Multi GPU parallelization of 3D bayesian CT algorithm and its application on real foam reonconstruction with incomplete data set

Nicolas Gac, Alexandre Vabre, Ali Mohammad-Djafari

Multi-GPU parallelization of a 3D Bayesian CT algorithm and its application on real foam reconstruction with incomplete data set

Nicolas Gac[1,a], Alexandre Vabre[2,b], and Ali Mohammad Djafari[1,c]

[1] L2S, Laboratoire des Signaux et Systemes (CNRS-SUPELEC-UPS), F-91191 Gif sur Yvette, France
[2] CEA, LIST, Laboratoire Images et Dynamique, , F-91191 Gif sur Yvette, France

Abstract. A great number of image reconstruction algorithms, based on analytical filtered backprojection, are implemented for X-ray Computed Tomography (CT) [1,2]. The limits of these methods appear when the number of projections is small, and/or not equidistributed around the object. That's the case in the context of dynamic study of fluids in foams for example, the data set are not complete due to the limited acquistion time. In this specific context, iterative algebraic methods are a solution to this lack of data. A great number of them are mainly based on least square criterion. Recently, we proposed a regularized version based on Bayesian estimation approach. The main problem that appears when using such methods as well as any iterative algebraic methods is the computation time and especially for projection and backprojection steps. In this study, first we show how we implemented some main steps of such algorithms which are the forward projection and backward backprojection steps on multi-GPU hardware, and then we show some results on real application of the 3D tomographic reconstruction of metallic foams from a small number of projections. Through this application, we also show the good quality of results as well as a significant speed up of the computation with GPU implementation (300 acceleration factor).

1 Iterative method

The inverse problem we solve is to reconstruct the object f from the projection data g collected by a cone beam 3D CT [3]. The link between f and g can be expressed as :

$$g = Hf + \epsilon \tag{1}$$

where H is the forward projection matrix operator modeling the acquisition system and ϵ represents all the errors (modeling and measurement noise). The element H_{ij} represents the participation of the j pixel in the i detector.

In this discretized presentation of the CT forward problem, the backprojection (BP) solution can be expressed as $\widehat{f_{BP}} = H^t g$ where H^t is the transpose of H and the filtered backprojection (FBP) (like the FDK method [4]) which is also equivalent to the Least squares (LS) solution can be expressed as $\widehat{f_{FBP}} = (H^t H)^{-1} H^t g$. The LS solution $\widehat{f_{LS}} = \arg\min_f \left\{ Q(f) = \|g - Hf\|^2 \right\}$ as well as the quadratic regularization (QR) solution

$$\widehat{f_{QR}} = \arg\min_f \left\{ J(f) = \|g - Hf\|^2 + \lambda \|Df\|^2 \right\} \tag{2}$$

[a] e-mail: nicolas.gac@lss.supelec.fr
[b] e-mail: alexandre.vabre@cea.fr
[c] e-mail: djafari@lss.supelec.fr

can be obtained by a gradient based optimization algorithm which can be described as follows:

$$\begin{cases} f^{(0)} = H^t g \\ f^{(i+1)} = f^{(i)} + \alpha \left[H^t (g - H f^{(i)}) + \lambda D^t D f^{(i)} \right] \end{cases} \tag{3}$$

where α is a fixed, variable or computed optimally step and (i) is the iteration number. Looking at this iterative algorithm, we can distinguish, at each iteration the following operations:

1. Forward projection operation: $\widehat{g} = H\widehat{f}$
2. Computation of the residuals: $\delta g = g - \widehat{g}$
3. Backprojection operation of the residual: $\delta f_1 = H^t \delta g$
4. Computation of the regularization or a priori term: $\delta f_2 = \lambda D^t D \widehat{f}$
5. Updating of the solution for the next iteration:
   $f^{(i+1)} = f^{(i)} + \alpha(\delta f_1 + \delta f_2)$

2 GPU Parallelization

For the iterative step of gradient descent, the two main consuming time operations are projection (Hf) and backprojection ($H^t \delta g$) which are used to estimate a convergence criterion and its gradient. These two operations represent 99 % of the computing time.

The follow up of the work aims at speeding up these two steps. GPU hardware, since 2006 is one of the most used tool inside research community. Both simplicity in implementation and performance improvements have imposed scientic community to migrate to such a tool. Recent improvements from Nvidia have allowed to dispose of CUDA, this developing environment allows to design operating software with high computing performances [5].

In order to compute the two matrix operations (Hf and $H^t \delta g$) without the too expensive memory use of H=(h_{ij}) (1 To is needed to store H for a 2048^3 reconstruction), projection and backprojection geometric operators are widely used. This operators compute in line the coefficient h_{ij}, instead of reading a matrix H stored in memory. For each operator, we choose the one which enables the best implementation on Nvidia GPUs with CUDA. As a consequence, our projection/backprojection pair is unmatched. Thus each operator defines a different matrix H: H_p for projection and H_{bp} for backprojection. Use of unmatched backprojection/projection pairs is widely used. Indeed, effect on convergence is in pratice not penalising during the first iterations [6]. Main difference on backprojection and projection algorithm is the main loop of computation : for backprojection, the loop is on voxels (voxel-driven) and for projection it is on X rays (ray-driven).

Afterwards in order to gain another acceleration factor, we have parallelized on a server with 8 GPU boards (8 Tesla C1060 on a server provided by the *Carri Systems* compagny). In order to store all the data on the 4 Go SDRAM memory of the GPU board, the data have been distributed as illustrated on the figure 1.

3 Reconstruction Time

On table 1, we present the acceleration speed up, we have obtained for one iteration of our regularized reconstruction method applied on a 1024^3 voxels volume from 256 projections on a 1024^2 pixels detector. Four versions of our implementation are presented : (v1) all steps made on CPU (non optimized code), (v2) backprojection and projection made on one GPU (Tesla C1060), (v3) backprojection and projection made on 8 GPUs, (v4) discrete derivation made with a 3D convolution of size 3^3 on one GPU. Finally, we obtain a 300 acceleration factor compared to non optimized CPU implementation.

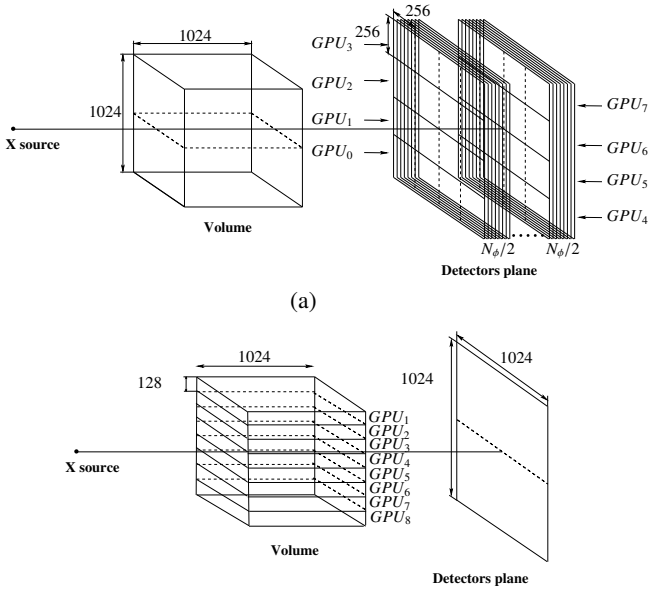


Fig. 1. multi-GPU Parallelization of the projection (a) and the backprojection (b).

Operators	**Computing Time**			
	v1	v2	v3	v4
Projection $2 \times H_P$	4.1 h *(42.5 %)*	7.1 mn *(64.9 %)* $\rightarrow \times$ **35**	57 s *(21.1 %)* $\rightarrow \times$ **7**	57 s *(63.3 %)*
Backprojection H^t_{RP}	5.5 h *(56.9 %)*	21.8 s *(3.3 %)* $\rightarrow \times$ **908**	4.0 s *(1.5 %)* $\rightarrow \times$ **5**	4.0 s *(4.4 %)*
Convolution $3 \times D$	3.2 mn *(0.6 %)*	3.2 mn *(29.2 %)*	3.2 mn *(71.1 %)*	12.1 s *(13.4 %)* $\rightarrow \times$ **16**
Autre	17 s *(0.0 %)*	17 s *(2.6 %)*	17 s *(6.3 %)*	17 s *(18.9 %)*
Total	9.7 h	10.9 mn $\rightarrow \times$ **53**	4.5 mn $\rightarrow \times$ **2.4**	1.5 mn $\rightarrow \times$ **3.0**

v1 : H_P, H^t_{RP} and D on CPU
v2 : H_P and H^t_{RP} on 1 GPU, D on CPU
v3 : H_P and H^t_{RP} on 8 GPUs, D on CPU
v4 : H_P and H^t_{RP} on 8 GPUs, D on 1 GPU

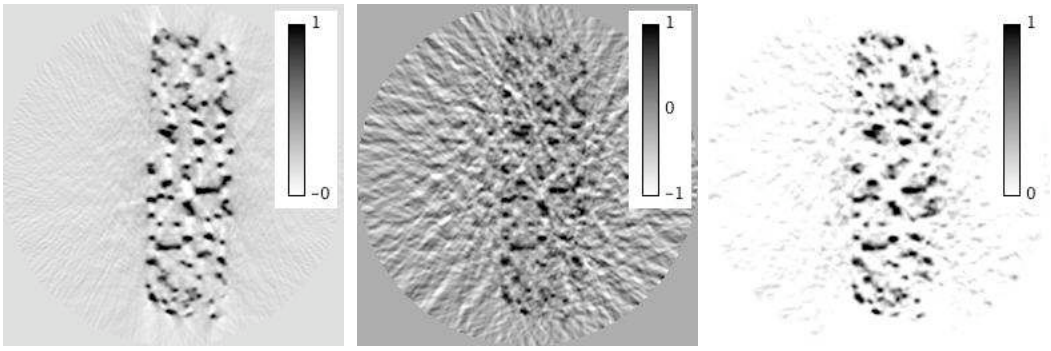
Table 1. Processing time for one iteration of a 1024^3 volume reconstruction

4 Real data reconstruction

4.1 Metallic foams

Solid foams are a class of materials with a complex behavior related to the properties of the constitutive material, the geometry and the topology of the material distribution [7]. These materials present a very high porosity, and are thus very light, but nevertheless very resistant due to a good distribution and architecture of matter. The most known examples of such materials are bone and wood, or also coral and sponge. Data set is made of 96 projections on the 256^2 detectors plane.

4.2 Foams reconstructed



(a) (b) (c)

Fig. 2. 256^3 foam reconstructions from N projections : direct method (filtered backprojection) with N=256 (a) and N=32 (b); regularized method with N=32 (c).

As illustrated on figure 2, the filtered backprojection (non iterative method) is creating lots of artefacts of reconstruction when the number of projection is only N=32 (incomplete data set). Our regularized present still a good quality of reconstruction even with N=32 projections.

5 Conclusion

We proved on a real data set that regularized methods offer a real benefit against standard filtered backprojecton methods when the data set is incomplete. We have accelerated on a 8 GPU server by a 300 factor a 1024^3 volume reconstruction with a such method. We are able to do one iteration of our regularized method in 1.5 minute. In perspective, we will explore others regularized methods on real data set thanks to this reduced reconstruction time.

References

1. K.J. Batenburg and J. Sijbers, "DART: a fast heuristic algebraic reconstruction algorithm for discrete tomography," 2007, vol. IV, pp. 133–136, IEEE Conference on Image Processing.
2. Steckmann S. Marone F. Kachelrie M., Knaup M. and Stampanoni M., "Hyperfast o(2048**4) image reconstruction for synchrotron?based x?ray tomographic microscopy," in *MIC proceedings*, 2008.
3. A. Mohammad-Djafari, Ed., *Inverse Problems in Vision and 3D Tomography*, ISTE, 2009.
4. L.A. Feldkamp, Davis L C, and Kress J W, "Practical cone-beam algorithm," *J Opt Soc Am*, vol. A6, pp. 612–619, 1984.
5. N. Gac et al, "High speed 3D tomography on CPU, GPU and FPGA," *EURASIP Journal on Embedded systems*, 2008.
6. G.L. Zeng and G.T. Gullberg, "Unmatched projector/backprojector pairs in an iterative reconstruction algorithm," *Medical Imaging, IEEE Transactions on*, vol. 19, no. 5, pp. 548–555, May 2000.
7. O. Gerbaux et al, "Transport properties of real metallic foams," *J Colloid & Interface Science*, vol. 342, pp. 155–165, 2010.