On the Efficiency of Iterative Ordered Subset Reconstruction Algorithms for Acceleration on GPUs

Fang Xu[1], Klaus Mueller[1],
Mel Jones[2], Bettina Keszthelyi[2], John Sedat[2], David Agard[2]

[1] Center for Visual Computing, Computer Science Department, Stony Brook University, NY
[2] Howard Hughes Medical Institute and the Keck Advanced Microscopy Center,
Department of Biochemistry & Biophysics, University of California at San Francisco, CA
Author contact: mueller@cs.sunysb.edu

Abstract. Expectation Maximization (EM) and the Simultaneous Iterative Reconstruction Technique (SIRT) are two iterative computed tomography reconstruction algorithms often used when the data contain a high amount of statistical noise, such as in functional imaging, or have been acquired from a limited angular range, such as in electron microscopy. A popular mechanism to increase the rate of convergence of these types of algorithms has been by performing the correctional updates within subsets of the projection data. This has given rise to the method of Ordered Subsets EM (OS-EM) and the Simultaneous Algebraic Reconstruction Technique (SART). However, we find that the special architecture and programming model of commodity graphics hardware (GPU) adds extra constraints on the real-time performance of ordered subsets algorithms. These counteract the speedup benefits of smaller subsets. Here, we study this behavior in the context of algebraic reconstruction, but similar trends are likely also observed with EM. We find that there are optimal subset sizes in terms of wall-clock time speed, given pre-set quality criteria.

Keywords: Iterative reconstruction, Computed tomography, Commodity graphics hardware, GPU

1 Introduction

The rapid growth in speed and capabilities of programmable commodity graphics hardware boards (GPUs) has propelled high performance computing to the desktop, spawning applications far beyond those used in interactive computer games. High-end graphics boards, such as the NVIDIA 8800GTX, featuring 500 G Flops and more, are now available for less than $500, and their performance is consistently growing at a triple of Moore's law that governs the growth of CPUs. Speedups of 1-2 orders of magnitude have been reported by many researchers when mapping CPU-based algorithms onto the GPU, in a wide variety of domains [15], including medical imaging [1][10][12][14]. These impressive gains originate in the highly parallel SIMD (Same Instruction Multiple Data) architecture of the GPU and its high-

bandwidth memory access. For example, one of the current top boards, the NIVIDIA 8800 GTX, has 128 such SIMD pipelines.

It is important to note, however, that the high speedup rates facilitated by GPUs do not come easy. They require one to carefully map the target algorithm from the single-threaded programming model of the CPU to the multi-threaded SIMD programming model of the GPU where each such thread is dedicated to computing one element of the (final or intermediate) result vector. Here, special attention must be paid to keep all of these pipelines busy. While there are 128 SIMD processors on the GPU, many more threads need to be created to hide data fetch latencies. It is important to avoid both thread congestion (too many threads waiting for execution) and thread starvation (not enough threads available to hide latencies). These conditions are in addition to avoiding possible contingencies in local registers and caches that will limit the overall number of threads permitted to run simultaneously. For example, in [12], it was shown that a careful mapping of Feldkamp's filtered backprojection algorithm to the GPU yielded a 20x speedup over an optimized CPU implementation, enabling cone-beam reconstructions of 512^3 volumes from 360 projections at a rate of 52 projections/s, greatly exceeding the data production rates of modern flat-panel X-ray scanners that have become popular in fully-3D medical imaging.

The subject of this paper is to explore iterative reconstruction algorithms in terms of these concerns. Iterative algorithms are different from analytical algorithms in that they require frequent synchronization which interrupts the stream of data, requires context switches, and limits the number of threads available for thread management. Iterative algorithms, such as Expectation Maximization (EM) [11] or the Simultaneous Iterative Reconstruction Technique (SIRT) [5] consist of three phases, executed in an iterative fashion: (1) projection of the object estimate, (2) correction factor computation (the updates), and (3) backprojection of the object estimate updates. Each phase requires a separate pass. Flexibility comes from the concept of ordered subsets, which have been originally devised mostly because they accelerated convergence. The projection data is divided into groups, the subsets, and the data within each of these groups undergo each of the three phases simultaneously. Here, it was found that the larger the number of subsets (that is, the smaller the groups) the faster is typically the convergence, but adversely also the higher the noise since there is more potential for over-correction. In EM, the method of Ordered Subsets (OS-EM) has become widely popular. OS-EM conceptually allows for any number of subsets, but the limit with respect to noise has been noted already in the original work by Hudson and Larkin [7]. For the algebraic scheme, embodied by SIRT, the Simultaneous Algebraic Reconstruction Technique (SART) [2] is also an OS scheme, but with each set only consisting of a single projection. In SART, the over-correction noise is kept low by scaling the updates by a relaxation factor $\lambda<1$. Block-iterative schemes for algebraic techniques have been proposed as well [3]. In fact, the original ART [6] is the algorithm with the smallest subset size possible: a single data point (that is, ray or projection pixel).

It is well known that SART converges much faster than SIRT, and a well chosen λ can overcome the problems with streak artifacts and reconstruction noise, allowing it produce good reconstruction results [1]. On the CPU, faster rate of convergence is directly related to faster time performance. But, as we shall show, when it comes to

acceleration on a streaming architecture such as the GPU, SART is not the fastest algorithm in terms of time performance. In fact, the time performance is inversely related to the number subsets, making SIRT the faster scheme. This due to the overhead incurred by the frequent context switching when repeatedly moving through the three iterative phases: projection, correction, and backprojection. In our experiments, we have found that there are specific subset sizes that optimize both reconstruction quality and performance. Here we note, however, that the optimal setting is likely application dependent, making the reasonable assumption that a certain application will always incur similar types of data and thus an optimal parameter setting, once found, will be close to optimal for all data within that application setting. In that sense, our aim for this paper is not to provide an optimal subset setting for all types of data, but rather to raise awareness to this phenomenon and offer an explanation.

Our paper is structured as follows. First, in Section 2, we discuss iterative algorithms in the context of ordered subsets, present a generalization of SIRT to OS SIRT, and describe their acceleration on the GPU. Then, in Section 3, we study the impact of subset size on GPU reconstruction performance and present the results of our studies. Finally, Section 4 ends with conclusions.

2 Iterative Reconstruction and its Acceleration on GPUs

In the following discussion, we have only considered algebraic reconstruction algorithms (SART, SIRT), but our arguments and conclusions readily extend to expectation maximization (EM) algorithms as well since they are very similar with respect to their mapping to GPUs [13].

2.1 Iterative Algebraic Reconstruction: Decomposition into Subsets

Most iterative CT techniques use a projection operator to model the underlying image generation process at a certain viewing configuration (angle) φ. The result of this projection simulation is then compared to the acquired image obtained at the same viewing configuration. If scattering or diffraction effects are ignored, the modeling consists of tracing a straight ray r_i from each image element (pixel) and summing the contributions of the (reconstruction) volume elements (voxels) v_j. Here, the weight factor w_{ij} determines the contribution of a v_j to r_i and is given by the interpolation kernel used for sampling the volume. The projection operator is given as:

$$r_i = \sum_{j=1}^{N} v_j \cdot w_{ij} \qquad i = 1, 2, ..., M \tag{1}$$

where M and N are the number of rays (one per pixel) and voxels, respectively. Since GPUs are heavily optimized for computing and less for memory bandwidth, computing the w_{ij} on the fly, via bilinear interpolation, is by far more efficient than

storing the weights in memory. The correction update for projection-based algebraic methods is computed with the following equation:

$$v_j^{(k+1)} = v_j^{(k)} + \lambda \sum_{p_i \in OS_s} \frac{p_i - r_i}{\sum_{l=1}^{N} w_{il}} \qquad r_i = \sum_{l=1}^{N} w_{il} \cdot v_l^{(k)} \tag{2}$$

For the purpose of this paper, we have written this equation as a generalization of the original SART and SIRT equations to support any number of subsets. Here, the p_i are the pixels in the M/S acquired images that form a specific (ordered) subset OS_s where $1 \le s \le S$ and S is the number of subsets. The factor λ is the relaxation factor, which will be chosen as a function of subset size (for SIRT where S=M, λ=1). The factor k is the iteration count, where k is incremented each time all M projections have been processed. In essence, all voxels v_j on the path of a ray r_i are updated (corrected) by the difference of the projection ray r_i and the acquired pixel p_i, where this correction factor is first normalized by the sum of weights encountered by the (back-projection) ray r_i. Since a number of back-projection rays will update a given v_j, these corrections need also be normalized by the sum of (correction) weights. For SIRT, these normalization weights are trivial.

2.2 GPU-Accelerated Reconstruction: Threads and Passes

As mentioned, the NVIDIA 8800 GTX board has 128 generalized SIMD processors. Up to very recently, the only way to interface with GPU hardware was via a suitable graphics API, such as OpenGL or DirectX, and using CG [4] (or GLSL or HLSL) for coding *shader programs* to be loaded and run on the SIMD *fragment processors*. With the introduction of a new API, CUDA (Compute Unified Device Architecture) [16], the GPU can now directly be perceived as a multi-processor. With CUDA, the CG fragments become the CUDA (SIMD) *computing threads* and the shader programs become the *computing kernels*. With the CUDA specifications, much more information about the overall GPU architecture is now available, which helps programmers to fine-tune thread and memory management to optimize performance, viewing GPUs as the multi-processor architecture it really is. Reflecting this GPGPU (General Programming on GPUs) trend, new GPU platforms have now become available that do not even have graphics display capabilities, such as the NVIDIA Tesla board. Although we used CG shaders to obtain the results presented in this paper, similar symptoms also occur in CUDA where synchronization operations have to be formally called to finish executions of all threads within a thread block to resume the pipeline. After all, the underlying hardware and its architecture remain the same, just the API is different.

A number of papers [12][13] have described in great detail how projection and backprojection operations (phases 1 and 3) can be efficiently performed on the GPU. Since the subject of this paper is mainly the impact of the iterative update schedule on the management of computing threads, we shall express all operations in that context, neglecting the API used for implementation (CG or CUDA). In this work, the 3D object estimate is stored as a stack of slices (2D arrays). For projection a thread is

spawned for each target pixel, interpolating the slices according to the projective viewing transform. For backprojection a thread is spawned for each target voxel and projection image, interpolating the projection images according to the (same) projective viewing transform. The computation of the correction factors and the normalizations are simple vector operations.

The GPU memory model differentiates itself significantly from its CPU counterpart, posing greater restrictions on memory access operations in order to reduce latency and increase bandwidth. Here not only registers and local memory are reduced or even completely eliminated – the global memory also allows only read instructions during the computation. Further, the write operator can be executed only at the end of a computation, when the thread (or fragment) is released from the pipeline, to be blended with the target. Therefore, in general-purpose computing using GPUs, computations are triggered by initializing a "pass". A pass includes setting up the computation region and attaching a kernel program to simultaneously apply specific operations on every thread generated. The data is then streamed into the pipeline, where the modification can be done only at the end of the pass. Cycles and loops within a program can be implemented either inside the kernel or by running multiple passes. The former is generally faster since evoking a rendering pass and storing intermediate results in memory are costly, but there exists a register count limit in the current hardware which prevents unconstrained use of loops in the kernel.

2.3 Ordered Subsets: GPU-Accelerated Reconstruction From Projections

Having described the relevant elements of the underlying GPU hardware we are now ready to describe their impact in the context of subsets. Equation 2 above described the generalization of algebraic reconstruction into an OS configuration. What is left to define is how the subsets OS_s are composed and how λ is chosen for given number of subsets S. As specified above, OS_s is the set of projections contained in each subset, to be used in a pair of simultaneous forward projections and simultaneous backward projections. In our application, each subset has the same number of projections, that is $|OS_s|=|\mathrm{OS}|$, which is typical. Thus, the total number of projections $M = |OS|\cdot S$. The traditional way of filling a certain subset OS_s is to select projections whose indices m ($1 \le m \le M$) satisfy $m \bmod S = s$. This is what has been adopted in OS-EM. In contrast, we use a randomized approach to fill the subsets, which we find yields better results than the regular subset population approach. For this, we simply generate a projection index list in random order and sequentially divide this list into S subsets.

The relaxation factor λ to be used for an arbitrary S is chosen by linearly interpolating the optimal λ_{SART} for SART and the typical value of λ=1 used for SIRT (we have found that $\lambda_{SART} = 0.1$ works well in practice):

$$\lambda = (\lambda_{SART} - 1)\left(\frac{S-1}{N-1}\right) + 1 \qquad 1 \le S \le N \tag{3}$$

This scheme balances the smoothing effect achieved by the application of a set of simultaneous projections with that obtained by using a lower relaxation factor: the lesser projections in a subset, the lower the λ.

In the projection phase, each (pixel) thread computes its entry point, exit point, and ray direction vector (or it looks these up from a texture) and interpolates the slices in SIMD lock step. So, when the size of the subset increases (the OS_s projections), more threads are being spawned. In the backprojection, each voxel (thread) computes its mapping to all OS_s projections and interpolates its updates. Having a greater number of projections in the set makes the kernel program longer and increases its efficiency. This is in addition to the reduction of the number of context switches as OS_s increases.

3 Experiments and Results

For the following experiments we used the 2D *Barbara* test image to evaluate the performance of the different reconstruction schemes described above. We used this image, popular in the image processing literature, since it has several coherent regions with high-frequency detail, which present a well observable test for the fidelity of a given reconstruction scheme. The target 2D image is obtained by cropping the original image to an area of 256×256 pixels resolution. We obtained 180 projections at uniform angular spacing of [-90˚,+90˚] in a parallel projection viewing geometry. We also simulated a limited-angle scenario, where iterative algorithms are often employed. Here, we produced 140 projections in the interval [-70˚, +70˚]. All reconstructions used linear interpolation.

We observed that for a given number of iterations, reconstruction results from SART achieved the best score but needed the longest time to compute. As motivated above, the reason for this stems from the fact that SART requires more rendering passes on the GPU. SIRT, on the other hand, requires the fewest number of passes per iteration and thus completes it the fastest, but the speed of convergence is the slowest, that is, a greater number of iterations are needed. Again, we note that these findings are specific to GPU-based reconstructions, since here the number of passes is particularly important. A CPU-based reconstruction, in contrast, would be much less sensitive to this relationship since the cost here is dominated by the elementary operations of projection/backprojection, which are massively accelerated on GPUs.

We shall now explore if there is an optimum in terms of the number of subsets. Such an optimal subset size could then be used to generate the best reconstruction in the smallest amount of (wall-clock) time. We will use the cross-correlation coefficient (CC) as the metric to measure the degree of similarity between the original image and its reconstruction:

$$CC = \frac{\sum_j (r_j - \mu_r)(o_j - \mu_o)}{\sqrt{\sum_j (r_j - \mu_r)^2 \sum_j (o_j - \mu_o)^2}} \tag{4}$$

where j counts the number of image pixels, r_j and o_j are pixels in the reconstruction and original image, respectively, and the μ factors are their mean values. Figures 1 and 2 show the reconstruction results, for both the full and the limited angle case, for a fixed CC (comparing reconstruction with the original) which means that all reconstructed images are nearly identical to each other (in terms of statistical error).

There we observe that the smaller the number of subsets, the greater the number of iterations that are required to reach the set convergence threshold. As mentioned above, the wall clock time on the CPU is directly related to the number of iterations, so SIRT is roughly 87/6=14 times slower than SART on the CPU. However, due to the mentioned overhead involved in the GPU-based framework, different wall-clock times are produced with a GPU implementation. We measured that SART on the GPU takes nearly twice as long as SIRT and using 10 subsets (5 for the limited angle case with fewer projections) achieves the best timing performance compared to the other subset configurations.

More insight can be gained by studying the development of the CC metric both in terms of the number of iterations and the actual wall clock time. Figure 3 shows plots for both. In the top graph we observe that the smaller the number of subsets, the slower the speed of convergence in terms of CC. At the upper limit is SART which converges fastest. While this relationship has been known, the plot on the bottom is more novel. It reveals that there is a certain subset number for which the highest CC value can be maintained, consistently at all times. In the current experiment (the Barbara image reconstructed from 180 (140) evenly distributed projections), this number is 10 (5). However, this optimal number may vary in different reconstruction scenarios and applications.

Our next experiment deals with the composition of the subsets themselves. Figure 4 shows the performance of OS SIRT with three different numbers of subsets (here, 10, 20, and SART) using a regular interleaved projection selection (as it has been suggested for OS-EM) and our proposed random projection selection scheme. The results show that the random approach always performs better than the regular method, and the difference margin increases when more subsets are used, with the largest difference obtained with SART.

The tendency of SART to produce reconstructions noisier than the original and that of SIRT to produce reconstructions smoother than the original is also demonstrated in Figure 5, where we show the renditions of a line profile across another area of the Barbara image (only for the original image and reconstructions with SART and SIRT).

Finally, Figure 6 shows the results obtained with a medical dataset, reconstructed with the same framework. The projection data were obtained by a high-quality X-ray simulation from a CT dataset, labeled ‘Original’ in Figure 6. Similar outcomes are observed: a subset size of 10 achieved the best wall-clock time performance. Here we used the R-factor as the error metric, where the difference between the simulated and the scanner projections (the data) is divided by the sum of all pixels values from the scanner projections. We observe that OS SIRT with 10 subsets of 18 projections each reaches a preset R-factor=0.007 26% faster than SIRT and 86% faster than SART.

4 Conclusions

We have shown that iterative reconstruction methods used in medical imaging, such as EM or SIRT, have special properties when it comes to their acceleration on GPUs. While splitting the data used within each iterative update into a larger number of

smaller subsets has long been known to offer greater convergence and computation speed on the CPU, it is vastly slower on the GPU. This was a direct consequence of the thread fill rate in the projection and backprojection phase. Larger subsets spawn a greater number of threads, which keeps the pipelines busier and also reduces the latencies incurred by a greater number of passes and context switches. Seeking to identify the optimal subset size and number, we then generalized the popular Simultaneous Iterative Reconstruction Technique (SIRT) to OS SIRT. This generalization allows researchers to optimize the wall clock time required for a GPU-accelerated reconstruction, enabling high-quality reconstructions to be obtained faster, taking full account of the particularities associated with the GPU architecture and programming model. This OS SIRT can optimize the reconstruction performance by choosing the optimal number of subsets into which the projections are distributed (in random order). Here, it is likely that this optimal number of subsets will vary depending on the domain application and the general reconstruction scenario. Thus, in order to identify the optimal subset number for a new application setting, to be used later for repeated reconstructions within this application setting, one may simply run a series of experiments with different numbers of subsets and use the setting with the shortest wall clock time required for the desired reconstruction quality. In fact, such strategies are typical for GPU-accelerated general-purpose computing applications (GPGPU). For example, the GPU bench was designed to run a vast benchmark suite [17] to determine the capabilities of the tested hardware. Our findings with SIRT readily extend to EM since the two methods have, as far as the computations are concerned, similar operations and overhead. In future work, we plan to devise a more sophisticated λ schedule, which we believe will positively affect the reconstruction performance. It will most probably shift the optimal subset size to some degree, but it will not negate the general finding of this paper, which is that larger subsets are to be favored.

Although we have used CG shaders to obtain the results presented in this paper, similar symptoms also occur in CUDA where synchronization operations have to be formally called to finish executions of all threads within a thread block to resume the pipeline. In general, the underlying hardware, its architecture, and the overall thread management remain the same – just the API is different, enabling a tighter control over the threads. In future work, we plan to study the reported effects to a more detailed extent in CUDA, to determine if a shift in the performance-optimal subset configuration occurs. In fact, this is likely to happen for every new generation of the hardware. The main goal of this paper was not to recommend a specific optimal number of subsets, but to raise awareness to these interesting platform-related phenomena.

References

[1] A. Andersen, "Algebraic reconstruction in CT from limited views," *IEEE Trans. on Medical Imaging,* 8:50-55, 1989.

[2] A. Andersen, A. Kak, "Simultaneous Algebraic Reconstruction Technique (SART): a superior implementation of the ART algorithm," *Ultrasonic Imaging*, 6:81-94, 1984.

[3] Y. Censor, "On block-iterative maximization," *J. Information and Optimization Science,* 8:275-291, 1987.
[4] R. Fernando, M. Kilgard, *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics,* Addison-Wesley Professional, 2003.
[5] P. Gilbert, "Iterative methods for the 3D reconstruction of an object from projections," *J. Theoretical Biology,* 76:105-117, 1972.
[6] R. Gordon, R. Bender, G. Herman, "Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography," *J. Theoretical Biology,* 29:471-481, 1970.
[7] H. Hudson, R. Larkin, "Accelerated Image Reconstruction Using Ordered Subsets of Projection Data," *IEEE Trans. Medical Imaging,* 13:601-609, 1994.
[8] J. Kole, F. Beekman, "Evaluation of accelerated iterative x-ray CT image reconstruction using floating point graphics hardware", *Physics in Medicine and Biology*, 51:875-889, 2006.
[9] J. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kruger, A.E. Lefohn, and T. Purcel, "A Survey of General-Purpose Computation on Graphics Hardware," *Eurographics 2005*, pp. 21-51, 2005.
[10] T. Schiwietz, T. Chang, P. Speier and R. Westermann, "MR image reconstruction using the GPU," *Proc. SPIE 6142*, pp. 1279–90, 2006.
[11] L. Shepp, Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *IEEE Trans. on Medical Imaging,* 1:113-122, 1982.
[12] F. Xu, K. Mueller, "Real-Time 3D Computed Tomographic Reconstruction Using Commodity Graphics Hardware," *Physics in Medicine and Biology*, 52:3405–3419, 2007.
[13] F. Xu, K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," *IEEE Trans. on Nuclear Science*, 52:654-663, 2005.
[14] X. Xue, A. Cheryauka, D. Tubbs, "Acceleration of fluoro-CT reconstruction for a mobile C-Arm on GPU and FPGA hardware: a simulation study," *Proc. SPIE 6142*, pp. 1494–501, 2006.
[15] http://www.gpgpu.org
[16] http://www.nvidia.com/object/cuda_develop.html
[17] http://graphics.stanford.edu/projects/gpubench

		
Original	SIRT	OS SIRT 10
# iterations	87	47
time (s)	5.89	5.17
		
OS SIRT 20	OS SIRT 60	SART
32	15	6
11.28	9.9	10.28

Fig. 1. Reconstructions obtained with various subset sizes for a fixed CC=0.95 and 180 projections in an angular range of 180˚. We observe that OS SIRT with 10 subsets of 18 projections each reaches this fixed CC value 12% faster than SIRT and 50% faster than SART.

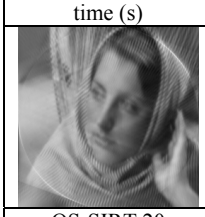
		
Original	SIRT	OS-SIRT 5
# iterations	75	56
time (s)	2.46	2.43
		
OS-SIRT 20	OS-SIRT 70	SART
30	15	12
8.96	4.53	15.84

Fig. 2. Reconstructions obtained with various subset sizes for a fixed CC=0.93 and 140 projections in an angular range of 140˚. We observe that OS SIRT with 5 subsets of 28 projections each reaches this set CC value 2% faster than SIRT and 84% faster than SART.

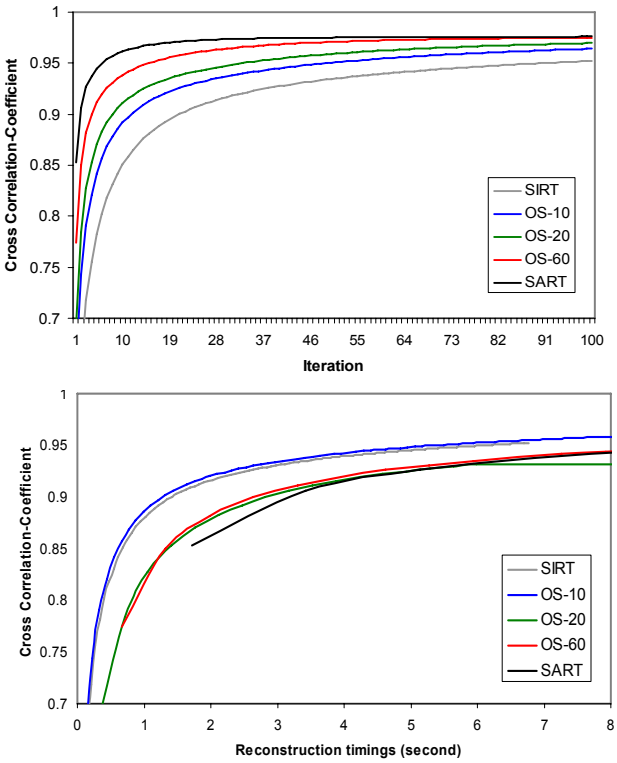


Fig. 3. (top) CC vs. number of iterations; (bottom): CC vs. wall clock time. We observe that while SART achieves the best results with the smallest number of iterations, OS SIRT achieves it in the smallest amount of time, within a GPU-accelerated framework.

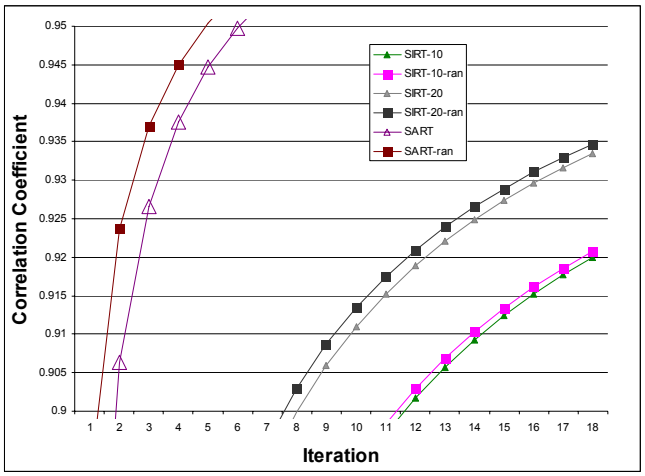


Fig. 4. CC values for regular OS SIRT and randomized OS SIRT.

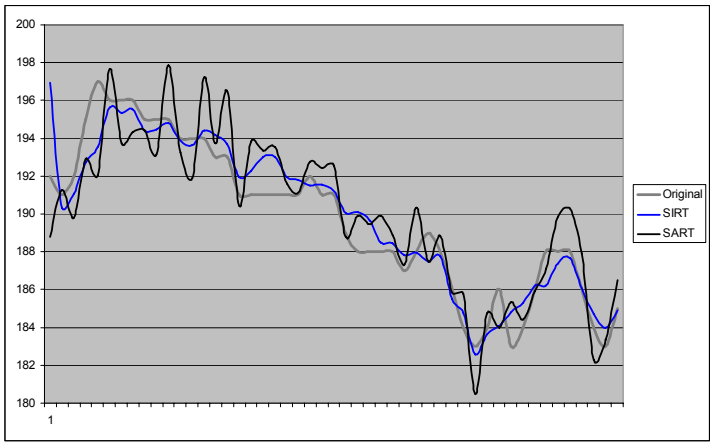
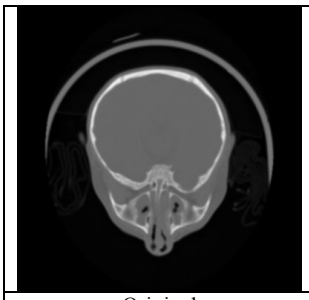


Fig. 5. Line profiles of the image background for SART, SIRT, and the original Barbara image.



Original	SIRT	OS-SIRT 3
# iterations / time (s)	100 / 18.86	85 / 17.69

OS-SIRT 10	OS-SIRT 60	SART
54 / 13.99	16 / 79.9	6 / 102.79

Fig. 6. Reconstructed baby head using high-quality simulated projection data of the volume labeled 'Original'. Results were obtained with various subset sizes for a fixed R-factor = 0.007 and 180 projections in an angular range of 180˚. OS SIRT with 10 subsets of 18 projections each reaches this set R-factor value 26% faster than SIRT and 86% faster than SART.