TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 02/20 pp11-16
Volume 15, Number 1, February 2010

Iterative Reconstruction for Transmission Tomography on GPU Using Nvidia CUDA

Damien Vintache**, Bernard Humbert, David Brasse

Institut Pluridisciplinaire Hubert Curien, CNRS/IN2P3, 23 rue du Loess BP28 67037 Strasbourg, France

Abstract: The iterative reconstruction algorithms for X-ray CT image reconstruction suffer from their high computational cost. Recently Nvidia releases common unified device architecture (CUDA), allowing developers to access to the processing power of Nvidia graphical processing units (GPUs), in order to perform general purpose computations. The use of the GPU, as an alternative computation platform, allows decreasing processing times, for parallel algorithms. This paper aims to demonstrate the feasibility of such an implementation for the iterative image reconstruction. The ordered subsets convex (OSC) algorithm, an iterative reconstruction algorithm for transmission tomography, has been developed with CUDA. The performances have been evaluated and compared with another implementation using a single CPU node. The result shows that speed-ups of two orders of magnitude, with a negligible impact on image accuracy, have been observed.

Key words: tomography; image reconstruction; parallel processing

Introduction

The Feldkamp algorithm[1] is the conventional tool to reconstruct images, in cone beam computed tomography (CT). Such algorithm is easy to implement, gives relatively good-quality images at high radiation doses, and can be started as soon as the acquisition process begins. Due to the large amount of data to be processed, in transmission tomography, the Feldkamp algorithm is often preferred compared to iterative algorithms that are more time-consuming. However, the image quality suffers from artifacts such as streaks due to detector or source characteristics.

Indeed several iterative reconstruction algorithms have been proposed for the X-ray CT image reconstruction, including the family of algebraic techniques[2] commonly known as algebraic reconstruction technique (ART), simultaneous iterative reconstruction technique (SIRT), as well as statistically based methods like maximum likelihood-expectation maximization (ML-EM)[3] and ordered subsets convex algorithm (OSC)[4]. Generally speaking, these algorithms are considered to produce images of high quality, but the computational cost is unfortunately also high. Due to recent advances in computer technology, the iterative reconstruction may prove to be practicable. Such iterative algorithms are good candidates for an implementation on graphical processing unit, as they are highly parallelizable.

Statistical algorithms allow correcting image artifacts and producing higher quality images, modeling the source emission, the photon transport, and the detection physical process[5,6].

Kole and Beekman[7] proposed an evaluation of iterative CT image reconstructions using the OSC algorithm implemented on graphics hardware, OpenGL. Depending on the volume and the detector size, speed-ups of factor 40 to 220 can be achieved, compared with the conventional CPU implementation.

Received: 2009-10-20; revised: 2009-12-10

** To whom correspondence should be addressed.
E-mail: damien.vintache@iphc.cnrs.fr; Tel: 33-3-88106350

Weinlich et al.[8] propose a comparison of high-speed ray casting on the graphical process unit (GPU) using common unified device architecture (CUDA) and OpenGL and have shown that the performance of the recent CUDA version is slightly better than an implementation using OpenGL. Moreover, the OpenGL implementation requires much more knowledge in computer graphics and implementation time.

In this article, we propose to implement the OSC algorithm using the Nvidia CUDA architecture.

1 Statistical Algorithms for Transmission Image Reconstruction

1.1 Statistical algorithms for transmission tomography

Lange and Fessler[3] first proposed to compute the maximum-likelihood estimate of transmission images via an EM algorithm based on Poisson statistics. For twenty years, several reconstruction algorithms based on the Poisson statistical model for transmission images and converging faster than the original ML-EM algorithm were introduced[4,7,9,10].

Iterative algorithms are based on the same scheme: to reconstruct a 3-dimensional (3-D) image from 2-dimensional (2-D) projections. An estimated volume image is chosen, this volume is projected in the detector plan for each acquisition angle. Then estimated projections are compared to acquired projections, correction coefficients are computed and applied to the estimated volume image.

Using ordered subsets is one way to reduce reconstruction time in statistical algorithms. Instead of modifying the estimated image volume for each projection angle, the correction is applied for each subset of projection angles. The reconstruction time decreases but this does not impact image quality.

1.2 Ordered subsets convex algorithm

In this article, we focus on one of these OSC algorithms, introduced by Beekman and Kamphuis[4], which iteratively tries to improve an image estimate to a solution[11-13].

As described in Ref. [14], this algorithm is based on the following update equation:

$$f_{v+1} = f_v \left(1 + \frac{\boldsymbol{R}_v^{\mathrm{T}}(\mathrm{e}^{-R_v f_v} - \mathrm{e}^{-p_v})}{\boldsymbol{R}_v^{\mathrm{T}}(\mathrm{e}^{-R_v f_v} \cdot R_v f_v)} \right) \tag{1}$$

where v denotes a given subset. f_v is the estimation of the attenuation coefficient for the subset v. p_v is the set of measured projections for the subset v. R_v denotes the projection operator from the volume image space to the detector plane space for subset v. $R_v f_v$ is the estimated projection of f at subset v. $\boldsymbol{R}_v^{\mathrm{T}}$ is the transposed operation, backprojection from detector plane space to volume image space for subset v.

After iterating through all angles of a subset, the estimated attenuation coefficients are updated for each voxel of the volume. The numerator in Eq. (1) corresponds to the backprojection of an error computed as the difference for each pixel between the exponential of the estimated value and the exponential of the measured value. The denominator in Eq. (1) is the backprojection of the product, for each pixel, of the estimated value and the exponential of this estimated value.

2 Operators

Two main functions, corresponding to the two geometrical operators of the forward projection and the backprojection, have been written to implement the OSC algorithm.

2.1 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. The following algorithm is used: For each pixel of the detector:

- Compute the unit vector on the ray from the source point to the current pixel;
- Choose the integration direction given by the largest coordinate of the unit vector;
- Scan all the voxels along this direction, compute their contribution in the projection via a bilinear interpolation;
- Set the value of the pixel intensity as the sum of these contributions.

2.2 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. The following algorithm is used:

For each voxel of the volume:

- Get the coordinates of its projection point;
- Compute the intensity value of this projection point via bilinear interpolation;
- Weight this value, according to the distance from the voxel to the projection point;
- Add this weighted value to the current voxel attenuation.

The forward projection, backprojection, and estimated volume correction are the more computationally expensive operations of the OSC algorithm and also parallelizable tasks. To get an accurate image in a shortest delay, we choose to perform these tasks on a graphical processor unit using CUDA, the hardware, and software architecture developed by Nvidia for GPU computing.

3 Implementation

3.1 CUDA architecture

CUDA refers to the hardware architecture as well as the software platform for massively parallel high performance computing. Designed by Nvidia, CUDA includes C/C++ software-development tools, function libraries, and an hardware abstraction mechanism that hides the GPU hardware from developers[15].

GPU is used as a data-parallel computing device capable of executing a very high number of automatically managed threads. The compilation result of a C function is called “kernel”. This kernel is downloaded and executed on multiprocessors of the GPU, as batch of threads.

With a minimal set of extensions to the C language, CUDA allows the developer to use the computing power of GPU without mapping the general purpose problem to the graphics application programmable interface (API), e.g., OpenGL.

Recently, several implementations of forward projection and backprojection using CUDA have been proposed by Refs. [16-18].

3.2 Implementation details

To get the lowest computation times, we have to avoid memory transfers between CPU and GPU. Textures are of interest in our case, because they provide low latency to access the memory and are able to perform hardwired interpolations.

Three kernels have been implemented to perform parallel computation on GPU, corresponding to the forward projection, backprojection, and estimated volume update operations.

4 Tests Description

4.1 Test-bench

The machine used to perform the tests contained an Intel Core 2 Quad Q6600 2.4 GHz CPU, 1 GB RAM, and an Nvidia GeForce 8800 GT with 1 GB VRAM. It runs on a Linux operation system.

4.2 Phantom and simulation set-up

To test our framework, we use a 3-D version of the Shepp-Logan phantom. The voxel size is 0.1×0.1×0.1 mm^3. Five hundred projections have been generated using the CPU implementation of the forward projection operator and an angular gap between two consecutive projections of 2xPI/500 radians. The size of the projection pixels is 0.05×0.05 mm^2. The focal length (distance between the cone vertex and the detector plane) equals 100 mm and the distance between the cone vertex and the rotation center is 50 mm.

Figure 1 presents the projection of the Shepp-Logan phantom at 0°.

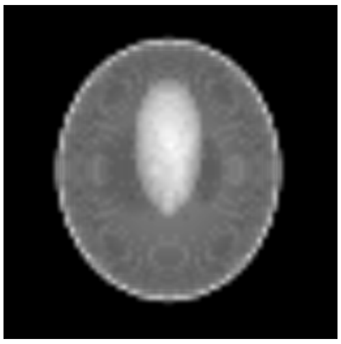

Fig. 1 Projection of the Sheep-Logan phantom at 0° on the 256^2 detector grid

As initial volume image estimate, a cube with a uniform attenuation factor of 0.01 is used.

4.3 Assessment of image accuracy

To assess the reconstructed image accuracy, four different figures of merit are described in Ref. [7].

For preliminary results, we just focus on the normalized mean square error (NMSE) defined as

$$\mathrm{NMSE}=\frac{\sqrt{\sum_k(\mu(k)-\nu(k))^2}}{\sqrt{\sum_k(\mu(k))^2}} \tag{2}$$

where $\mu(k)$ denotes the attenuation of the voxel k in the phantom and $\nu(k)$ denotes the reconstructed attenuation of the voxel k.

5 Results

5.1 Comparison between CPU and GPU implementations

A basic CPU version of the OSC algorithm is implemented to get a reference of image accuracy and processing time that we could expect. In this part the Sheep-Logan phantom is sampled on a 64^3 voxels grid and projected on a 256^2 pixels grid.

The basic CPU implementation only uses one thread on one processor core: no optimization has been performed.

The same algorithms for the forward projection, the backprojection, and the volume correction, are implemented using CUDA.

Table 1 gives the time performance measurements for operators like the forward projection, the backprojection, and the volume correction. The projection processing corresponds to the set of instructions related to the same projection (parameters initialization, measured projection reading, estimated volume forward projection, error, and product backprojection). GPU times are given with and without memory transfer between CPU and GPU. Speedup factors between CPU and GPU implementation are given in parenthesis.

Table 1 Time performance measurements

Operator	CPU time (μs)	GPU time with memory transfer in speedup (μs)	GPU time without memory transfer in speedup (μs)
Forward projection	110 730	10 910 (10)	9870 (11)
Backprojection	10 050	1820 (6)	522 (19)
Projection processing	135 190	13 130 (10)	11 381 (12)
Volume correction	4280	740 (6)	105 (41)

NMSE and total processing time depend on subset size and iteration number. Ten iterations have been performed for each test. We use two configurations in both GPU and CPU tests, varying the number N of projections in subsets. The projections in subsets are defined by the angle of projection, i.e. the subset S_n can be defined by the set θ_n of angles used to perform the projections.

$\theta_n=\{n, n+\Delta\theta, n+2\cdot\Delta\theta,\ldots,n+k\cdot\Delta\theta,\ldots, n+(N-1)\cdot\Delta\theta\}$,

where n is varying from 0 to $500/N$; $\Delta\theta$ is the angular gap between two successively acquired projections.

Table 2 gives the normalized mean square error for GPU and CPU implementations with 10 subsets of 50 projections or with 50 subsets of 10 projections. The volume correction is performed at the end of each subset processing: the configuration using 50 subsets gives more accurate results. GPU and CPU implementations give the same NMSE value in both configurations.

Table 2 NMSE for GPU and CPU

Subset number	NMSE	
	CPU	GPU
10 subsets with 50 projections	4.8×10^{-2}	4.8×10^{-2}
50 subsets with 10 projections	0.9×10^{-2}	0.9×10^{-2}

As, in our case, a very small volume size is used, the volume correction time is lower than projection processing, the subset configuration does not affect total reconstruction time.

Table 3 shows performances with 10 subsets of 50 projections.

Table 3 Time performance measurements for 10 subsets of 50 projections

Operation	CPU time (s)	GPU time with memory transfer in speedup (s)	GPU time without memory transfer in speedup (s)
Subset processing	6.8	0.7 (10)	0.6 (11)
Iteration processing	67.0	6.6 (10)	5.6 (12)
Total reconstruction time	677.0	66.0 (10)	58.0 (12)

Figure 2 presents the central axial slice of the Sheep-Logan phantom (left image) and the central axial slice of the reconstructed volume using GPU with fifty subsets of ten projections after ten iterations (right image). In this case, NMSE is lower than 1×10^{-2}.

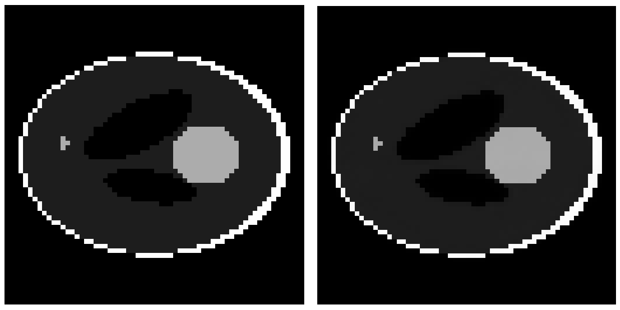

Fig. 2 Reconstruction results after ten iterations: Sheep-Logan phantom (left) reconstructed volume using GPU (right)

5.2 Optimization

The reconstruction qualities using CPU or GPU are very similar. We optimized the reconstruction time, using CUDA textures that allow us to perform hardware interpolation and provide a cached memory access, in read mode.

In the forward projection step, the estimated volume is mapped to a 3-D texture. In the backprojection step, error and product projections are mapped to 2-D textures.

Using textures gives us a speed-up factor of 7.5 compared to the un-optimized GPU version and a speed-up factor of 76 compared to the un-optimized CPU version.

Several configurations have been tested with different volume and projection plan sizes. The times needed for each operation are given in Table 4.

The read projection times and the backprojection times depend on the projection size. The correction time changes with the volume size. The projection time depends on the volume size and the projection size.

Table 4 Time performance measurements for 10 subsets of 50 projections

Volume and projection sizes	Time (s)				
	Read projection	Projection	Back projection	Correction	Total
64^3 256^2	1.1	5.5	0.3	1	9
64^3 512^2	4.0	17.0	1.0	1	25
64^3 1024^2	172.0	64.0	2.0	1	240
64^3 2048^2	654.0	242.0	8.0	1	907
128^3 1024^2	185.0	91.0	2.0	7	292
128^3 2048^2	662.0	333.0	8.0	8	1020
300^3 2048^2	621.0	656.0	8.0	88	1435

6 Discussion

CUDA implementation of the OSC algorithm gives us good performances, in terms of image quality and reconstruction time. The next step consists of reconstructing small animal images and trying to reduce artifacts due to detector or source characteristics.

7 Conclusions

In this paper, we have shown that using a GPU as computing coprocessor for iterative image reconstruction in transmission tomography can easily yield speed-ups of two orders of magnitude, with negligible impact on image accuracy. Nvidia CUDA has been used to implement the OSC algorithm on graphics hardware, with a minimal learning cost and in a very short time.

References

[1] Feldkamp L A, Davis L C, Kress J W. Practical cone-beam algorithm. *Journal of the Optical Society of America A*, 1984, **1**(6): 612-619.

[2] Kak A C, Slanet M. Principles of Computerized Tomographic Imaging. New York: IEEE Press, 1988.

[3] Lange K, Fessler J A. Globally convergent algorithms for maximum a posteriori transmission tomography. *IEEE Transactions on Image Processing*, 1995, **4**(10): 1430-1438.

[4] Beekman F J, Kamphuis C. Ordered subset reconstruction for x-ray CT. *Physics in Medicine and Biology*, 2001, **46**: 1835-1844.

[5] Elbakri I A, Fessler J A. Statistical image reconstruction for polyenergetic x-ray computed tomography. *IEEE Transactions on Medical Imaging*, 2002, **21**(2): 89-99.

[6] Chueh H S, Tsai W K, Chang C C, et al. An iterative reconstruction for poly-energetic X-ray computed tomography. *Medical Imaging and Informatics*, 2008, **4987**: 44-50.

[7] Kole J S, Beekman F J. Evaluation of accelerated iterative x-ray CT image reconstruction using floating point graphics hardware. *Physics in Medicine and Biology*, 2006, **51**: 875-889.

[8] Weinlich A, Keck B, Scherl H, et al. Comparison of high-speed ray casting on GPU using CUDA and openGL. In: Proceedings of the First International Workshop on New Frontiers in High-Performance and Hardware-Aware Computing. Lake Como, Italy, 2008.

[9] Ollinger J M. Maximum-likelihood reconstruction of transmission images in emission computed tomography via the EM algorithm. *IEEE Transactions on Image Processing*, 1994, **13**(1): 89-101.

[10] Shetye A, Shekhar R. A statistical approach to high-quality CT reconstruction at low radiation doses for real-time guidance and navigation. In: Proceedings of SPIE Medical Imaging 2007. San Diego, CA, 2007, **6510**: 65105U.

[11] Kole J S, Beekman F J. Evaluation of the ordred subset convex algorithm for cone-beam CT. *Physics in Medicine and Biology*, 2005, **50:** 613-623.

[12] Erdogan H, Fessler J A. Ordered subsets algorithms for transmission tomography. *Physics in Medicine and Biology*, 1999, **44:** 2835-2851.

[13] Quan E, Lalush D S. A faster orederd-subset convex algorithm for iterative reconstruction in rotation-free micro-CT system. *Physics in Medicine and Biology*, 2009, **54**: 1061-1072.

[14] Kachelrieβ M, Berkus T, Kalender W. Quality of statistical reconstruction in medical CT. In: Proceedings of IEEE Nuclear Science Symposium Conference. Portland, 2003, **4**: 2748-2752.

[15] NVIDIA. NVIDIA CUDA Compute Unified Device Architecture Programming Guide. http://developer.download.nvidia.com/computer/cuda/2.0/docs/NVIDIA_CUDA_Programming-Guide_2.0.pdf, 2007.

[16] Yang H, Li M, Koizumi K, et al. Accelerating backprojections via CUDA architecture. In: Proceedings of the 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine. Lindau, Germany, 2007: 52-55.

[17] Scherl H, Keck B, Kowarschik M, et al. Fast GPU-based CT reconstruction using the common unified device architecture (CUDA). In: Proceedings of 2007 IEEE Nuclear Science Symposium Conference. Honolulu, Hawaii, 2007: 4464-4466.

[18] Knaup M, Steckmann S, Kachelrieβ M. GPU-based parallel-beam and cone-beam forward- and backprojection using CUDA. In: Proceedings of 2008 IEEE Nuclear Science Symposium Conference. Dresden, Germany, 2008: 5153-5157.