Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 00 (2013) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

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.

© 2013 The Authors. Published by Elsevier B.V.
Selection and/or peer-review under responsibility of the organizers of the 2013 International Conference on Computational Science

Keywords: CT image reconstruction; GPU based algoruthm; CUDA C.

1. Introduction

In medicine, the diagnosis based on computed tomography (CT) is fundamental for the detection of abnormal tissues by different attenuation on X-ray energy, which frequently is not clearly distinguished for radiologists. In CT imaging, a set of projections taken with a scanner is used to reconstruct the internal structure of an object. The intensity of a beam of X-ray that passes through some object is observed to decrease. By moving the source and detector, it is possible to obtain a set of projections. A single *k*-th projection at angle *r* can be defined as an integral of image intensities *f(x,y)* along a line *l* and is given by the formula:

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

The reconstruction problem consists of determining the values of the function *f(x,y)* from the set of the experimental projection data. Presently, the reconstruction process in clinical scanners is based on analytical

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

algorithms which use the inverse Fourier transform. Filtered Back Projection algorithm (*FBP*) is one of the widely used algorithms and is well described in literature [1]. 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, algebraic methods do not require complete data collection and do provide the optimal reconstruction in noisy conditions in the image [2]. 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 [3-5].

Nevertheless, the major drawback of the algebraic methods is given by their high computational cost. In our previous work we have reported some 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 [6-7]. In this research, our aim is to take advantage of the massive computing power of graphics processing unit (GPU) to improve the efficiency of the reconstruction process. In this paper, we will present a description and validation of our algorithm. The rest of the paper is organized as follows: in part 2 we describe mathematical aspects of the problem, the reconstruction algorithm, and the GPU implementation of this algorithm. In part 3 we present the results obtained in the experiment and, finally, we summarize the conclusions in part 4.

2. Methodology

2.1. Mathematical aspects

Fundamentally, the algebraic methods of image reconstruction from projections are schemes for solving a linear 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.

For a given angle, we assume that 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 *mk*xn^2 rectangular matrix:

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

Many properties of the reconstructed image depend on the approximations when calculating the system matrix. In this work we use Siddon algorithm to calculate elements of the matrix in a rectangular grid [8]. It has been found that Siddon algorithm gives a good approximation of the system matrix [9]. The main characteristics of the matrices used in the experiment are summarized in Table 1 and Figure 1 shows the structure of such matrices.

In practice, $\boldsymbol{A}$ 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) o 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

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

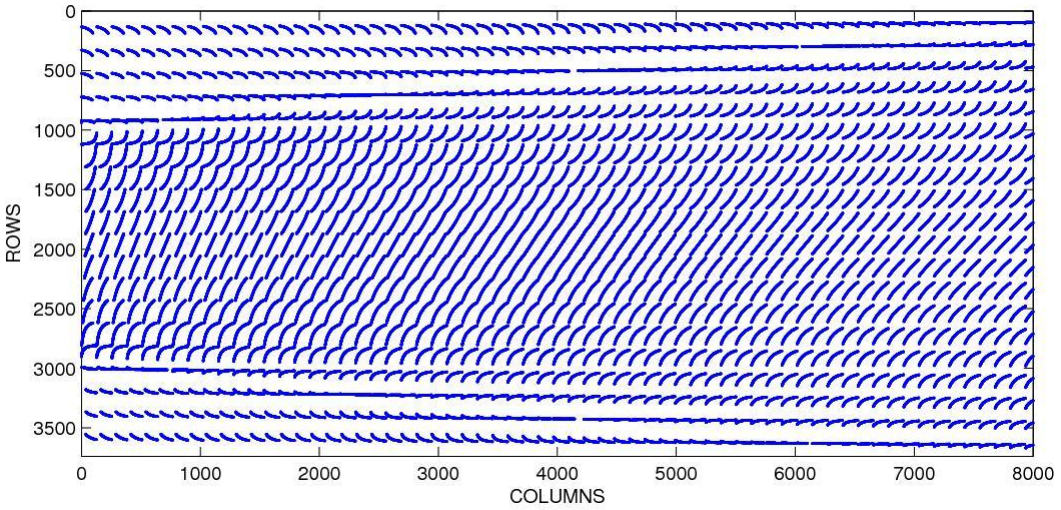

Fig. 1. The system matrix data structure

dimensions of $\boldsymbol{A}$ 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 $\boldsymbol{A}$ and the right hand side vector $\boldsymbol{P}$ 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 computing time.

2.2. Algorithm

We implemented the Least Square QR method (LSQR) [10] to solve the system (2) by minimizing $\min\|Ax-P\|_2$. The matrix $\boldsymbol{A}$ is normally large and sparse and is used only to compute products of the form $\boldsymbol{Av}$ and $\boldsymbol{A^Tu}$ for various vectors $\boldsymbol{v}$ and $\boldsymbol{u}$. The input data is stored in binary format. Figure 2 illustrates the following main steps of the reconstruction process:

- 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.

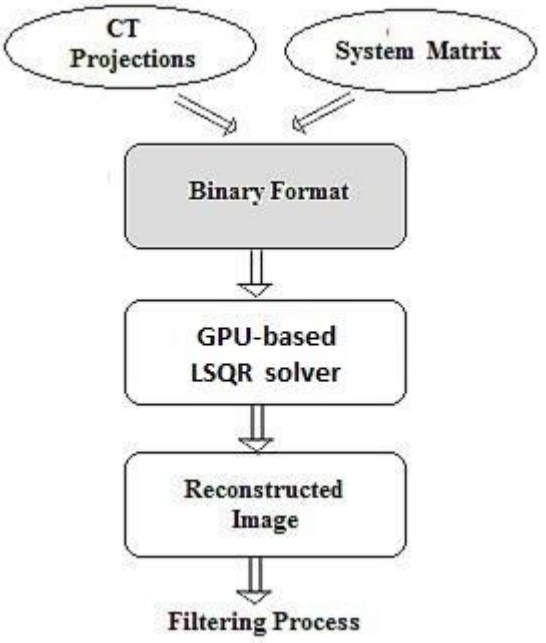

Fig. 2. LSQR solver uses input data in binary format to reconstruct an image

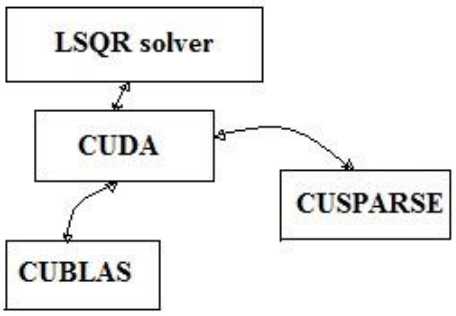

Fig. 3. Libraries used for the implementation of the algorithm

The efficiency of the LSQR solver in parallel image reconstruction on CPU we have analyzed in our previous work [6]. The 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.

2.3. 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™, 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 and CUSPARSE 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).

3. Results And Discussions

For experimental purposes we used real projections and original 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 GPU node 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.

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

System Matrix (rows x columns)	CPU (one core) (seconds)	GPU (seconds)
M1 = (256x100) x (256x256)	2.7	4.4
M2 = (256x200) x (256x256)	5.3	4.6
M3 = (256x400) x (256x256)	10.5	4.7
M4 = (512x100) x (512x512)	12.3	5.1
M5 = (512x200) x (512x512)	24.4	5.3

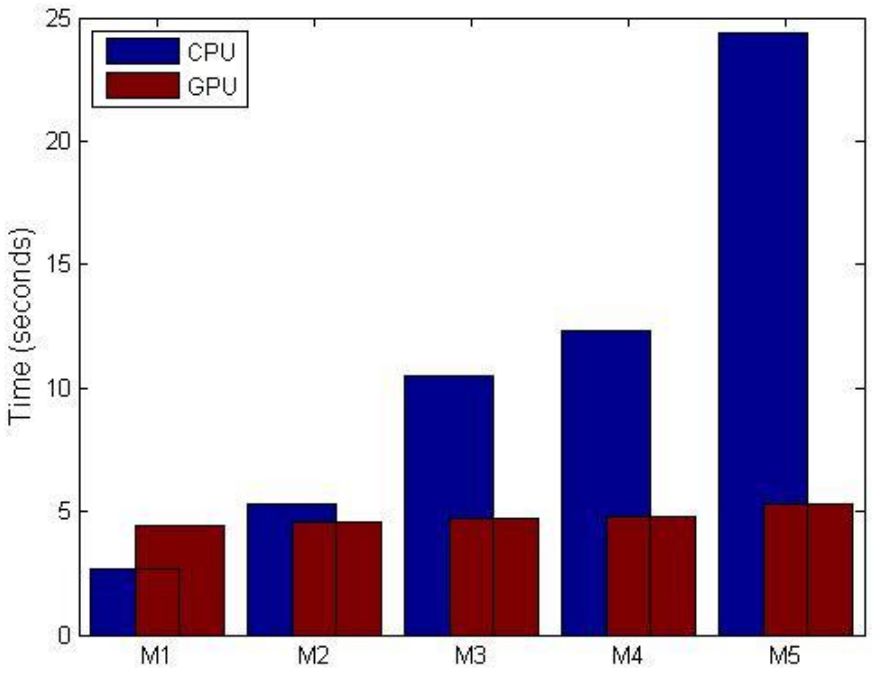

Fig. 4. Reconstruction time on CPU (one core) and GPU from different number of projections; the matrices corresponds to: M1=[(256x100)x(256x256)], M2=[(256x200)x(256x256)], M3=[(256x400)x(256x256)], M4=[(512x100)x(512x512)], M5=[(512x200)x(512x512)] where the rows represent the number of the projections and the columns - the size (256x256 or 512x512 pixels) of the reconstructed image

For the images of 256x256 and 512x512 pixels the solving time of the system (2) on CPU with 1 core and GPU is given in Table 2 and shown in Figure 4. 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).

The results show the efficiency of the algorithm based on a GPU parallel computing ability. It can be seen that the usage of GPUs becomes more efficient for large scale problems.

Finally, Figure 5 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.

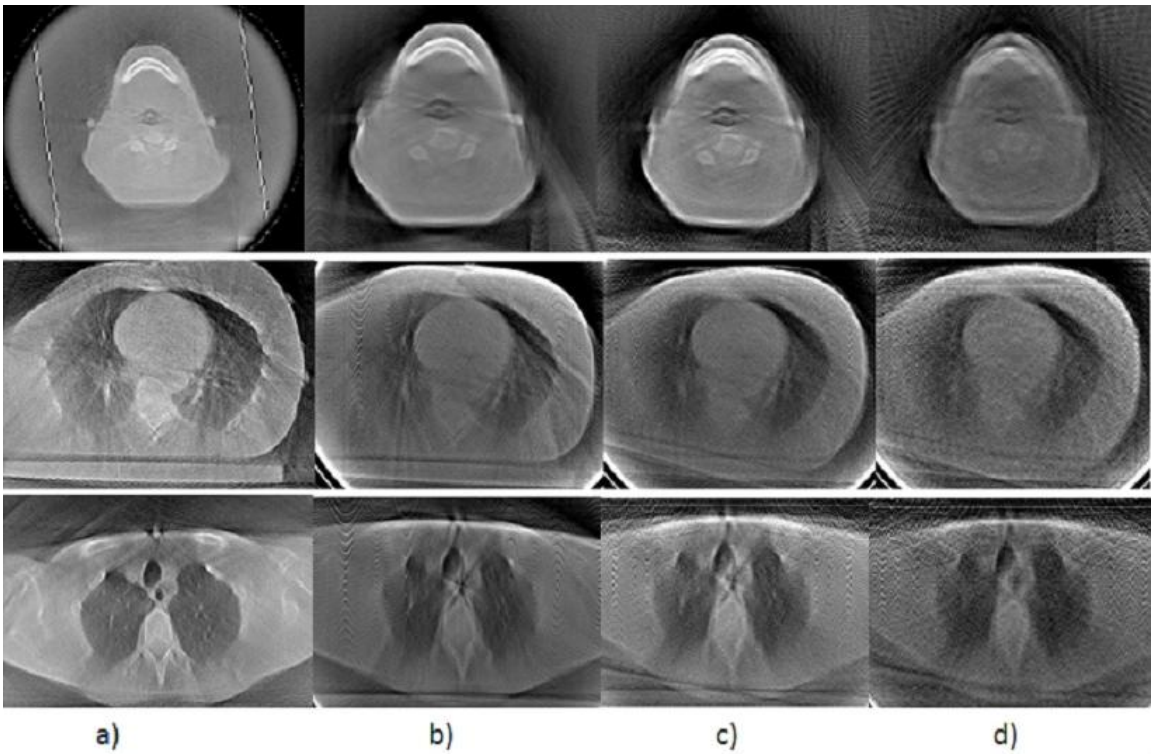

Fig. 5. Reconstructed images (512x512 pixels): a) original images; b), c), d) iterative reconstruction from 400, 200 and 100 angles at the iteration 12 when the given tolerance is achieved

4. Conclusions

The GPU-based iterative algorithm of image reconstruction presented in this paper shows that the algebraic methods are capable to reconstruct images with 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

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] Gonzales R. C. and Woods R. E. *Digital Image Processing*. 3rd ed. Prentice Hall; 2008.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] Flores L., Vidal V., Mayo P., Rodenas F., Verdú G., 2011. Iterative reconstruction of CT images with PETSc. *BMEI; vol.* 1 p. 343-346.

[7] Flores L.,Vidal V., Mayo P., Rodenas F., Verdú G., 2012.Fast parallel algorithm for CT image reconstruction. Proceedings of the IEEE EMBC ; p. 4374-4377. ISBN: 978-1-4577-1787.

[8] Siddon R.., 1985. Fast calculation of the exact radiological path length for a three dimensional CT array. Med. Phys. 12: p. 252-255.

[9] Cibeles Mora Mora M.T., 2008. Tesis PhD. Métodos de Reconstrucción Volumétrica Algebraica de Imágenes Tomográficas.

[10] 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.

http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf Last access 11.2012

http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUBLAS_Library.pdf. Last access 10.2012

http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUSPARSE_Library.pdf. Last access 10.2012