Title

Parallelization of Iterative Reconstruction Algorithms in Multiple Modalities

Permalink

https://escholarship.org/uc/item/1z61d77c

Authors

Mitra, Debasis
Pan, Hui
Alhassen, Fares
et al.

Publication Date

2014-11-01

DOI

10.1109/nssmic.2014.7430944

Published in final edited form as:

IEEE Nucl Sci Symp Conf Rec (1997). 2014 November ; 2014: .

Parallelization of Iterative Reconstruction Algorithms in Multiple Modalities

Debasis Mitra [Senior Member, IEEE],
Computer Science Department, Florida Institute of Technology, 150 West Univ. Blvd., Melbourne, FL, 32901

Hui Pan,
Computer Science Department, Florida Institute of Technology, 150 West Univ. Blvd., Melbourne, FL, 32901

Fares Alhassen, and
UCSF Physics Research Laboratory, Department of Radiology and Biomedical Imaging, University of California, San Francisco

Youngho Seo [Senior Member, IEEE]
UCSF Physics Research Laboratory, Department of Radiology and Biomedical Imaging, University of California, San Francisco

Hui Pan: hpan2010@my.fit.edu

Abstract

In this work we have parallelized the Maximum Likelihood Expectation-Maximization (MLEM) and Ordered Subset Expectation Maximization (OSEM) algorithms for improving efficiency of reconstructions of multiple pinholes SPECT, and cone-bean CT data. We implemented the parallelized versions of the algorithms on a General Purpose Graphic Processing Unit (GPGPU): 448 cores of a NVIDIA Tesla M2070 GPU with 6GB RAM per thread of computing. We compared their run times against those from the corresponding CPU implementations running on 8 cores CPU of an AMD Opteron 6128 with 32 GB RAM. We have further shown how an optimization of thread balancing can accelerate the speed of the GPU implementation.

I. Algorithms

We developed three versions of the reconstruction algorithms: (1) MLEM for pinhole-SPECT and cone beam CT (CBCT), both implemented in CUDA C++ with the computation of system-matrix embedded [1]; (2) MLEM for parallel-hole SPECT with computation of point spread function embedded (implemented in CUDA C++) [2]; and (3) an OSEM version for parallel-hole SPECT. For each version we have a correspondingly similar CPU implementation for comparing the results. A sketch of the pinhole-SPECT MLEM GPU implementation is given in Fig. 1 as a sample of different types of the algorithms.

Steps 1, 2, 5, 7 and 9 are parallelized for computation on GPU cores with each thread as a CUDA kernel function. For parallel implementation, we need to set the dimension of blocks and the number of threads in each block. Usually, the coronal (y) and transverse (z) dimensions of the input sinogram (or reconstruction image) are set to the *block size* and the sagittal (x) dimension is set to the *thread size*, thus, each thread represents one pixel (or voxel). In the kernel function, we get thread index and block index to identify the index of a pixel (or a voxel). In the OSEM implementation step 4 runs over the subsets of the projections, and an additional sequential loop within it runs over the projections in each subset.

II. Results

Our single pinhole-SPECT dataset has 60 projections, each of size 256^2 pixels, and the four pinhole SPECT data set has 60 projections each with 128^2 pixels, both are from simulated mouse heart (MOBY phantom for 64^3 voxels). The reconstructions are done with 200 MLEM iterations in each case. A simulated hot rod phantom is used to generate a sinogram with 600 angular projections, each with 128^2 matrix, for the CBCT reconstruction over 10 MLEM iterations. The parallel-hole SPECT data is from an ^{111}In-capromab pendetide patient scan over a full body torso with a SPECT/CT camera where a medium-energy general purpose (MEGP) collimator was used. The sinogram comprises of 120 angular projections with 128^2 pixels on each projection. Both MLEM and OSEM reconstructions are performed for a volume of 128^3 matrix size. Each algorithm was run over iterations 10, 20 and 40.

Slices shown in Figs. 2–5 are from the 1 pinhole, 4 pinholes, CBCT, and parallel-hole SPECT input sinograms (top left) and their corresponding CPU and GPU reconstructed images (bottom left, first and second images respectively). Line profiles on CPU and GPU generated images show that they are very similar (subject to a constant multiplicand).

In order to compare the images we have computed the *root-mean square* and *normalized mean square* differences between the resulting images that are not shown here for the lack of space.

We have recorded the detailed timing information from the parallel-hole MLEM CPU, parallel-hole MLEM GPU and parallel-hole OSEM GPU implementations. The overall reconstruction times are shown in Table I.

III. Grid, Block and Threads Reorganization for Improvement in GPU implementation

We used a GPU system that is capable of handling up to 1024 threads per block. More the number of threads used per block the better the GPU performance will be. In our conventional algorithm, the y, z dimensions of the reconstruction image are mapped to the block size and the x dimension is mapped to the thread size. Fig. 6(a) shows the structure of that GPU organization. To accelerate the algorithm further, we redesigned the organization in such a way that the maximum available threads are fully utilized. In this adapted implementation we split the y dimension into two parts. The first part is put in the maximum-thread size 1024 and the second multiplicand part is assigned the block. Fig. 6(b) shows our new thread-balancing structure.

We implemented this improvement within our GPU MLEM implementation for pinhole SPECT, and compared their reconstruction times. The result is shown in Table II.

IV. Comparing Multiple Pinhole Data

We created a 4-pinhole dataset of size 128^2×60 for comparing the time complexity with one pinhole dataset. It is expected that the algorithms would need more time to reconstruct the 4-pinhole dataset than to reconstruct 1-pinhole dataset. Fig. 7 shows the input projection of our 128^2×60 4-pinhole dataset and the reconstructed images for both CPU and GPU implementations. Fig. 8 is the result of time comparison between CPU, GPU and GPU with the new reorganized structure for 1- and 4-pinhole datasets.

V. GPU Performance on Different Data Sizes

In order to measure the GPU performance on different sinogram sizes, we created four simulation datasets with the same objects but different dimensions. First, using MATLAB we have generated 3D volume data sets or binary matrices (1 or 0 values) with different dimensions. The dimensions are 16×16×16, 32×32×32, 64×64×64 and 128×128×128. In order to produce a sinogram for each volume, we generated SPECT system matrices corresponding to each of these volumes (with some standard acquisition parameters). The acquisition is assumed to go over 60 projections over 360 degrees, with detector head sizes corresponding to respective volumes (e.g., for 64×64×64 voxels volume, detector head is of 64×64 pixels) and with the low-energy high-resolution (LEHR) parallel-hole collimator. Finally, Poisson noise was added to each original sinogram. Figs. 9 and 10 show the volumes and sinograms of these datasets, respectively. Fig. 11 is the reconstructed images for these datasets.

After we compared the timings between our CPU and GPU implementations, we found that even though the performance of the GPU reconstruction was much better as expected, it still has rooms for further improvement. As in the cases of multiple pinholes and single-pinhole experiments, we reorganized the GPU memory for improving the performance. We followed the basic strategy which we discussed before (in Section III above). Fig. 12 shows the reconstructed images between CPU, GPU and GPU implementations with the new reorganized structure. Fig. 13 is the time performance between CPU, GPU and GPU implementations on different data sizes. From these results, the implication is that both GPU and GPU reconstructions with the new structure have better performance than the CPU implementation. Meanwhile, with the increased size of system matrix, the efficiency of GPU with the new structure improved significantly over the other implementations with the old structure. Table III provides the details of time performance for each implementation.

When the size of system matrix is over a threshold (larger than the 64^3×64^2 ×120), both GPU and GPU reconstructions with the new structure will save more time and the ratio of improvement (the last column of Table III) will increase as well. In general, the GPU reconstruction with the new structure has the better performance than the others we have included in our comparison study. Our results clearly show that the performance of a GPU reconstruction algorithm is associated with the organization of memories. For example, optimizing the organization can accelerate the forward projection, the backprojection, and the system matrix generation.

VI. Future Work

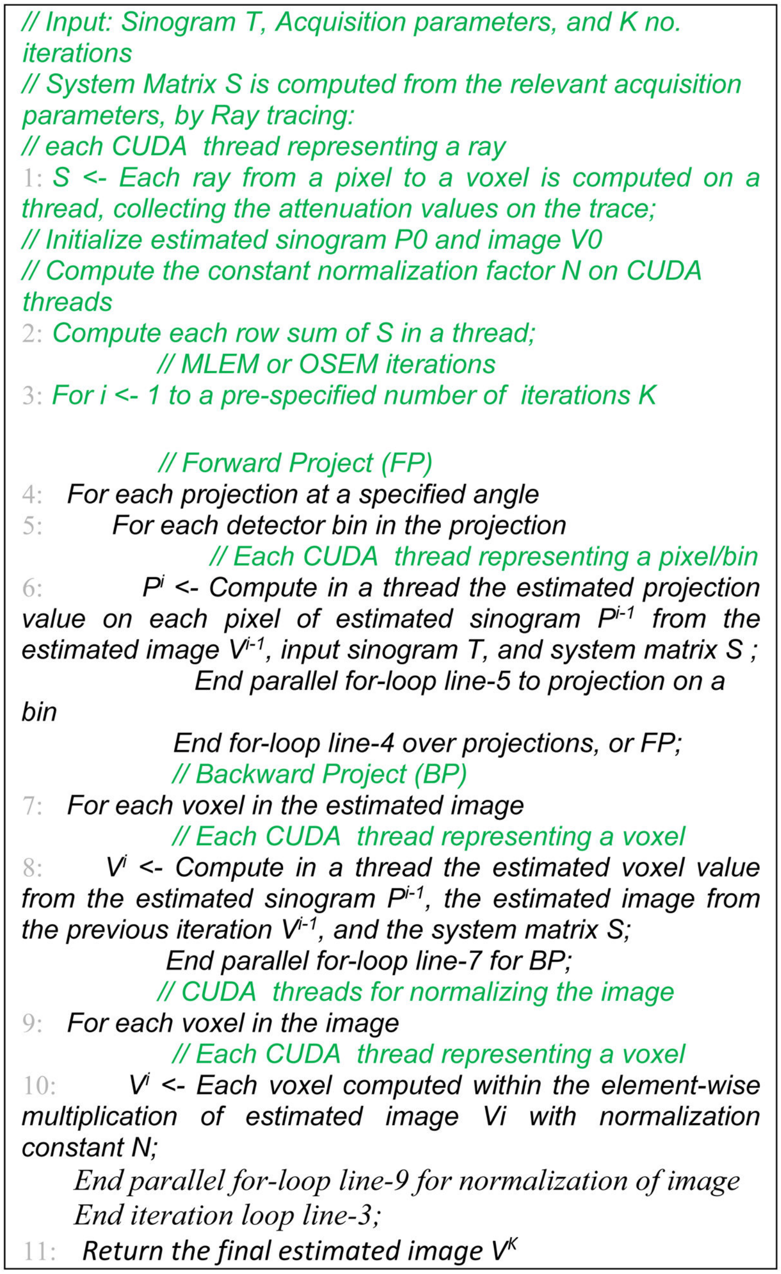
We have observed similar performance improvements with GPU implementation of MLEM reconstruction for CBCT (cone beam computed tomography) data, over the corresponding CPU implementation. In the near future we plan to perform more experiments with different dimensions of input sinogram data for measuring the GPU reconstruction efficiency. We also plan to automate optimal thread reorganization for varying sizes of data. The optimization process could include utilization of more than one GPU units using a higher level parallelization as in [3].

Acknowledgments

This work is supported in part by National Institute of Biomedical Imaging and Bioengineering under grant #R01 EB012965.

References

1. Alhassen F, et al. Ultrafast Multipinhole Single Photon Emission Computed Tomography Iterative Reconstruction Using CUDA. IEEE Nuclear Science Symposium Conference Record. 2011
2. Xia W, Lewitt RM, Edholm PL. Fourier Correction for Spatially Variant Collimator Blurring in SPECT. IEEE Trans Med Imaging. 1995; 14
3. Pratx G, Surti S, Levin C. Fast List-Mode Reconstruction for Time-of-Flight PET Using Graphics Hardware. IEEE Trans Med Imaging. 2011; 58(1)
4. Owens JD, Houston M, Luebke D, Green S, Stone JE, Phillips JC. GPU Computing. IEEE Trans Med Imaging. 2008; 96(5)
5. Liu, H.; Ma, T.; Chen, S.; Liu, Y.; Wang, S.; Jin, Y. Development of GPU based image reconstruction method for clinical SPECT. Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC); IEEE; 2012.



```
// Input: Sinogram T, Acquisition parameters, and K no. iterations
// System Matrix S is computed from the relevant acquisition parameters, by Ray tracing:
// each CUDA thread representing a ray
1: S <- Each ray from a pixel to a voxel is computed on a thread, collecting the attenuation values on the trace;
// Initialize estimated sinogram P0 and image V0
// Compute the constant normalization factor N on CUDA threads
2: Compute each row sum of S in a thread;
        // MLEM or OSEM iterations
3: For i <- 1 to a pre-specified number of iterations K

        // Forward Project (FP)
4:   For each projection at a specified angle
5:       For each detector bin in the projection
            // Each CUDA thread representing a pixel/bin
6:           $P^i$ <- Compute in a thread the estimated projection value on each pixel of estimated sinogram $P^{i-1}$ from the estimated image $V^{i-1}$, input sinogram T, and system matrix S ;
            End parallel for-loop line-5 to projection on a bin
         End for-loop line-4 over projections, or FP;
         // Backward Project (BP)
7:   For each voxel in the estimated image
         // Each CUDA thread representing a voxel
8:       $V^i$ <- Compute in a thread the estimated voxel value from the estimated sinogram $P^{i-1}$, the estimated image from the previous iteration $V^{i-1}$, and the system matrix S;
         End parallel for-loop line-7 for BP;
        // CUDA threads for normalizing the image
9:   For each voxel in the image
         // Each CUDA thread representing a voxel
10:      $V^i$ <- Each voxel computed within the element-wise multiplication of estimated image Vi with normalization constant N;
     End parallel for-loop line-9 for normalization of image
     End iteration loop line-3;
11:  Return the final estimated image $V^K$
```

Fig. 1.

Pinhole MLEM GPU implementation steps

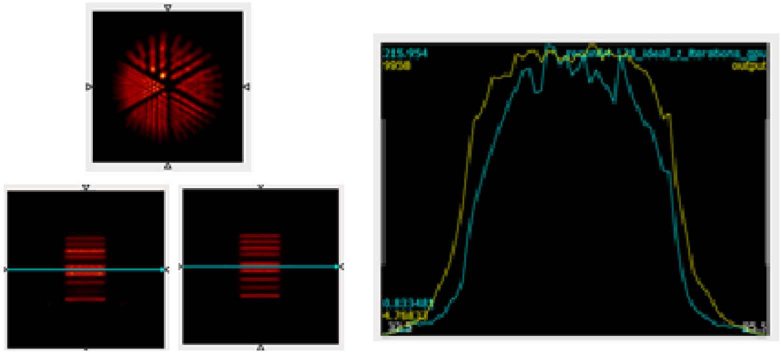


Fig. 2.

Top left is a projection of the single pinhole SPECT sinogram, and the bottom images are slices from our CPU and GPU based reconstructions respectively. Plots on the right are scaled values of the voxels along the lines on the image slices on left bottom.

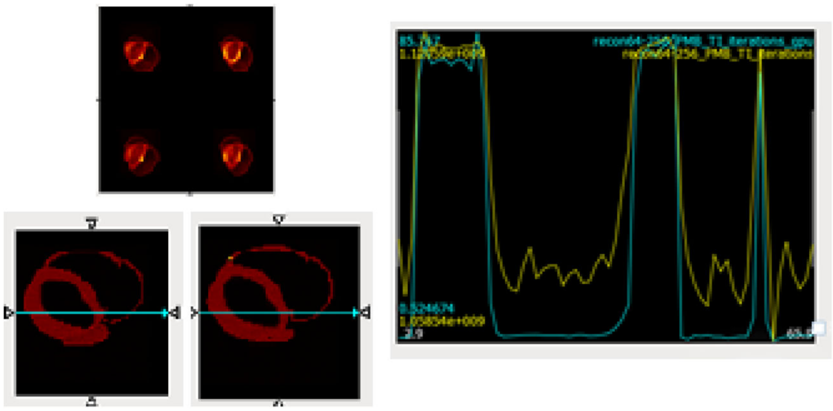

recon64-256_PMB_T1_iterations_gpu
recon64-256_PMB_T1_iterations
0.524674
1.05854e+009

Fig. 3.

Top left is a projection of the four pinholes SPECT sinogram, and the bottom images are slices from CPU and GPU based reconstructions respectively. Plot on the right are scaled values of the voxels along the lines on the image slices on left bottom.

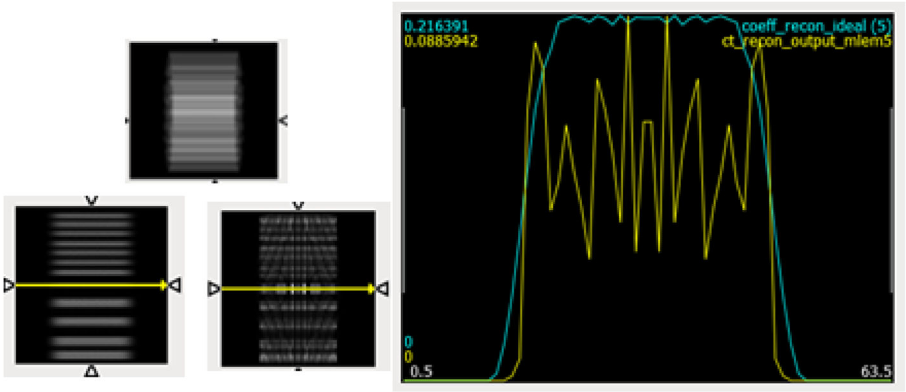


Fig.4.

Top left is a projection of the CBCT sinogram, and the bottom images are slices from CPU and GPU based reconstructions respectively. Plot on the right are scaled values of the voxels along the lines on the image slices on left bottom.

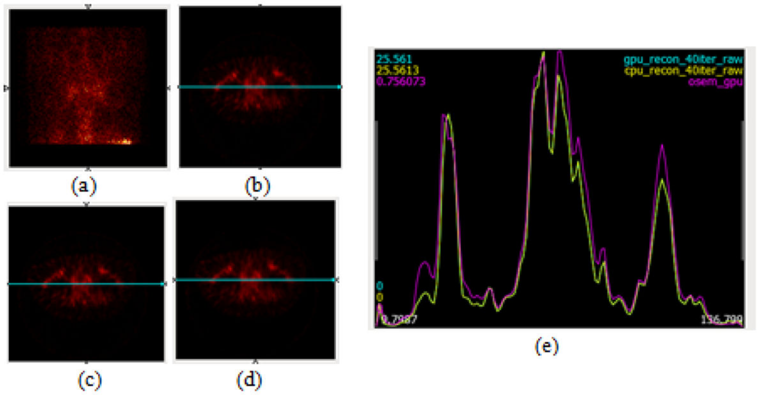


Fig. 5.

(a) A projection of the parallel-hole SPECT whole-body data. (b) A slice from the CPU MLEM based reconstruction. (c) A slice from the GPU MLEM reconstruction. (d) GPU OSEM reconstruction-slice. (e) Scaled plots of the voxels values along the lines on (b)–(d).

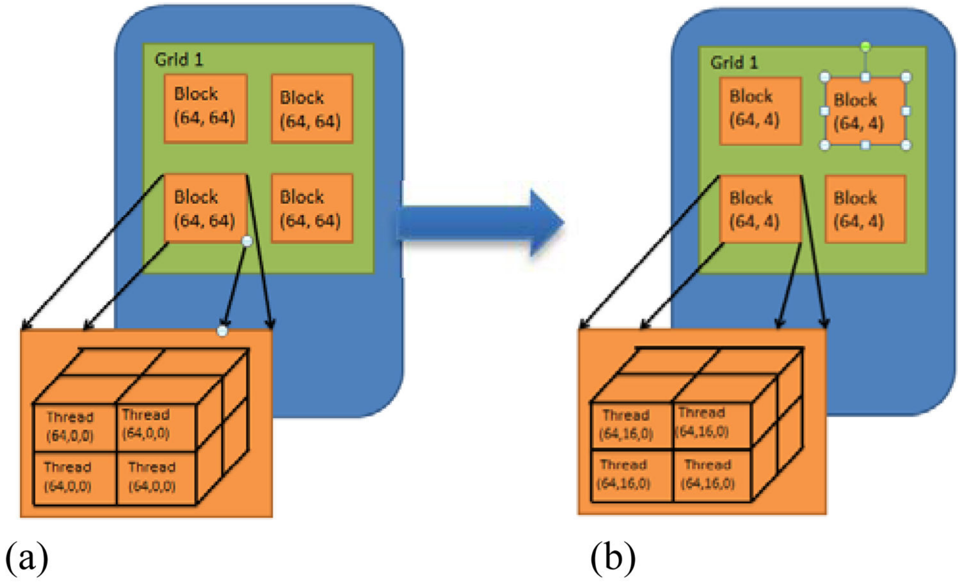


Fig. 6.

(a) Conventional threading structure for the image dimension 64×64×64. (b) Modified structure for the same image size.

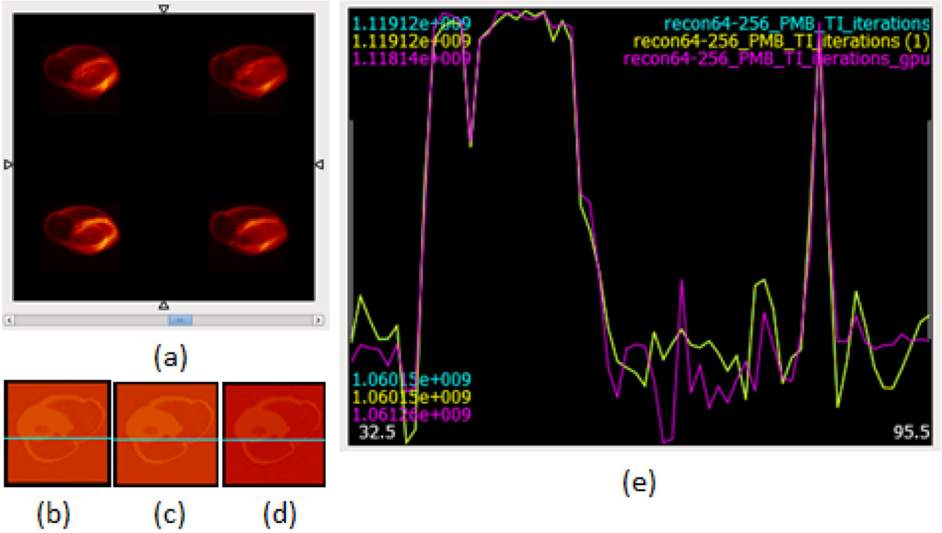

1.11912e+009
1.11912e+009
1.11814e+009
recon64-256_PMB_TI_iterations
recon64-256_PMB_TI_iterations (1)
recon64-256_PMB_TI_iterations_gpu
1.06015e+009
1.06015e+009
1.06126e+009
32.5
95.5
(a)
(b)
(c)
(d)
(e)

Fig. 7.

(a) A projection from the 4-pinhole sinogram with dimension $128^2 \times 60$. (b)–(d) are slices from CPU, GPU and GPU with new structure based reconstructions respectively. Plots on the right are scaled values of the voxels along the line profile on the image slices on left bottom.

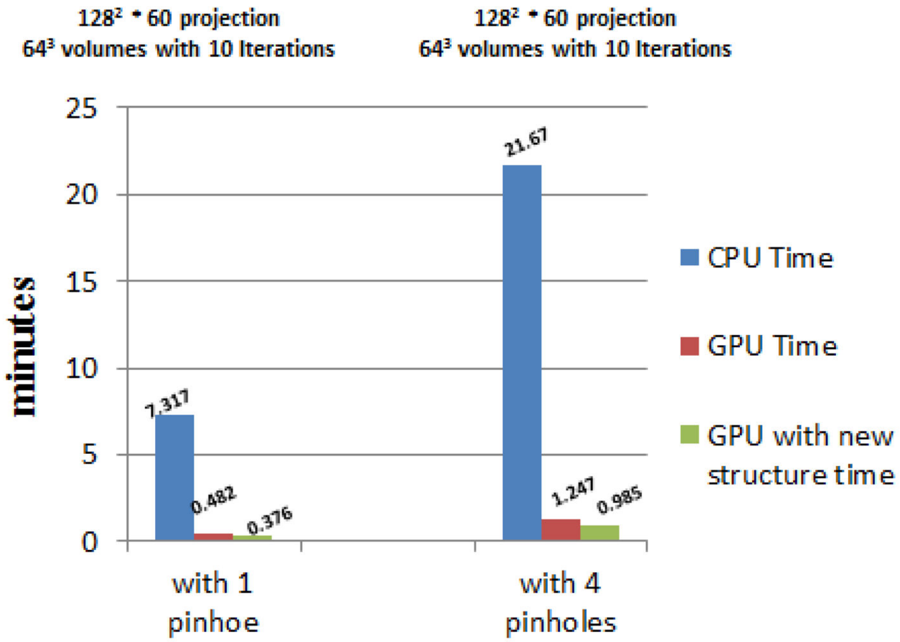


Fig. 8.

Time comparison between CPU, GPU and GPU reconstructions with the new reorganized structure for 1- and 4-pinhole datasets.

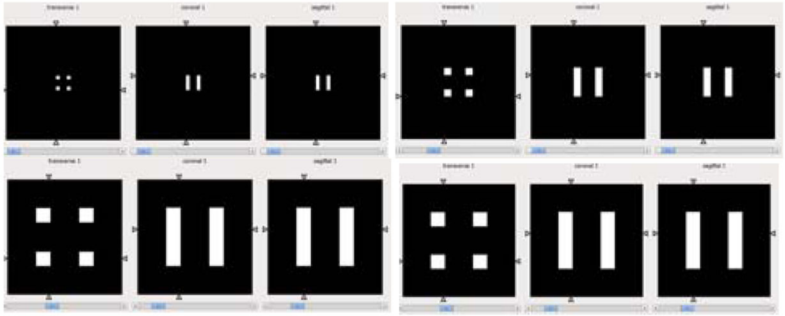


Fig. 9.

Top images are $16^3$and 32^3volumes respectively, Bottom images are $64^3$and 128^3volumes respectively

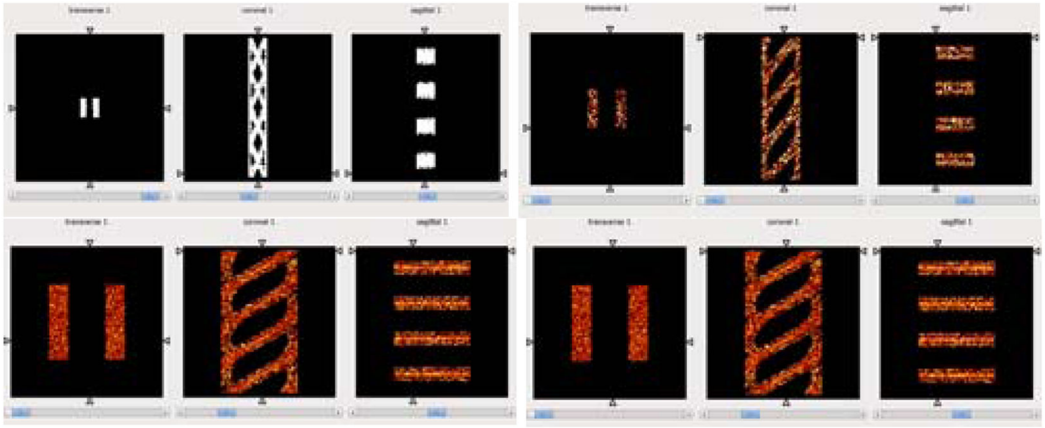


Fig. 10.

Top images are 16^2×120 and 32^2×120 sinograms respectively. Bottom images are 64^2×120 and 128^2×120 sinograms respectively.

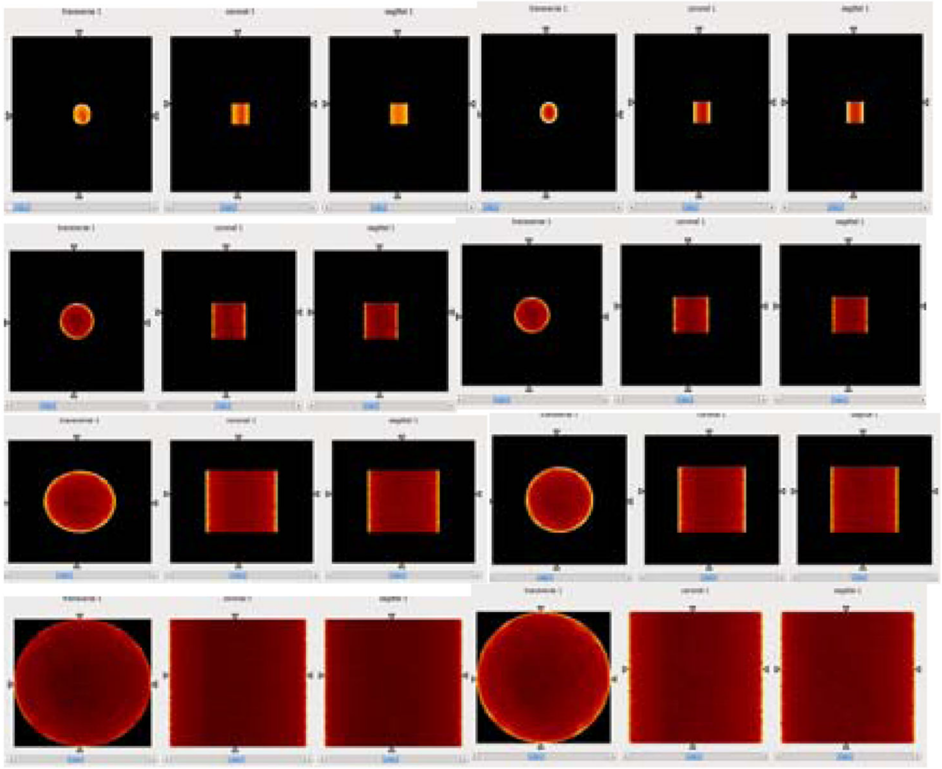


Fig. 11.

Reconstructed images (three columns on the left are produced by GPU, three on right are by CPU). From top to bottom rows, the volume sizes are 16^3, 32^3, 64^3 and 128^3 respectively.

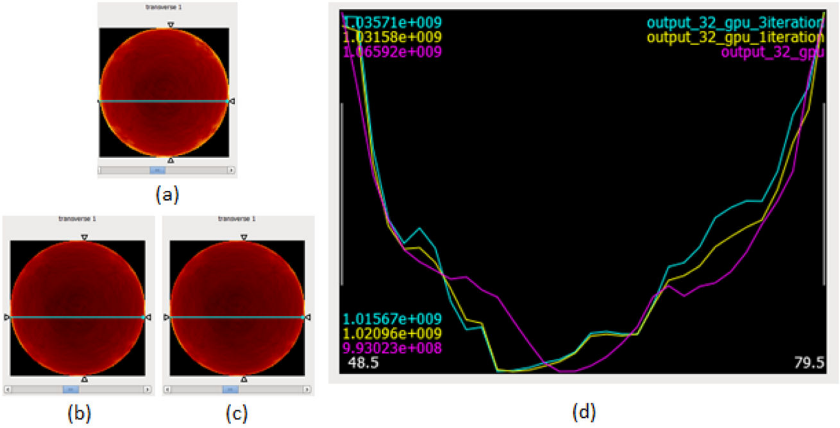


Fig. 12.

(a)–(c) are slices from CPU, GPU and GPU with new structure based reconstructions respectively. Plot on the right are scaled values of the voxels along the lines on the image slices on left bottom.

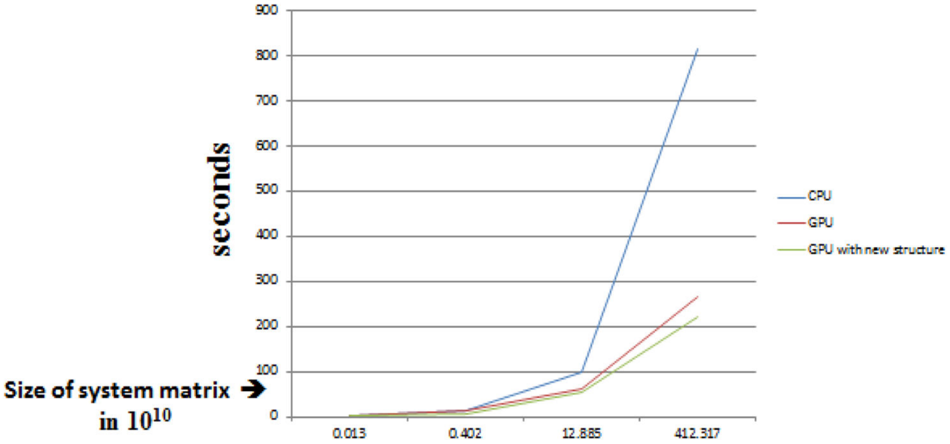


Fig. 13.

Time Performance between CPU, GPU and GPU with new structure on different data sizes

Table I

Timing comparison between Parallel-hole CPU, Parallel-hole MLEM GPU and Parallel-hole OSEM GPU, over a sinogram of 128^2 × 120 parallel-hole SPECT data as input

CPU MLEM 30 iterations (min)	GPU MLEM 30 iterations (min)	CPU OSEM 15 subsets and 2 iterations (min)	GPU OSEM 15 subsets and 2 iterations (min)
755.69	129.78	28.40	8.88

Table II

Timing comparison for Pinhole-SPECT GPU algorithm with thread reorganization

Input: 256^2×60 SPECT data	**Reconstruction time on CPU (sec)**	**Reconstruction time on GPU with old structure (sec)**	**Reconstruction time on GPU with new structure (sec)**
with 1 pinhole	1188.41	60.17	51.79
with 4 pinholes	49.51	4.01	3.26

Table III

Time comparison between CPU, GPU and GPU reconstructions with the new reorganized structure

Data Size (measured by system matrix size)	CPU(s)	GPU(s)	GPU with new structure (s)	Ratio (CPU vs GPU)	Ratio (CPU vs GPU with new structure)
16^3×16^2×120	2.377	2.902	0.863	0.819	2.754
32^3×32^2×120	12.934	11.499	6.936	1.125	1.864
64^3×64^2×120	98.703	61.879	52.842	1.595	1.867
128^3×128^2×120	815.927	265.284	219.690	3.076	3.714