerably elevate the computation efficiency of our algorithm.

NVIDIA also introduced CUDA$^{T}$ [17], a general purpose parallel computing architecture – with a new parallel programming model and instruction set architecture – that leverages the parallel compute engine in NVIDIA GPUs to solve many complex computational problems in a more efficient way than on a CPU. CUDA comes with a software environment that allows developers to use C or C++ as high-level programming languages and overcome the challenge to develop application software that transparently scales its parallelism to leverage the increasing number of processor cores.

We also use CUBLAS [18] and CUSPARSE [19] libraries that allow the user to access the computational resources of NVIDIA Graphical Processing Unit (GPU). The CUBLAS library is an implementation of BLAS (Basic Linear Algebra Subprograms) on top of the NVIDIA®CUDA™ runtime. To use the CUBLAS library, the application must allocate the required matrices and vectors in the GPU memory space, fill them with data, call the sequence of desired CUBLAS functions, and then upload the results from the GPU memory space back to the host. The CUBLAS library also provides helper functions for writing and retrieving data from the GPU.

The NVIDIA® CUDA™ CUSPARSE library contains a set of basic linear algebra subroutines used for handling sparse matrices and is designed to be called from C or C++. These subroutines include operations between vector and matrices in sparse and dense format, as well as conversion routines that allow conversion between different matrix formats. Fig. 3 shows the libraries used for the implementation of the algorithm and their relationship.

The following piece of the code represents the usage of the library functions used to compute norm of a vector:

```
01:  cublasCreate ( &handle_b );
02:  cublasSetVector ( nrow, sizeof(float), h_U, 1, d_U, 1 );
03:  cublasSnrm2 ( handle_b, nrow, d_U, 1, &beta );
04:  cublasScal ( handle_b, nrow, &beta1, d_U, 1 );
05:  cublasGetVector ( nrow, sizeof(float), d_U, 1, h_U, 1 );
```

and matrix - vector product:

```
06:  cusparseCreate (&handle_s);
07:  cusparseCreateMatDescr(&descra);
08:  cusparseSetMatType(descra, CUSPARSE_MATRIX_TYPE_GENERAL);
09:  cusparseSetMatIndexBase(descra,CUSPARSE_INDEX_BASE_ZERO);
10:  cusparseScsrmv ( handle_s, CUSPARSE_OPERATION_NON_TRANSPOSE, ncol, nrow,
       1.0, descra, csc_values,  cscColPtr, cscRowInd, d_U, 0.0, d_V );
11:  cublasGetVector ( ncol, sizeof(float), d_V, 1, h_V, 1 );
```

CUBLAS and CUSPARSE are written using the CUDA parallel programming model and take advantage of the

computational resources of the NVIDIA graphics processor (GPU).

## 5. Experimental results

For experimental purposes we used real projections and reference images acquired from the Hospital Clinico Universitario in Valencia. We worked with fan-beam projections collected by the scanner with 512 sensors in the range 0 - 180 with 0.9 degree spacing. To be able to reconstruct the image with the iterative method we complete the given set up to 360 degrees using the symmetry of the system matrix. We wanted to analyze the capacity of iterative algorithms in parallel reconstruction of images from less number of projections. With this purpose, from the initial set, three sets of equally spaced (with the angle steps 0.9, 1.8, and 3.6 degrees) projections have been derived.

The results have been measured on a one GPU card of the cluster system Euler that belongs to the Alicante University in Spain. The GPU computing node consists of 2 x CPU Intel Xeon X5660, each with 6 cores of 2,80 GHz and 3 x GPU NVIDIA TESLA M2050 with 448 cores and 3GB memory each of them. In Euler, it is used Grid Engine function, general purpose Distributed Resource Management (DRM) tool. The scheduler component in Grid Engine supports a wide range of different compute scenarios. Jobs are queued and executed remotely according to defined policies.

For the images of 256x256 and 512x512 pixels the solving time of system (2) on CPU with 1 core and on one GPU card is given in Table 2. The execution time on CPU has been measured with gettimeofday() function . On the GPU card, we used cudaEventRecord() function that measures only the execution time of operations on a device not taking into account time spent in queues. The standard deviation of the results after running the application ten times is 2.9e-004. In the system matrix, the number of rows is obtained by multiplying the number of used sensors and angles and corresponds to the number of the projections used to reconstruct the image; the number of columns corresponds to the size of the reconstructed image (256x256 and 512x512 pixels).

Table 2. The reconstruction time of images on CPU and using one GPU card on Euler cluster

| System Matrix (rows x columns) | One core CPU (seconds) | One GPU card (seconds) |
|---|---|---|
| M1 = (256x100) x (256x256) | 2.7 | 0.1569 |
| M2 = (256x200) x (256x256) | 5.3 | 0.3056 |
| M3 = (256x400) x (256x256) | 10.5 | 0.6127 |
| M4 = (512x100) x (512x512) | 12.3 | 0.6584 |
| M5 = (512x200) x (512x512) | 24.4 | 1.2741 |

The results show the efficiency of the algorithm based on a GPU parallel computing ability. SpeedUp of 19.2 has been achieved to reconstruct images of 512x512 pixels. Also, it can be seen that usage of GPUs becomes more efficient for large scale problems.

Finally, Figure 3 shows the images reconstructed in parallel from different number of equally spaced projections. It is needed to be mentioned that usually post processing procedure (as filtering) is applied to the reconstructed image in order to improve the quality. In this work we present the images right after the reconstruction stage without any filtering.

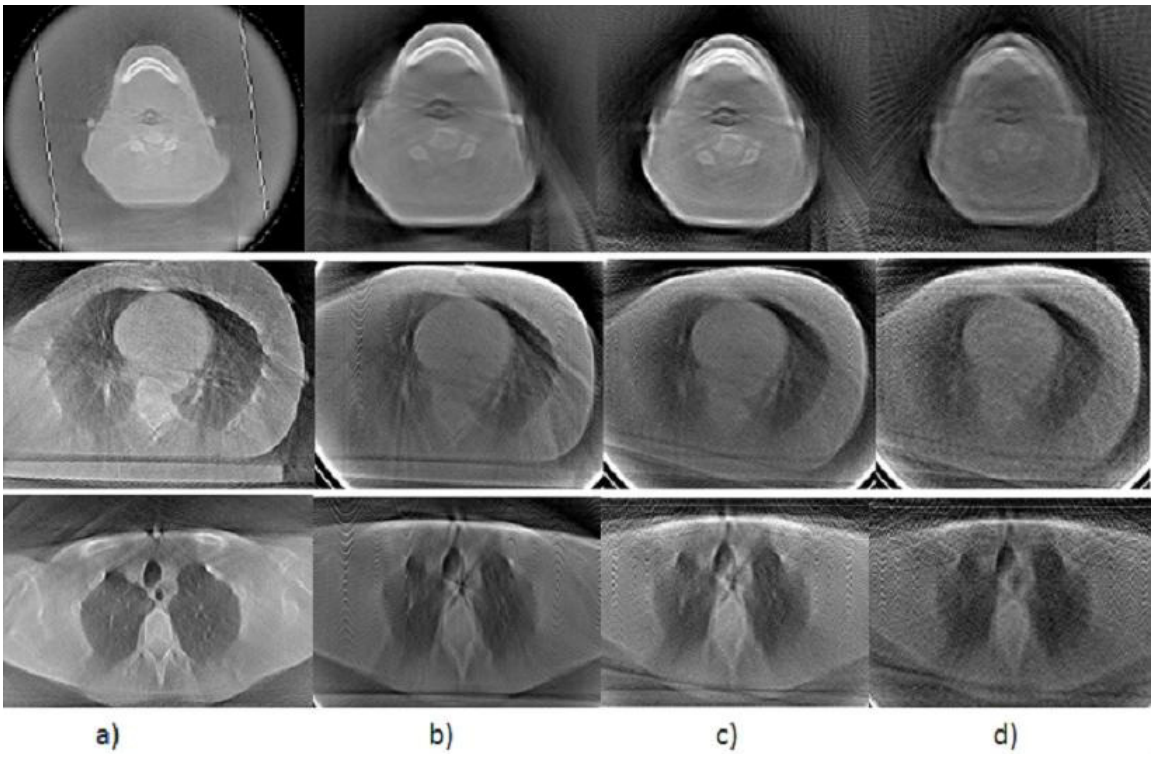


Fig. 3. Reconstructed images (512x512 pixels): a) reference images; b), c), d) iterative reconstruction from 400, 200 and 100 angles at the iteration 12 when the given tolerance is achieved

## 6. Conclusions

The GPU-based iterative algorithm of image reconstruction presented in this paper shows that the algebraic iterative methods are capable to reconstruct images at low computational cost.

CUDA parallel programming model with CUBLAS and CUSPARSE libraries allow overcoming the challenge to solve complex computational problems and take advantage of the computational resources of the NVIDIA graphics processor (GPU). We expect more significant results in undergoing work of 3D image reconstruction when a huge amount of computing is involved.

## Acknowledgements

This research was supported by ANITRAN Project PROMETEO/2010/039.

We wish to thank Dr. Sergio Díez, Head of the Radiology and Radiophysics Protection Service of the hospital Clinico Universitario, for the collaboration in carrying out this work.

We are also grateful to the Alicante University for allowing testing our algorithms on Euler cluster system.

## References

[1] Herman, G. T. *Fundamentals of computerized tomography: Image reconstruction from projection*. 2nd ed. Springer; 2009.
[2] A. C. Kak and M. Slaney. *Principles of Computerized Tomographic Imaging*. IEEE Press, 1988. Republished by SIAM, electronic copy, 1999.
[3] Jiang Hsieh. *Computed Tomography: Principles, Design, Artifacts, and Recent Advances*. SPIE Press; 2003.
[4] Wang G.,Yu H. and De Man B. An outlook on X-ray CT research and development. *Medical Physics* 2008; 35(3): p. 1051-1064.
[5] Stone S. S., Haldar J. P., Tsao S.C., Hwu W.-m W., Sutton B. P., Liang Z. P., 2008. Accelerating advanced MRI reconstructions on GPUs. Journal of Parallel and Distributed Computing, vol. 68, issue 10, 1307-1318.
[6] Jang B, Kaeli D., Do S., Pien H., 2009. Multi GPU implementation of iterative tomographic reconstruction algorithms. Biomedical Imaging: From Nano to Macro, pp. 185-188.
[7] Crawford B. M. and Herman G. T. Low-dose, large-angled cone-beam helical CT data reconstruction using algebraic reconstruction techniques. *Image and Vision Comp.* 2007; 25: p. 78-94.
[8] Nuyts J, De Man B., Dupont, M. P., Defrise, Suetens P., and Mortelmans L. Iterative reconstruction for helical CT: A simulation study. *Phys. Med. Biol 1998;* 43: p. 729-737.
[9] Wells R. G., King M. A., Simkin P. H., Judy P. F., Brill A. B, Licho R., Pretorius P. H., Schneider P. B., and Seldin D. W. Comparing Filtered back projection and ordered-subsets expectation maximization for small-lesion detection and localization in 67Ga SPECT. *J. Nucl. Med.* 2000; 41: p. 1391-1399.
[10] Johnson C.A., Sofer. A., 1999. A data-parallel algorithm for iterative tomographic image reconstruction. Frontiers of Massively Parallel Computation, pp. 126-137.
[11] Pratx G., Chinn G., Olcott P.D., Levin C. S., 2009. Fast, Accurate and Shift-Varying Line Projections for Iterative Reconstruction Using the GPU. IEEE Transactions on Medical Imaging, 28(3), pp. 435-445.
[12] Flores L., Vidal V., Mayo P., Rodenas F., Verdú G. Iterative reconstruction of CT images with PETSc. *BMEI 2011; vol.* 1 p. 343-346.
[13] Flores L.,Vidal V., Mayo P., Rodenas F., Verdú G. Fast parallel algorithm for CT image reconstruction. Proceedings of the IEEE EMBC 2012; p. 4374-4377. ISBN: 978-1-4577-1787.
[14] Siddon R.. Fast calculation of the exact radiological path length for a three dimensional CT array. *Med. Phys*. 1985; 12: p. 252-255.
[15] Cibeles Mora Mora M.T., 2008. Tesis PhD. Métodos de Reconstrucción Volumétrica Algebraica de Imágenes Tomográficas.
[16] Paige C. C. and Saunders M. A., 1982. LSQR: An Algorithm for Sparse Linear Equations and Sparse Least Squares, *ACM Trans. Math. Sof*., **8**, 1, p. 43-71.
[17] Cuda C Programming Guide. Downloaded in Oct. 2012 from URL http://docs.nvidia.com/cuda/index.html.
[18] CUBLAS Library. Downloaded in Oct. 2012 from URL http://docs.nvidia.com/cuda/index.html.
[19] CUSPARSE Library. Downloaded in Oct. 2012 from URL http://docs.nvidia.com/cuda/index.html.