Combined algorithmic and GPU acceleration for ultra-fast circular conebeam backprojection

Jeffrey Brokish[a], Paul Sack[a], Yoram Bresler[ab]
[a]InstaRecon, Inc., Champaign, IL USA 61820
[b]Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign

ABSTRACT

In this paper, we describe the first implementation and performance of a fast $O(N^3 \log N)$ hierarchical backprojection algorithm for cone beam CT with a circular trajectory[1],developed on a modern Graphics Processing Unit (GPU). The resulting tomographic backprojection system for 3D cone beam geometry combines speedup through algorithmic improvements provided by the hierarchical backprojection algorithm with speedup from a massively parallel hardware accelerator. For data parameters typical in diagnostic CT and using a mid-range GPU card, we report reconstruction speeds of up to 360 frames per second, and relative speedup of almost 6x compared to conventional backprojection on the same hardware.

The significance of these results is twofold. First, they demonstrate that the reduction in operation counts demonstrated previously for the FHBP algorithm can be translated to a comparable run-time improvement in a massively parallel hardware implementation, while preserving stringent diagnostic image quality. Second, the dramatic speedup and throughput numbers achieved indicate the feasibility of systems based on this technology, which achieve real-time 3D reconstruction for state-of-the art diagnostic CT scanners with small footprint, high-reliability, and affordable cost.

Keywords: Tomography, Circular Cone-Beam, Fast Backprojection, Hierarchical Algorithm

1. INTRODUCTION

Filtered backprojection (FBP) is the typical reconstruction algorithm used in CT reconstruction for circular cone-beam (CCB) geometries. The backprojection step of the algorithm is the computational bottleneck, giving FBP algorithms a complexity of $O(N^4)$ for 3D cone beam reconstruction of an $N \times N \times N$ volume . The backprojection equationalso has a very simple form, where each pixel in the image is determined by an accumulation of contributions from each filtered projection in the acquired scan data. FBP therefore falls into the class of "embarrassingly parallel" algorithms, where the work can be easily divided among many execution units. This results in significant speedups on parallel hardware.

Fast hierarchical backprojection[1,2,3] (FHBP) reduces the complexity of the backprojection operation to $O(N^3 \log N)$, offering the potential for tremendous speedups. Implementing FHBP on specialized hardware platforms is particularly attractive in that the inherent reduction of computation of FHBP can be combined with the speedup potential of a parallel platform to deliver unprecedented reconstruction rates. The only previous implementation of a fast hierarchical backprojection on massively parallel hardware was reported by Brokish and Bresler[4] for the 2D fanbeam geometry In this paper, we describe the first implementation and performance of a fast $O(N^3 \log N)$ hierarchical backprojection algorithm for circular cone beam CT, developed on a modern Graphics Processing Unit (GPU). For data parameters typical in diagnostic CT and using a mid-range GPU card, we report reconstruction speeds of up to 360 frames per second, and relative speedup of almost 6x compared to conventional backprojection on the same hardware.

The significance of these results is twofold. First, they demonstrate that the reduction in operation counts demonstrated previously for the FHBP CCB algorithm can be translated to a comparable run-time improvement in a massively parallel hardware implementation, while preserving stringent diagnostic image quality. Second, the dramatic speedup and throughput numbers achieved indicate the feasibility of systems based on this technology, which achieve real-time 3D reconstruction for state-of-the art diagnostic CT scanners with small footprint, high-reliability, and affordable cost.

2. BACKPROJECTION ALGORITHMS

The circular cone beam geometry considered uses a curved detector panel. For simplicity, the detector panel is logically translated to the center of rotation. The source orbits on a circular trajectory in the x-y plane at a distance D from the center of rotation. Projection data is sampled at P evenly spaced positions along this trajectory. The reconstructed volume has cross-sections with N x N voxels. Based on sampling conditions[2,5] for circular cone-beam CT, for good image quality the number of projections P should be O(N). The projection data $g[u, v, p]$ is indexed by detector channel u, row v, and source position p.

2.1 Conventional FBP

The conventional FBP reconstruction is the FDK algorithm. It is divided into two parts. In the first step, the acquired projection data $g[u, v, p]$ is filtered according to

$$g_f[u,v,p] = g[u,v,p]W_1(u,v) \otimes_u h[u], \tag{1}$$

where $W_1(u, v)$ is the cone-beam path length correction weighting, $\otimes_u$ indicates 1-D convolution along the u dimension, and $h[u]$ is the ramp filter. The convolution can be efficiently implemented in the frequency domain using the FFT. The filtering step has a complexity of O(N^3logN).

In the second step, the reconstructed image volume is formed via the backprojection equation. The image at point $\underline{x}$ in is determined by

$$f(\underline{x}) = \sum_p g_f[u(\underline{x}, p), v(\underline{x}, p), p]W_2(\underline{x}, p) \tag{2}$$

The functions $u(\underline{x}, p)$ and $v(\underline{x}, p)$ determine the location of the projection of point $\underline{x}$ onto the detector plane for source location p. $W_2(\underline{x}, p)$ is the inverse distance weighting function. Since $\underline{x}$ is generally projected to a point between discrete detector elements, it is typical to use bilinear interpolation of the filtered data. This equation illustrates how each voxel in f is determined by a contribution from each view. This yields the overall complexity of O(N^4) for a conventional backprojection operation for an N x N x N volume.

2.2 Hierarchical FBP

Since the backprojection step has greater computational complexity than the filtering step, it is the bottleneck of the FBP reconstruction algorithm. The aim of the hierarchical FBP algorithm is to reduce the complexity of this step. It uses the same filtering step as in (1), but replaces (2) with an accelerated version.

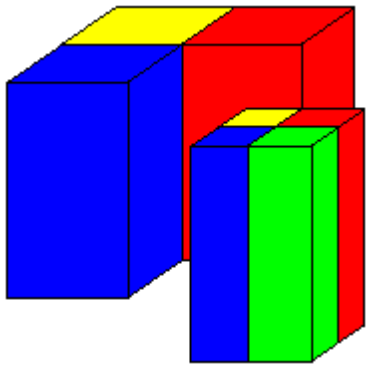

Figure 1. Pillar-Style hierarchical decomposition

The hierarchical FBP algorithm is based on two main concepts. The first concept is “divide and conquer,” in which the reconstructed volume is successively divided into smaller non-overlapping volumes, and the backprojection operation is decomposed into backprojection onto these subvolumes. For CCB with small to moderate cone angles, a pillar-style

decomposition is used, as shown in Figure 1. By itself, this decomposition scheme does not yield any reduction in operation counts.

The second concept invokes the sampling conditions for CCB reconstruction[2,5], where the number of projections needed to accurately reconstruct a bandlimited subvolume at the center of the source of rotation is proportional to the size of the subvolume. For the reconstruction of a half-sized subvolume, the projection data set can be angularly decimated by a factor of two, from P to $P/2$ projections, without any loss in reconstructed image quality.

This property is extended to apply to subpillars located at any position in the image. First, projections of the subpillars are "centered" by shifting the projection data in the u dimension, based on the projection of the subpillar's center onto the detector panel. For a particular subpillar center $\underline{x}_c$, this is expressed as

$$g_s[u,v,p] = g_f[u - u(\underline{x}_c, p), v, p]. \tag{3}$$

This shift reduces the angular bandwidth of the subpillar's projection data, allowing a sampling argument to be invoked[2]. The centered projections may then be decimated in source position p without information loss. The decimated projections are then "de-centered" by shifting the data in u back to their correct positions. The subpillar can then be backprojected with half the computational effort.

Applying the shifting and decimation procedure to each subpillar in the volume reduces the computational requirements of the backprojection of the full volume by half. Applying this process recursively (at most $\log_2 N$ times) to each subpillar as shown in Figure 1, reduces the number of views by a factor of two at each stage in the decomposition. The final backprojection stage consists of P/N projections, reducing the computational complexity of just the backprojection step to $O(N^3)$. The application of the shifting and decimation procedure to each subpillar in the volume is $O(PN^2)$. Applying the decomposition $\log_2 N$ times yields a total complexity of the hierarchical algorithm of $O(N^3 \log_2 N)$.

3. GPU COMPUTATION PLATFORM

Graphics processing units have seen steady increases in computational capabilities over the years, driven by the computer-gaming market. Early efforts at general purpose programming on the GPU were difficult due to the graphics pipeline paradigm that the code had to adhere to. With the introduction of the CUDA development tools (and more recently OpenCL), GPUs have become viable platforms for general purpose computation on GPUs (GPGPU).

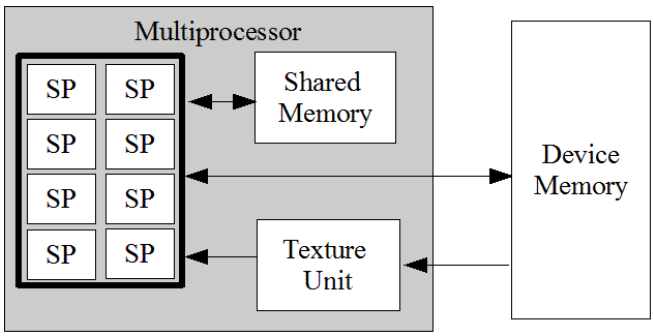

Figure 2. GPU hardware configuration

As shown in Figure 2, the primary processing element of an Nvidia GPU is organized as an array of simple processing units (SP) grouped together into multiprocessor units (MP). A GPU consists of several such multiprocessors operating in parallel. Typical GPUs have a combined SP count of around 200. Each multiprocessor contains a small amount of local shared memory and a set of registers that are shared among all of the simple processing units in that multiprocessor. Additionally, each multiprocessor has a dedicated texture unit, which can perform array lookup and interpolation operations. The texture unit also has a small cache, improving performance for temporal & spatial locality in texture references. Global device memory is accessible to all multiprocessors and texture units and provides high bandwidth but long-latency accesses to the on-board DRAM.

CUDA applications are composed of many threads executing in parallel, which are grouped into blocks. All the threads in each block execute on a single MP in "warps" of 32 threads at a time. Threads within a block can communicate and synchronize with each other efficiently, but communication and synchronization across blocks is limited and slow.

While GPUs offer tremendous computational resources, care must still be taken in programming the computation kernels. Extracting high performance requires taking into account architectural features of the GPU. A sufficient number of parallel threads of execution must be available to keep all the SP units busy and tolerant of any delays in computation or memory accesses. The resources (registers and shared memory) required by each thread must be minimized to enable a large number of threads to run on an MP. Additionally, global memory accesses should be kept to a minimum, and threads running on the same MP should access neighboring locations in global memory. Further, for good performance, threads in the same warp should have the same control flow..

3.1 Conventional FBP Implementation

As mentioned in the introduction, conventional filtered backprojection has a very obvious parallel structure, demonstrated in equation (2), where each voxel can be reconstructed independently. The GPU is particularly amenable to an FBP implementation, as a key component of the algorithm is linear interpolation of projection data. The texture unit serves as a data cache that provides linear interpolation in hardware, offloading a good portion of the work required by the FBP kernel. In our implementation of the conventional algorithm, each thread of execution is responsible for reconstructing a particular voxel, performing the geometric calculations on the SP, using the texture unit to interpolate the projection data, and using shared memory to accumulate the contributions of each view. After fully reconstructing the voxel, the result is written to global memory for display or transfer back to the host computer.

3.2 Hierarchical FBP Implementation

While less computationally intensive than the conventional algorithm, the hierarchical backprojector involves much more complex control and data flow. However, the algorithm still exposes a large amount of computational parallelism that can be exploited by the GPU. Our approach involves looking at the decomposition of the hierarchical algorithm in a tree representation, as shown in Figure 3. Each node represents a particular subvolume (with the top-most node representing the full reconstruction volume). Following an edge in the tree represents the application of the shifting and decimation operation for that subvolume. Under this algorithmic representation, we consider grouping work into stages. Each stage involves the processing of all the nodes of a particular depth in the hierarchical decomposition tree. This involves decomposing each subvolume into its four constituent subpillars and applying the shifting and decimation operation. When the desired final subvolume size has been reached, the last stage is invoked, which performs the conventional backprojection operation on each subvolume in parallel.

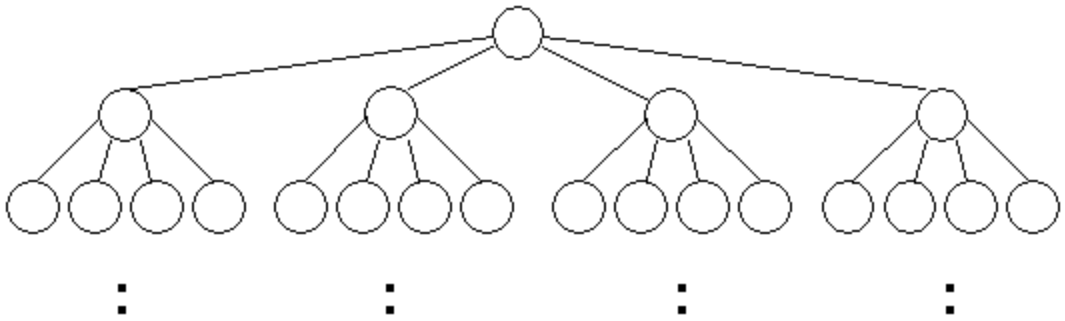

Figure 3. Tree representation of hierarchical decomposition

Organizing the hierarchical algorithm into stages in this manner provides a mechanism for efficiently exposing the parallel computation in the algorithm. In each stage, the decompositions of different subpillar with a common parent subvolume are independent of one another. Furthermore, the shifting and decimation of a dataset involves calculations that are view and detector independent, exposing a further dimension of parallelism. For a stage in the algorithm, each

thread of execution is responsible for a portion of the shifting and decimation effort. The texture cache is used as a memory caching mechanism for accessing the projection data. Finally, the backprojection stage requires only small modifications to the conventional backprojection kernel.

The small amount of local, shared memory, and limited GPU device memory (about 900 MB, much less than modern CPU platforms) challenged us to derive memory-efficient kernels. The tree structure of the algorithm suggests maximum performance could be obtained by executing the parallel work in a given stage all at once. However, this requires that the GPU have sufficient memory to buffer the input and output datasets, as the shifting and decimation operation produces intermediate results that must also be stored. This nearly doubles the memory footprint of the algorithm over the conventional algorithm, exceeding the memory capacity of the GPU.

Our first solution was to process the work in the decomposition tree in a mixed manner, initially starting the decomposition depth-first for the first stage and reverting to the more efficient breadth-first strategies for remaining stages. For efficient dataflow, we allocate four buffers: one for the input dataset for the first stage of decomposition, two to be used as output and input for subsequent stages, and a fourth for the output image. This reduces the memory footprint from double that of the conventional algorithm to about 50% more. With this approach, our datasets fit in the limited memory available, and the input dataset need only be copied once from the CPU. If our memory needs outpace future GPU memory growth, we have several options, including using half-precision 16-bit floating point numbers, dividing the decomposition and backprojection vertically, or processing more stages in a depth-first manner.

As with the conventional algorithm, we exploit the texture unit both for bilinear interpolation and for its cache. We also extensively use shared memory in both the decomposition and backprojection kernels. As a result, only the short initialization phases of each kernel load values directly from slow, global memory, and only the output phases store directly to global memory. In the decomposition kernel, shared memory is primarily used to store state for the filters we apply. It is also used to store small, frequently-used tables. In the backprojection kernel, it used to store sine and cosine values, other tables, and the voxels themselves while the contributions from each projection are summed. Other read-only data structures that are too large to be stored in shared memory are accessed as textures.

The grouping of threads has a significant impact on system performance. Threads are carefully mapped to multiprocessors so that texture loads have good locality, global memory accesses are coalesced to sequential addresses within aligned blocks of memory, and the tables and temporary storage fit in shared memory.

4. RELATED WORK

This is the first GPU implementation of a fast hierarchical $O(N^3 \log N)$ filtered backprojection algorithm for the circular conebeam geometry, but many have implemented conventional $O(N^4)$ algorithms on GPUs. More common are projects that implement backprojection using the GPU graphics pipeline rather than the general-purpose general-purpose computation on GPU (GPGPU) environment. Xu and Mueller[6] developed a GPGPU and graphics-pipeline backprojector and reported a three-fold performance improvement in using the graphics-pipeline version. Zhao et al[7] also developed a backprojector using the graphics pipeline. Their work exploited rotational and mirror symmetry to avoid redundant trigonometric calculations.

In 2008, Noel et al[8] developed a FBP system using GPGPU. They acknowledge the gap between graphics pipeline and GPGPU performance and predict that future GPUs will bridge this gap. Nvidia indeed has continued to expose more of the hardware in each release of their CUDA GPGPU platform, and our evaluation shows that our conventional GPGPU algorithm is broadly competitive with graphics pipeline-based algorithms. The decomposition algorithm simply does not conform to the rigid sequence of stages in the graphics pipeline, so we chose the GPGPU approach.

5. RESULTS

The goal of the hierarchical backprojection algorithm is to increase reconstruction rates while preserving image quality relative to the conventional backprojection operation. Projection data was simulated for the Forbild head and thorax phantoms[9] with parameters typical of a 64-row CT scanner. The acquisition and reconstruction configuration parameters are listed in Table 1. Note that the detector spacing values are after the detector panel has been translated to the center of rotation.

Table 1. System geometry

Acquisition		Reconstruction	
Source to object distance	600 mm	Ramp filter window	Hamming
Number of detector channels	983	Cross-section size	512 x 512
Detector channel spacing	10^{-3} rad	Voxel size (thorax)	0.7812 mm
Number of detector rows	64	Number of slices (thorax)	53
Row spacing	1 mm	Voxel size (head)	0.4688 mm
Number of views	1024	Number of slices (head)	105

The backprojection algorithm was executed on an NVIDIA GTX 260 GPU with 24 multiprocessors and 896 MB of global memory. Comparisons of reconstructions of the two phantoms with the conventional and hierarchical FBP algorithms are shown in Figures 4, 5, 6, and 7. These figures demonstrate the excellent agreement of the two algorithms, indicating the preservation of image quality of the conventional reconstruction. The comparison in runtime was calculated by execution time of the kernels on the GPU; no transfer time to/from the host PC was added. Results are shown in Table 2. The runtime of the hierarchical algorithm is more dependent on the size of the projection data occupied by the object than is the conventional algorithm. Thus the thorax, being the larger object, requires more projection data to be processed, increasing memory pressure and reducing performance. We are currently investigating techniques for improving the efficiency of the GPU implementation under these conditions. A higher-end graphics card with more memory would also increase our reconstruction rates, and increase the speedup provided by the hierarchical algorithm relative to the conventional algorithm. In comparing the performance of our conventional FBP algorithm to other published GPU implementations, we find similar runtimes after normalizing for dataset sizes and hardware specifications. Therefore our conventional FBP implementation serves as a reasonably well optimized competitor.

Table 2. Algorithm runtime comparison

	Thorax Phantom	**Head Phantom**
Conventional FBP runtime	1185 ms	1678 ms
Conventional FBP slices per second	42	63
Hierarchical FBP runtime	485 ms	289 ms
Hierarchical FBP slices per second	109	363
Speedup	2.4x	5.8x

6. CONCLUSION

These results demonstrate the unprecedented reconstruction rate of the hierarchical backprojection algorithm compared to the conventional algorithm running on a GPU. This speedup, coupled with the positive results of the image quality evaluation demonstrate the feasibility of an ultra-fast, high-quality GPU-based image reconstruction engine using hierarchical backprojection algorithms. Trends of ever-increasing performance in GPUs indicate the potential of a real-time, low-cost 3D reconstruction system.

With the release of their next generation “Fermi”-based GPUs, NVidia has emphasized general supercomputing over graphics performance. This bodes well for the performance of our hierarchical algorithm on future hardware. Whereas the literature clearly shows the triumph of GPU conventional backprojection over CPU conventional backprojection on contemporary hardware, the decomposition kernel mapping is much more challenging, with full performance limited by the meager memory resources on each MP. Fermi provides far greater memory resources, from which the hierarchical algorithm can derive significant benefit.

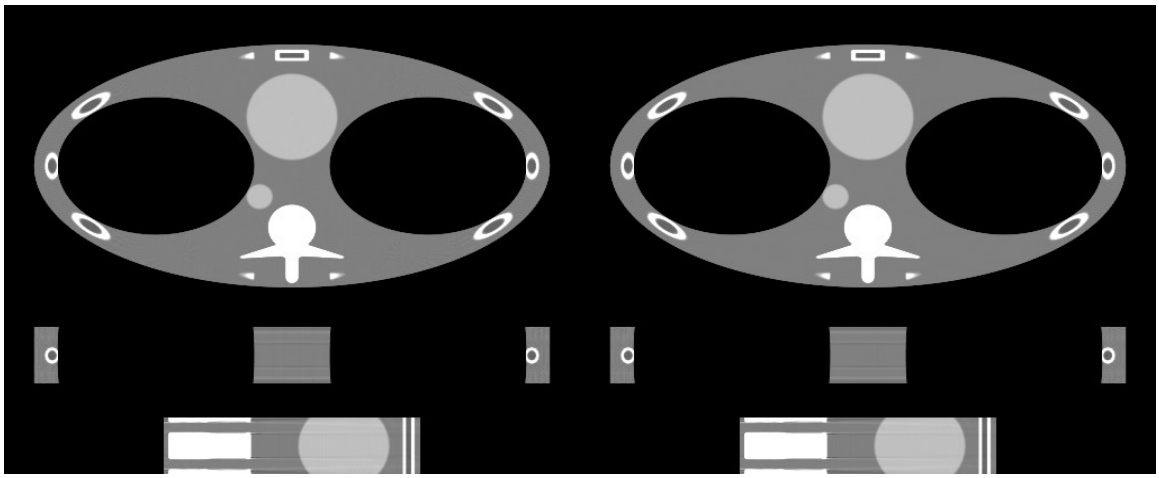

Figure 4. Slices through thorax phantom reconstructions for conventional (left column) and hierarchical (right column). Grayscale window is [900 1100].

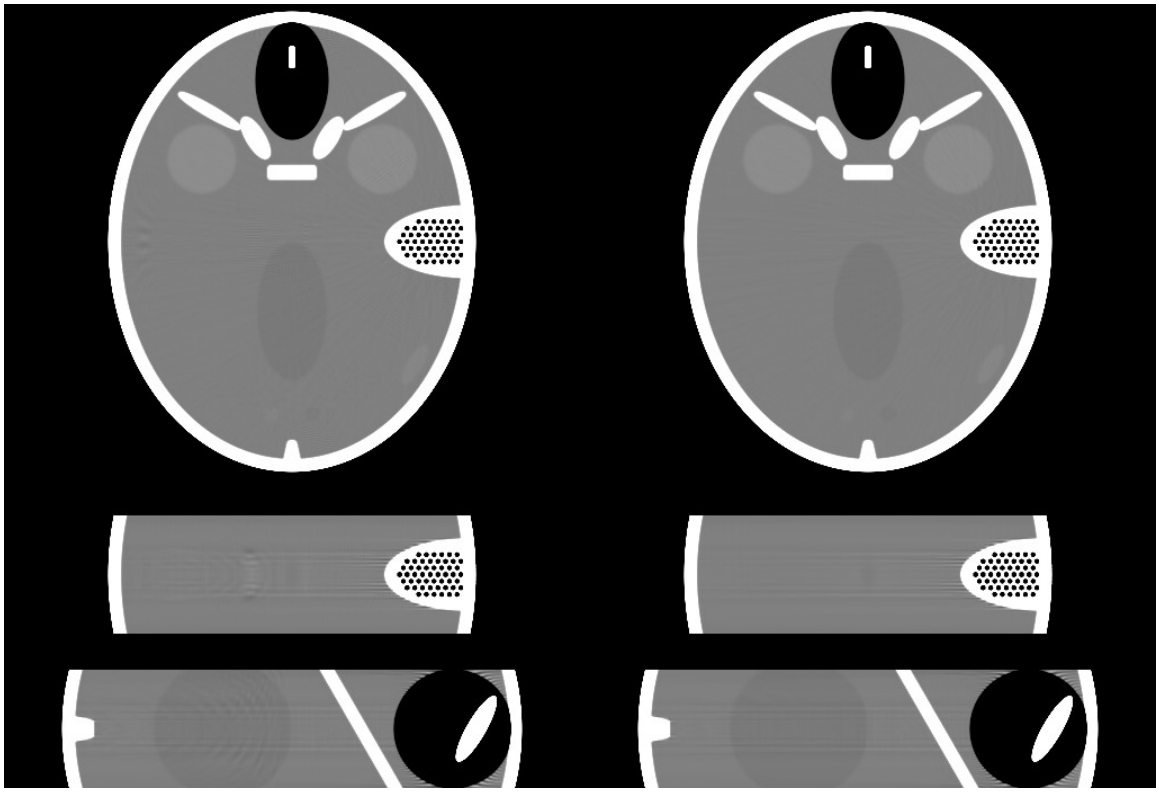

Figure 5. Slices through head phantom reconstructions for conventional (left column) and hierarchical (right column). Grayscale window is [950 1150].

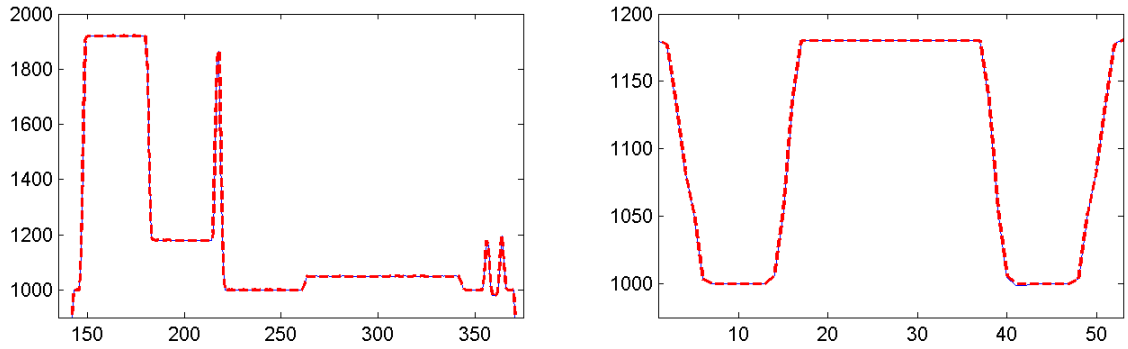

Figure 6. Cuts through thorax phantom reconstructions for conventional (solid, thin, blue) and hierarchical (dashed, thick, red) . Left column is (x=256, z=27) and right column is (x=256, y=200).

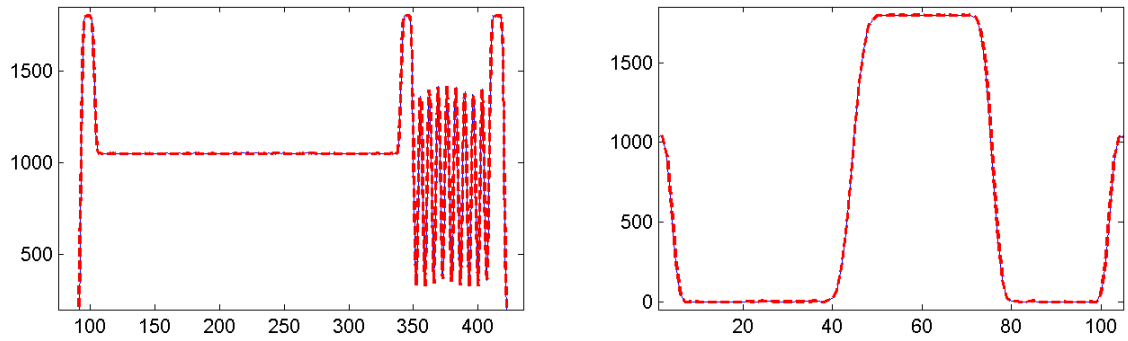

Figure 7. Cuts through head phantom reconstructions for conventional (solid, thin, blue) and hierarchical (dashed, thick, red) . Left column is (x=256, z=53) and right column is (x=256, y=400).

ACKNOWLEDGEMENTS

The project described was supported by a Phase II SBIR Award Number R44EB005067 from the National Institute of Biomedical Imaging and Bioengineering to InstaRecon, Inc. The content of this publication is solely the responsibility of the authors and does not necessarily represent the official views of the National Institute of Biomedical Imaging and Bioengineering or the National Institutes of Health.

REFERENCES

[1] Xiao, S. Bresler, Y. and Munson, D. C., “Fast Feldkamp Algorithm for Cone-Beam Tomographic Reconstruction,” Proc. ICIP 2, 819-822 (2003).

[2] Brokish, J., Fast Backprojection Algorithms for Divergent Beam Tomography, Ph.D. Dissertation, University of Illinois at Urbana-Champaign (2004).

[3] Brokish, J. and Bresler, Y., “Ultra-fast hierarchical backprojection for micro-CT reconstruction," 2007 IEEE Nuclear Science Symposium Conference Record, 4460 – 4463 (2008).

[4] Brokish, J. and Bresler, Y., “Combined algorithmic and hardware acceleration for ultra-fast backprojection," 2007 IEEE Nuclear Science Symposium Conference Record, 3915-3918 (2008).

[5] Brokish, J. and Bresler, Y., “Sampling requirements for circular cone beam tomography," 2006 IEEE Nuclear Science Symposium Conference Record, 2882-2884 (2007).

[6] Xu, F. and Mueller K., "Real-time 3D computed tomographic reconstruction using commodity graphics hardware," Physics in Medicine and Biology 52, 3405-3419 (2007).

[7] Zhao, X., Hu, J., and Zhang, P., "GPU-Based 3D Cone-Beam CT Image Reconstruction for Large Data Volume," International Journal of Biomedical Imaging, (2009).

[8] Noel, P., Walczak, A., Hoffmann, K., Xu, J., Corso, J., and Schafer, S., "Clinical Evaluation of GPU-Based Cone Beam Computed Tomography," in Proc. of MICCAI Workshop: High-Performance Medical Image Computing and Computer Aided Intervention, (2008).

[9] www.imp.uni-erlangen.de/phantoms/