Parallel CT image reconstruction based on GPUs[☆]

Liubov A. Flores [a,*], Vicent Vidal [a], Patricia Mayo [b], Francisco Rodenas [c], Gumersindo Verdú [d]

[a] *Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, España*
[b] *Servicios Tecnológicos, Grupo Dominguis, Sorolla Center, local 10 Avda. de las Cortes Valencianas, 46015 Valencia, España*
[c] *Departamento de Matemática Aplicada, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, España*
[d] *Departamento de Ingeniería Química y Nuclear, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, España*

H I G H L I G H T S

- We developed GPU-based iterative algorithm to reconstruct images.
- Iterative algorithms are capable to reconstruct images from under sampled set of projections.
- The computer cost of the implementation of the developed algorithm is low.
- The efficiency of the algorithm increases for the large scale problems.

A B S T R A C T

In X-ray computed tomography (CT) iterative methods are more suitable for the reconstruction of images with high contrast and precision in noisy conditions 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
GPU-based algorithm
CUDA C

1. Introduction

In medicine, the diagnosis based on computed tomography (CT) is fundamental for 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 X-ray beam that passes through some object is observed to decrease. By moving source and detector, it is possible to obtain a set of projections. A single *k*-th projection at the angle *r* can be defined as an integral of image intensities $f(x,y)$ along the line *l* and is given by the formula:

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

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 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 (i.e. in Gonzales and Woods, 2008). However, in CT, it is common to find under sampled set of no equally spaced projections. In these cases, images reconstructed with the 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 (Wang et al., 2008). 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 (Crawford and Herman, 2007; Nuyts et al., 1998; Wells et al., 2000).

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. (2008) describe the accelerated reconstruction algorithm on

* Corresponding author. Tel.: +34 64 504 5360.
E-mail addresses: liuflo@posgrado.upv.es, flores.liubov@gmail.com (L.A. Flores), vvidal@dsic.upv.es (V. Vidal), p.mayo@titaniast.com (P. Mayo), frodenas@mat.upv.es (F. Rodenas), gverdu@iqn.upv.es (G. Verdú).
☆Research supported by ANITRAN Project PROMETEO/2010/039.

graphical processing units (GPUs) for advanced magnetic resonance imaging (MRI). They reconstruct images of 128^3 voxels in over 1 min. Johnson and Sofer (1999) 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 ($128 \times 128 \times 23$ in voxels) is over 3 min. Pratx et al. (2009) show results of iterative reconstruction using GPU in PET. The required time on a single GPU to reconstruct an image of 160^3 voxels is 8.8 s. Multi GPU implementation of tomographic reconstruction (Jang et al., 2009) accelerates reconstruction of images $350 \times 350 \times 9$ up to 67 s on a single GPU and 32 s on four GPUs.

It seems that the resolution of an image to be reconstructed remains to be a problem. 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 (Flores et al., 2011, 2012). In this research, our aim is to take advantage of the massive computing power of GPU in order to reconstruct the CT images with higher resolution without losing quality. We present a description and validation of our algorithm.

2. Mathematical aspects

Fundamentally, the iterative 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 depend on the projection number, the angle at which the projections are taken and may not be square, $\boldsymbol{x}$ is a column matrix whose values represent intensities of the image, the column matrix $\boldsymbol{p}$ corresponds to the 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 Eq. (2) $\boldsymbol{p}$ is a column matrix with mxk elements, $\boldsymbol{x}$ is a column matrix with n^2 elements and A is a $mkxn^2$ rectangular matrix:

$$A = \begin{bmatrix} W_{11}(11) & W_{12}(11) & \cdots & W_{nn}(11) \\ \hdashline W_{11}(m1) & W_{12}(m1) & \cdots & W_{nn}(m1) \\ \hdashline W_{11}(mk) & W_{12}(mk) & \cdots & W_{nn}(mk) \end{bmatrix}, \tag{3}$$

$$p = [p_{11} \ldots p_{m1} \ldots p_{mk}]^T, \quad x = [x_{11} \ldots x_{12} \ldots x_{nn}]^T.$$

Many properties of a 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 (Siddon, 1985). It has been found that Siddon algorithm gives a good approximation of the system matrix (Mora, 2008). The main characteristics of the matrices used in the experiment are summarized in Table 1.

In practice, $\boldsymbol{A}$ is a rectangular nonsymmetrical sparse matrix and therefore it is recommendable to store only nonzero elements. 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 $\boldsymbol{A}$ grow proportionally to the resolution of the image to be reconstructed and the number of projections, increasing therefore the computational cost. The input matrix $\boldsymbol{A}$ and the right hand side vector $\boldsymbol{p}$ are generated previously and can be stored in two formats: as a plain text (ASCII format) or in a binary format. In our experiment the input data is used in binary format, which allows reducing the memory storage (up to three times) and loading time of the input data.

2.1. Algorithm

We implemented the Least Square QR method (LSQR) to solve the system (2) by minimizing: $\min\|Ax-P\|_2$ (Paige and Saunders, 1982). 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 $\boldsymbol{u}$. LSQR is an iterative algorithm. The process exits if the following stopping criterion is met:

$$\frac{\|Ax-p\|}{\|p\|} \leq rtol. \tag{4}$$

In Eq. (4), $\|Ax-p\|$, $\|p\|$ are norms of residual and right-hand side vector respectively, $rtol$ is a given tolerance. In our experiments $rtol=0.1$. The input data is stored in binary format. Fig. 1 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.
-

Table 1
The main characteristics of the system matrix.

Matrix size (pixels)	Generation time (sec)	#Nonzero elements	Matrix size in binary format (MB)	Loading time (s) of the matrix	
				ASCII	*Binary*
$(256 \times 100) \times (256 \times 256)$	11.3	7,872,591	91	9.41	2.3
$(256 \times 200) \times (256 \times 256)$	22.4	15,745,104	181	28.3	4.2
$(256 \times 400) \times (256 \times 256)$	45.4	31,490,052	361	42.7	5.4
$(512 \times 100) \times (512 \times 512)$	72.5	31,496,952	361	76.0	7.8
$(512 \times 200) \times (512 \times 512)$	203.2	62,993,644	721	171.1	15.4
$(512 \times 400) \times (512 \times 512)$	446.4	125,986,768	1500	297.1	28.3

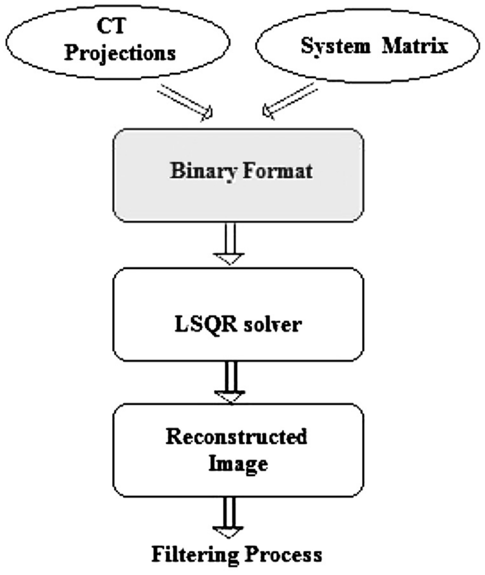

Fig. 1. LSQR solver uses input data in binary format to reconstruct the image.

In binary format these data are used by LSQR solver to find the solution of the system (2) that represents the reconstructed image.

In a previous work, we have analyzed the efficiency of the LSQR solver in parallel image reconstruction on CPU (Flores et al., 2012). A speed up of 1.8 has been achieved to reconstruct images of 512×512 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.

2.2. 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^{TM}, 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^{TM} 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^{TM} 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.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 reference images acquired from the Hospital Clinico Universitario in Valencia. The reference images have been reconstructed with Filtered back-projection method (FBP) from a complete set of projections collected by the scanner with 512 sensors in the range 0–180 with 0.9° 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. So, we consider a complete set as a set of projections that corresponds to 512 sensors and 400 angles. With the purpose to analyze the capacity of iterative algorithms in parallel reconstruction of images from less number of projections from the initial set the following subsets of equally spaced (with the angle steps 0.9°, 1.8°, and 3.6°) projections have been derived: set1 = 256×100, set2 = 256×200, set3 = 256×400, set4 = 512×100, set5 = 512×200.

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 2xCPU Intel Xeon X5660, each with 6 cores of 2,80 GHz and 3xGPU 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 computational scenarios. Grid Engine is a facility for executing Unix-like batch jobs (shell scripts or binaries) on a pool of cooperating CPUs. Jobs are queued and executed remotely according to defined policies.

For the images of 256×256 and 512×512 pixels reconstructed from different number of projections, the solving time of the system (2) on one CPU and GPU is given in Table 2 and shown in Fig. 2. 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 (256×256 and 512×512 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, Fig. 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.

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

System matrix (rows x columns)	**CPU (s)**	**GPU (s)**
$(256 \times 100) \times (256 \times 256)$	2.7	4.4
$(256 \times 200) \times (256 \times 256)$	5.3	4.6
$(256 \times 400) \times (256 \times 256)$	10.5	4.7
$(512 \times 100) \times (512 \times 512)$	12.3	5.1
$(512 \times 200) \times (512 \times 512)$	24.4	5.3

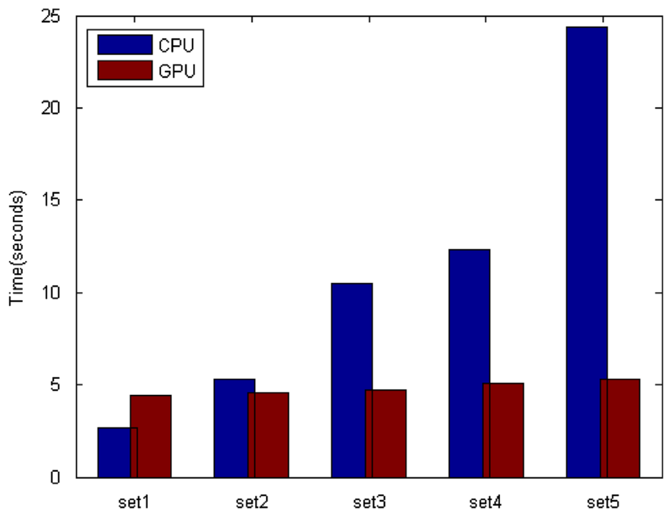

Fig. 2. Reconstruction time on CPU and GPU from different number of projections: set1 = 256×100, set2 = 256×200, set3 = 256×400, set4 = 512×100, set5 = 512×200.

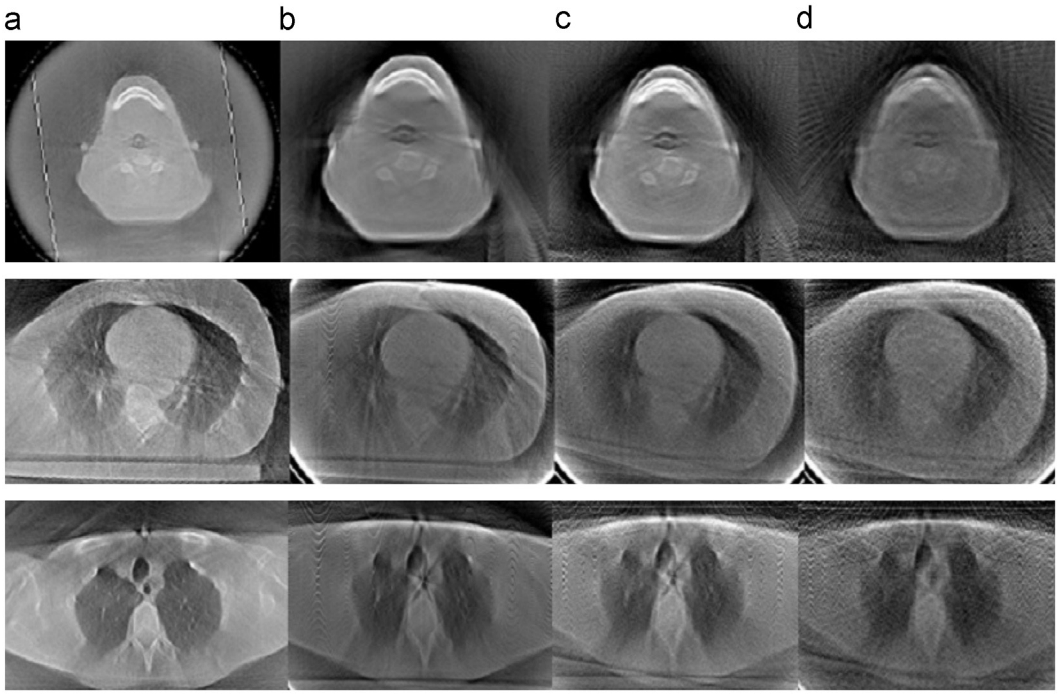

Fig. 3. Reconstructed images (512 × 512 pixels): (a) FBP reconstruction; (b), (c), and (d) iterative reconstruction from 400, 200 and 100 angles at the iteration when a given tolerance is achieved. Number of iterations is data dependent, for example, images (b), reconstructed from a complete set, are taken at iteration 12.

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 allows 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.

Acknowledgments

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 to test our algorithms on Euler cluster system.

References

Mora, C., 2008. Métodos de Reconstrucción Volumétrica Algebraica de Imágenes Tomográficas. Tesis Ph.D.

Crawford, B.M., Herman, G.T., 2007. Low-dose, large-angled cone-beam helical CT data reconstruction using algebraic reconstruction techniques. Image Vision Comput 25, 78–94.

Flores, L., Vidal, V., Mayo, P., Rodenas, F., Verdú, G., 2011. Iterative reconstruction of CT images with PETSc. BMEI 1, 343–346.

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

Gonzales, R.C., Woods, R.E., 2008. Digital Image Processing, 3rd ed. Prentice Hall, Upper Saddle River, NJ, USA.

Jang, B., Kaeli, D., Do, S., Pien, H., 2009. Multi GPU implementation of iterative tomographic reconstruction algorithms. Biomed. Imaging: From Nano to Macro. ISBI '09, 185–188.

Johnson, C.A., Sofer, A., 1999. A data-parallel algorithm for iterative tomographic image reconstruction. Front. Massively Parallel Comput. Frontiers '99, 126–137.

Nuyts, J., De Man, B., Dupont, P., Defrise, M., Suetens, P., Mortelmans, L., 1998. Iterative reconstruction for helical CT: a simulation study. Phys. Med. Biol. 43, 729–737.

Paige, C.C., Saunders, M.A., 1982. LSQR: an algorithm for sparse linear equations and sparse least squares. ACM Trans. Math. Software 8 (1), 43–71.

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 Trans. Med. Imaging 28 (3), 435–445.

Siddon, R., 1985. Fast calculation of the exact radiological path length for a three dimensional CT array. Med. Phys. 12, 252–255.

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. J. Parallel Distributed Comput. 68 (10), 1307–1318.

Wang, G., Yu, H., De Man, B., 2008. An outlook on X-ray CT research and development. Med. Phys. 35 (3), 1051–1064.

Wells, R.G., King, M.A., Simkin, P.H., Judy, P.F., Brill, A.B, Licho, R., Pretorius, P.H., Schneider, P.B., Seldin, D.W., 2000. Comparing Filtered back projection and ordered-subsets expectation maximization for small-lesion detection and localization in 67Ga SPECT. J. Nucl. Med. 41, 1391–1399.

⟨http://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf⟩2012.

⟨http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUBLAS_Library.pdf⟩2012.

⟨http://developer.download.nvidia.com/compute/DevZone/docs/html/CUDALibraries/doc/CUSPARSE_Library.pdf⟩2012.