ANN. MULT. GPU PROG.

GPU based algorithms in CT imaging

Liubov A. Flores[1*], Vicent Vidal[1] and Gumersindo Verdú[1]

[*]Correspondence:
liuflo@posgrado.upv.es
[1]Polytechnic University of Valencia, Camino de Vera s/n, 46022 Valencia, Spain

Full list of author information is available at the end of the article

Abstract

In X-ray computed tomography (CT) imaging, projections taken by a scanner are used to reconstruct the internal structure of an object. Due to the complexity of the data, the problem of reconstruction is a time consuming process. Although modern processors have gained sufficient power to be competitive in 2D reconstruction, it is not the case for 3D reconstruction especially when iterative methods are used. Today, the technology allows reducing this drawback effectively. In this work we compare two iterative algorithms of image reconstruction based on GPU implementation.

Keywords: CT image reconstruction; GPU based algorithm; CUDA C

1. Introduction

In medicine, the diagnosis based on computed tomography 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:

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

The reconstruction problem consists of determining the values of the function *f(x,y)* from the set of the experimental projection data *P*.

In the last three decades, in computer tomography have been proposed different methods to reconstruct images. If analytical methods have been derived from Radon transform [1], in iterative methods, it is optimized an objective function such as a function of maximum likelihood or minimum error. All iterative algorithms have in common operations that dominate the computational cost.

In CT, it is common to find under sampled set of no equally spaced projections. In these cases, images reconstructed with analytical methods 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 [2].

However, for practical use the iterative algorithms must be as efficient as possible.

The compute operations used in the reconstruction process are pixel-voxel operations. These operations have few dependencies and are executed in large loops. The appropriate platforms for such operations are vector processors or massively parallel architectures and graphical process units (GPUs).

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*. [3] 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 [4] 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* [5] 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 the 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 [7-8]. 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 algorithms.

The rest of the paper is organized as follows: in section 2, the mathematical aspects of two iterative algorithms more relevant to this work are described; GPU implementation of these algorithms is presented in section 3; Then we describe the methodology used to carry out experiments and present some results. Finally, we summarize our conclusions.

2. Theoretical considerations

In literature have been presented various algorithms to resolve the reconstruction problem (1) (see [9] for revision).

An algebraic approach to the reconstruction problem is reduced to the lineal system

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

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

In this approach, to reconstruct the internal structure of an object is equivalent to solve the system (2) in terms of measured projections.

Many properties of the reconstructed image depend on the approximations when calculating the system matrix of (2). 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. A could be computed previously, which would accelerate the reconstruction process.

In our work, we have implemented two iterative algorithms to resolve the reconstruction problem (2). In expectation maximization method (EM), first an object function is defined and then this function is optimized. It is considered that noise in projections follows the Poisson distribution [10].

2.1. Maximum likelihood expectation maximization for transmition tomography (MLEM)

In MLEM, it is looking for an image that makes more likely to occur the experimental data. The basic idea is as follows: for the given set of the experimental data y, find the distribution of the lineal attenuation coefficient μ that maximize the probability $P(\mu / y)$.

The iterations in MLEM have the following form:

$$x_j^{k+1} = x_j^k \sum_{i=1}^{m} \frac{p_i \, A_{ij}}{\sum_{i=1}^{m} x_i^k \, A_{ij}}, \tag{3}$$

where $x = \{x_j \mid j = 1,2,..n^2\}$ is an image vector of n^2 pixels, $p = \{p_i \mid i = 1,2...m\}$ represents the projection vector, $A = \{A_{ij}\}$ is the system matrix of $m \times n^2$, whose elements give the length of the segment of the i th X-ray going through the *j*th image pixel.

2.2. Least Square QR method (LSQR)

The second algorithm we implemented is the Least Square QR method (LSQR) [11]. This method solves 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 ATu for various vectors v and u. In this work we use Siddon's algorithm to compute elements of the matrix in a rectangular grid [12]. It has been found that Siddon's algorithm gives a good approximation of the system matrix A [13].

The main steps of LSQR:

$$
\begin{aligned}
&\textit{Initializa tion}: \\
&01: \quad \beta_1 u_1 = b \\
&02: \quad \alpha_1 u_1 = A^T u_1 \\
&03: \quad w_1 = v_1 \\
&04: \quad x_0 = 0 \\
&05: \quad \phi = \beta_1 \\
&06: \quad \rho = \alpha_1 \\
&\textit{Iteration} \quad \textit{process}: \\
&07: \quad \textit{for } i = 1 : \textit{iter} \max \\
&08: \qquad \beta_{i+1} u_{i+1} = A v_i - \alpha_i \\
&09: \qquad \alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i \\
&\qquad\quad \textit{Update} \quad \textit{scalars} \\
&\qquad\quad \textit{Update} \quad \textit{vectors}: \\
&10: \qquad x_i = x_{i+1} + (\phi_i / \rho_i) w_i \\
&11: \qquad w_{i+1} = v_{i+1} - (\theta_{i+1} / \rho_i) w_i
\end{aligned}
$$

3. GPU implementation

Computer graphic cards, such as NVIDIA Tesla K20c have been used to carry out the experiment. Such a GPU card has a total number of 2496 cuda cores with 5GB memory, shared by all processor cores. Utilizing such a GPU card with tremendous parallel computing ability considerably elevate the computation efficiency of our algorithms.

NVIDIA also introduced CUDA^T [14], 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.

One optimization technique considered in the implementation of LSQR is utilization of functions of CUBLAS [15] and CUSPARSE [16] libraries. CUBLAS 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 download 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.

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.

The most effective and important optimization opportunities are presented in exploration and effective use of the device memory. We use global memory of the device to allocate input data. However, the read only data have been allocated in constant memory and the fastest shared memory was used for temporary results whenever it was possible.

In case of MLEM, the execution kernel has been written. Selecting the correct size for a thread block is particularly key for performance since it determines the

number of threads that can be run simultaneously. We used 512 threads per block and chose to generate the number of blocks on a pixel bases.

4. Results and discussions

In the reconstruction process, we have analyzed two iterative algorithms (LSQR and MLEM) implemented en one GPU card. The results have been obtained on the system *gpu.dsic.upv.es with* CPU of 2.6 GHz and NVIDIA TESLA K20c GPU. The system belongs to the Departamento de Sistemas y Computación of Polytechnic University of Valencia.

We worked with real projections and reference images acquired from the Hospital Clinico Universitario in Valencia. The experimental data have been 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 reconstruction time for images of 256x256 and 512x512 pixels is given in Table1. 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).

System Matrix (rows x columns)	***CPU time (seconds)***	***GPUtime (seconds)***	***Speed Up Factor***
(256x100) x (256x256)	2.7	0.10	27
(256x200) (256x256)	5.3	0.18	29.4
(256x400) x (256x256)	10.5	0.32	32.8
(512x100) x (512x512)	12.3	0.33	37.3
(512x200) x (512x512)	24.5	0.67	36.6

Table1. Matrix size reconstruction time dependence for LSQR

GPU implementation of LSQR accelerates the reconstruction process up to 37 times. As for MLEM algorithm, only one iteration requires 132 seconds.

Figure 1 shows the resulting image reconstruction from running 10 iterations on one GPU. It was also important not to lose quality of reconstructed images.

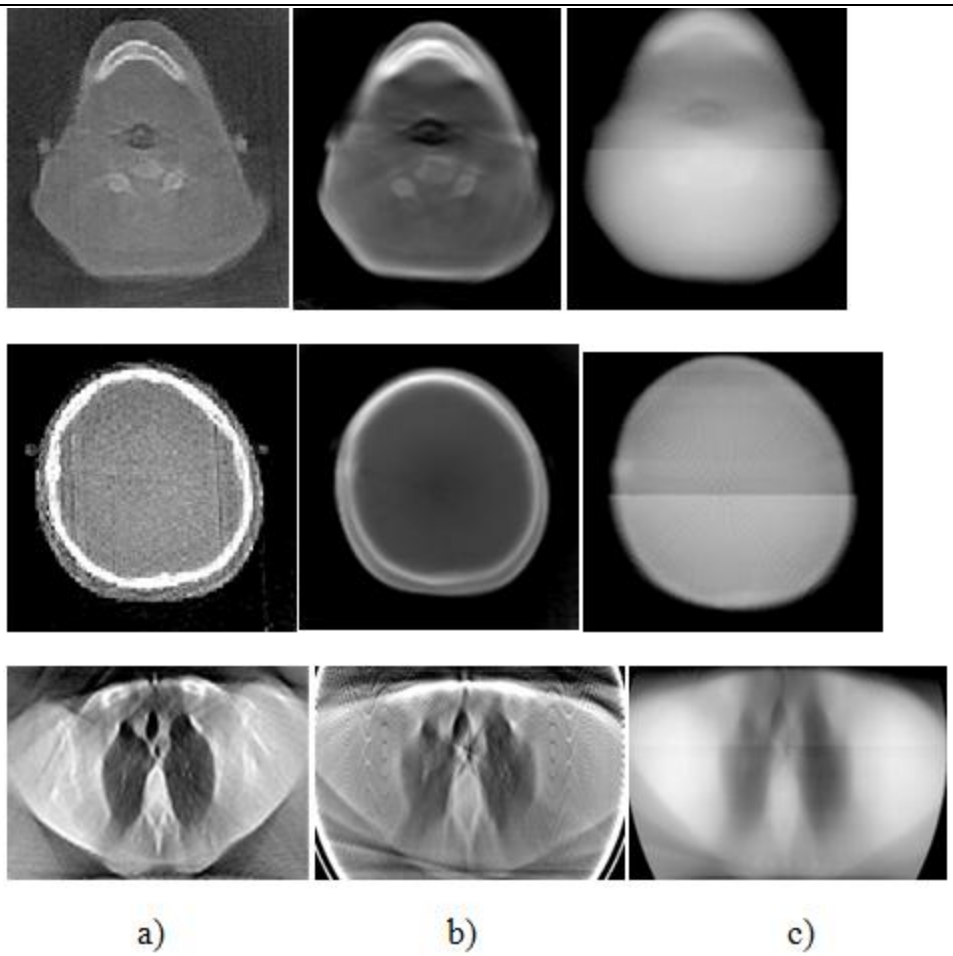

Figure 1. Reconstructed images: a) reference image, b) LSQR reconstruction, c) MLEM reconstruction

The quality comparison between the reference and reconstructed images has been performed and the results are summarized in Table 2. As a measure of quality the following comparisons between reference (I1) and reconstructed (I2) images have been used:

- Mean square error:

$$MSE = \sum_{i=1}^{n}\sum_{j=1}^{n}[I_1(i,j) - I_2(i,j)]^2,$$

- Peak signal-to-noise ratio:

$$PSNR = \frac{1}{nn}\log_{10}\frac{MAX_I^2}{MSE},$$

where n corresponds to the resolution (nxn pixels) of the reconstructed image, MAXI is the maximum possible pixel value of the image.

Image 256x256	**MSE**	**PSNR**
LSQR	0.0054	70.7817
MLEM	0.0355	62.6334

Table 2. Quantitative comparison of reconstructed images

5. Conclusions

Two methods have been analyzed in the iterative reconstruction of images. The results show that the GPU implementation of the Least Square QR method is capable to reconstruct images using under sampled set of projections with comparatively acceptable quality.

LSQR algorithm reconstructs images at low computational cost. This allows reducing the time of data acquisition process as well as the radiation dose for patients.

Although MLEM is one of the widely used algorithms, our implementation of it is not satisfactory and we continue our work on it.

Iterative algorithms do not need a complete set of projections which allows reducing the reconstruction time and could lead to more significant results in 3D reconstruction where a huge amount of computing is involved.

Acknowledgments

This work was partially funded by ANITRAN PROMETEO/2010/039, the Spanish Ministry of Science and Innovation (Project ENE2011-22823, TIN2011-26254).

References

1. R. S. Deans, The Radon transform and some of its applications. Dover Publications, INC. Mineola, New York, 2007.
2. G. Wang, H.Yu, and B. De Man, “An outlook on X-ray CT research and development”. Medical Physics, 35(3):1051-1064, Mar. 2008
3. 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.
4. Johnson C.A., Sofer. A., 1999. A data-parallel algorithm for iterative tomographic image reconstruction. Frontiers of Massively Parallel Computation, pp. 126-137.
5. 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.
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. Flores L., Vidal V., Mayo P., Rodenas F., Verdú G. Iterative reconstruction of CT images with PETSc. *BMEI 2011; vol.* 1 p. 343-346.
8. Flores L.,Vidal V., Mayo P., Rodenas F., Verdú G., ”Fast parallel algorithm for CT image reconstruction.” Proceedings of the IEEE EMBC , p. 4374-4377, 2012.
9. Herman, G. T. *Fundamentals of computerized tomography: Image reconstruction from projection*. 2nd ed. Springer; 2009.
10. L. A. Shepp and Y. Vardi, Maximum Likelihood Reconstruction for Emission Tomography. IEEE Transactions on Medical Imaging, vol. MI-1, NO. 2, October 1982
11. 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.
12. Siddon R.. Fast calculation of the exact radiological path length for a three dimensional CT array. *Med. Phys*. 1985; 12: p. 252-255.
13. T. Cibeles Mora Mora, Tesis PhD. Métodos de reconstrucción volumétrica algebraica de imágenes tomográficas, 2008.
14. Cuda C Programming Guide. Downloaded in Oct. 2012 from URL http://docs.nvidia.com/cuda/index.html.
15. CUBLAS Library. Downloaded in Oct. 2012 from URL http://docs.nvidia.com/cuda/index.html.
16. CUSPARSE Library. Downloaded in Oct. 2012 from URL http://docs.nvidia.com/cuda/index.html.