# Clinical Evaluation of GPU-Based Cone Beam Computed Tomography.

Peter B. Noël[1,2], Alan M. Walczak[2], Kenneth R. Hoffmann[1,2], Jinhui Xu[1], Jason J. Corso[1], and Sebastian Schafer[2]

[1] Department of Computer Science and Engineering, The State University of New York at Buffalo, USA, pbnoel@buffalo.edu,
[2] Toshiba Stroke Research Center, The State University of New York at Buffalo, USA.

**Abstract.** The use of cone beam computed tomography (CBCT) is growing in the clinical arena due to its ability to provide 3-D information during interventions, its high diagnostic quality (sub-millimeter resolution), and its short scanning times (60 seconds). In many situations, the short scanning time of CBCT is followed by a time consuming 3-D reconstruction. The standard reconstruction algorithm for CBCT data is the filtered backprojection, which for a volume of size $256^3$ takes up to 25 minutes on a standard system. Recent developments in the area of Graphic Processing Units (GPUs) make it possible to have access to high performance computing solutions at a low cost, allowing for use in applications to many scientific problems. We have implemented an algorithm for 3-D reconstruction of CBCT data using the Compute Unified Device Architecture (CUDA) provided by NVIDIA (NVIDIA Cor., Santa Clara, California),which was executed on a NVIDIA GeForce 8800GT. Our implementation results in improved reconstruction times from on the order of minutes, and perhaps hours, to a matter of seconds, while also giving the clinician the ability to view 3-D volumetric data at higher resolutions. We evaluated our implementation on ten clinical data sets and one phantom data set to observe differences that can occur between CPU and GPU based reconstructions. By using our approach, the computation time for $256^3$ is reduced from 25 minutes on the CPU to 4.8 seconds on the GPU. The GPU reconstruction time for $512^3$ is 11.3 seconds, and $1024^3$ is 61.4 seconds.

## 1 Introduction

Computed Tomography is one of the most popular modalities in the clinical arena, but reconstruction of cone beam computed tomography (CBCT) data can be time consuming on a standard system. Solutions that reduce the turnaround time would provide advantages during both diagnostic and treatment interventions, e.g., real-time reconstruction and high resolution reconstruction.

The high demand for realism in computer games has pushed the development of Graphic Processing Units (GPUs). As a result, the performance of these units themselves are multiple times higher than the supercomputers of only a decade

ago. Therefore, it is practical to apply the power of GPUs to problems that exist in the field of medical imaging.

We use a NVIDIA GeForce 8800GT which provides high performance for a relatively low cost (300 US dollars). The advantage of a NVIDIA product is that a C-like programming environment, called Compute Unified Device Architecture (CUDA), is provided.

CUDA has several advantages over traditional low-level GPU programming languages. For example, it uses the standard C language, it allows for access to arbitrary addresses in the device's memory, it allows user-managed shared memory (16KB in size) that can be shared amongst threads, and it utilizes faster downloads and readbacks to and from the GPU. However, in comparison to traditional CPU calculations, the GPU computations have some disadvantages. These include no support for recursive functions, bottlenecks may result due to bandwidth limitations and latencies between the CPU and the GPU, and the GPU's deviations from the IEEE 754 standard [1], which includes no support for denormals and signalling NaNs, support for only two IEEE rounding modes, and lower precision in floating-point math functions.

Since computed tomographic reconstruction is computational very demanding, several approaches to speed up the process have been developed in recent years. A comprehensive summary of the different approach is given in [2], where four different approaches are compared (PC Reference, Field Programmable Gate Arrays (FPGAs) Platform, Graphic Processing Unit (GPU) Platform and Cell Platform). The system parameter for all techniques are 512 projections, with a projection size of $1024^2$ and a volume of $512^3$. The reconstruction times are as follows: PC 201 seconds, FPGA 25 seconds, GPU 37 seconds and Cell 17 seconds. A direct comparison between the different approaches is difficult since the architecture of the hardware used, especially for GPUs, is frequently updated and may include additional new features.

Several groups have worked on implementing CT reconstructions on the GPU. Through the last decade, the main contributions in accelerated CT have been made by Klaus Mueller, et al.[3][4], where different implementations and programming platforms are used to show the ability of the graphic accelerator. In [4], a streaming CT framework is presented which pipelines the process; the convolution is done on the CPU and the backprojection on the GPU. A similar implementation by using CUDA for parallel beam and cone beam is presented in Haiquan Yang et. al. [5]. Reconstruction of CBCT data from mobile C-arm units by using NVIDIA devices is presented in [6][7].

Our approach is distinct from the previous work. We have developed a solution that takes advantage of the available shared memory, loads all projection images into the GPU memory, and computes the intensity of each voxel by back-projecting in parallel. We investigate the limitation and differences between the reconstruction on GPUs and on CPUs, which most likely occur as a result of the

[1] The IEEE Standard for Binary Floating-Point Arithmetic (IEEE 754) is the most widely-used standard for floating-point computation, and is followed by many CPU implementations.[1]

deviation from the IEEE 754 standard. We monitored the differences by performing a clinical evaluation of ten animal cases and one phantom case. Due to the hardware differences between GPUs, e.g. clock speed and memory size, and variations in the system parameters of different computed tomography modalities, a direct comparison between implementations is difficult to perform.

The rest of the paper is organized as follows. In sections 2.1, we revisit the filtered backprojection method, and in Section 2.2, we present our CUDA implementation. In section 3, we show how the algorithm proposed is evaluated, and in section 4 the results are shown. Finally, in section 5, a discussion is given.

## 2 Method

### 2.1 Cone-Beam Computed Tomography (CBCT)

In this section, we revisit a reconstruction method for CBCT data as introduced by Feldkamp, et al. [8]. Since we use a rotational angiographic system (Toshiba Infinix VSI/02) equipped with a flat panel detector, we only discuss the case of equally spaced planar detectors.

In Figure 1, the schematic drawing of the cone beam system with a planar detector is presented. During acquisition, the system follows a circular trajectory, with a radius of $D$ placed at the origin. The detector lies perpendicular to the central axis of the x-ray beam.

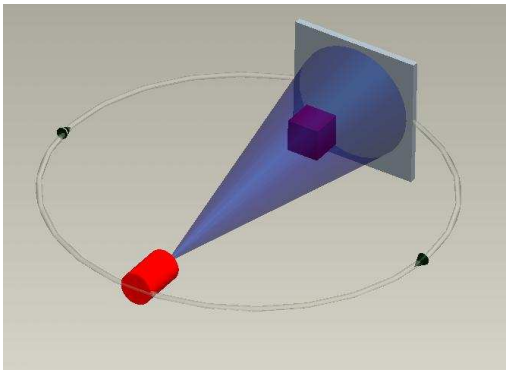

**Fig. 1.** Systematic drawing of a cone beam system.

The projection image $P(\cdot)$ at angular position $\Theta$ is the line integral along the x-ray beam. A set of projections are acquired at $t$ discrete source positions with uniform angular spacing $\Delta\Theta$. During CBCT, $\Theta$ range is about $210^\circ$ with angular separations of $2^\circ$. A full rotation is not possible due to mechanical limitations.

The reconstruction method is formulated as a weighted filtered backprojection. As an initial step, the projection data are log converted, individually

weighted and ramp filtered ($P_f$). Next, the 3-D volume is reconstructed by a backprojection. Let $r = [x, y, z]$ be the 3-D position in the volume, and let ($u$,$v$) denote the position of the intersection with the detector plane of the ray starting from the source and passing through point $r$. Therefore, the backprojection is:

$$f(\boldsymbol{r}) = \sum_{\Theta} P_f[u(x, z, \Theta), v(y, z, \Theta), \Theta], \tag{1}$$

where

$$u = (SID * x)/(ISO - z), \tag{2}$$

$$v = (SID * y)/(ISO - z), \tag{3}$$

SID is the source-to-image-distance, and ISO is the source-to-isocenter distance, where the isocenter is the point about which the system rotates. Since $u$ and $v$ usually do not correspond to a discrete pixel position, we use bilinear interpolation to determine its value in the image. The computational cost of cone-beam computed tomography for a volume of size $N^3$ is $O(N^4)$.

### 2.2 GPU-Based Implementation

**GPU Architecture.** The architecture of the GPU is built for high performance because it is needed for the intensive and highly parallel computations necessary for computer graphics. They are designed with more transistors devoted to data processing rather than data caching and flow control. More specifically, the GPU is well suited to address problems that can be expressed as data-parallel computations, where the same program or kernel is executed on many data elements simultaneously. Data-parallel processing maps data elements to parallel processing threads.

For the NVIDIA architecture, the kernel is compiled to the instruction set of the device and the resulting program can be called by multiple threads. A thread block is a batch of threads that can cooperate with each other by efficiently sharing data using the fast shared memory and synchronizing their execution to coordinate memory accesses. There is a maximum number of threads that a block can contain. However, blocks of same dimensionality and size that execute the same kernel can be batched together into a grid of blocks, so that the total number of threads that can be launched to execute a single kernel is much larger.[9]

**GPU-Based 3-D Computed Tomography.** The backprojection algorithm is the most computationally intensive portion of the reconstruction process. Therefore, we will focus on its implementation. However, the first step in the algoritm is the logarithmic conversion and filtering of the projection images. Both steps are implemented as GPU routines by using shared memory. As a separate step, the time for these operations is reduced from minutes to seconds.

Since the on-board GPU memory is limited, in our case 512Mb, it is not possible to load both the entire set of projection images and the full volume

into the memory. Therefore, two different possibilities exist: either to load the full volume and each projection consecutively, or to load all of the projections and then sub-parts of the volume consecutively. We decided to use the second approach for two reasons. First, the rotational angiographic system acquires 106 projection of size $1024^2$ with an angular separation of $2^\circ$ which makes it possible to load all projections at once, and second, experiments have shown that this approach performs better in terms of the total running time.

We present the pseudo-code for our implementation in Algorithm 1. After filtering, the projection images are uploaded to the GPU memory as 2-D textures, allowing us to utilize the efficient bilinear interpolation function provided by CUDA during the backprojection step. Next, we start a voxel-based backprojection by first splitting the problem up into separate slices of the volume, and then separating the slice into several sub-rows of length 256 or 512 depending on the volume size as illustrated in Figure 2. On the GPU, each slice translates to a grid that represents each sub-row of voxels, and each of these sub-rows creates a single block of threads (one thread for each voxel) on which our kernel will execute. For each block of threads, we use the shared memory to save original xyz coordinates of the sub-row, the voxel intensities of the sub-row (initially set to zero), and the translation and rotations matrices, which describe the geometric relationship between the different projection images (determined previously in a calibration step). The xyz coordinates are calculated in the following way:

$$x = T_{N_x} * G_x + T_x \tag{4}$$

$$y = slice \tag{5}$$

$$z = G_y \tag{6}$$

where $T_{N_x}$ is the size of a block of threads (256 or 512), $G_x$ and $G_y$ are the grid blocks indices and $T_x$ is the thread index. For proper projection of a voxel, the thread block index and grid index must be translated to the correct position in the volume. In our case, we use a Right-handed system to determine the direction of the coordinate axes.

**Algorithm 1** Algorithm for GPU-based backprojection

```
copy all projection into GPU memory as textures
for each slice in volume do
    initialize voxel intensities of the current sub-row to zero in shared memory
    calculate coordinates of voxel in sub-row into shared memory
    copy all rotation and translation matrices into shared memory
    for each projection image do
        apply rotation and translation matrices to voxel coordinates
        project voxel into the image
        add pixel intensity in image to voxel intensity
    end for
    write sub-row back into volume on host
end for
```

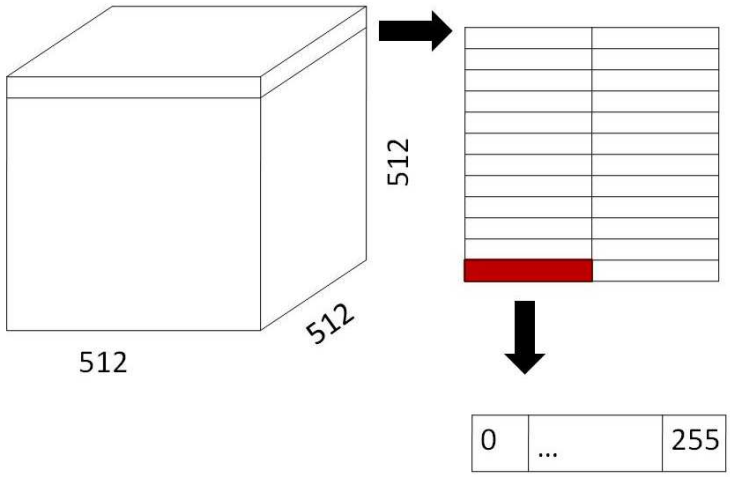


**Fig. 2.** Systematic drawing defining a sub-row. Different sub-rows are backprojected in parallel.

Due to mechanical limitations of the gantry for our cone beam computed tomography unit, the rotation range is 210 degrees. In a short-scan case like this, the introduction of a weighting function to handle redundant data is needed. Parker introduced such a weighting function for a scan over $\Pi$ plus the opening angle of the fan. [10] These weighting functions lead to mathematically exact reconstructions in the continuous case. Therefore, we implemented these weights and also the original weights defined by Feldkamp to achieve mathematically correct reconstructions.

## 3 Evaluations

For all evaluations, we used a standard system (Pentium 4, 3.2 GHz, 4 GB of RAM) equipped with a NVIDIA Geforce 8800GT. The performance profile of the GPU is: 112 Stream Processors with 1500 MHz Shader Clock which equals a peak performance of 366 GFlops[2].

To evaluate the speed up over the CPU provided by the GPU, we determine total time, convolution time, and the backprojection time. The total time is the sum of convolution and backprojection time. Additionally, we determine frames per second [fps] and megavoxels per second [Mvps]. Frames per second is the number of projections divided by the time need for convolution . Megavoxels per second is the number of voxels within the volume divided by the time needed for backprojection.

[2] In high performance computing, Flops is an acronym meaning Floating point Operations Per Second, which is a measure of a computer's performance.

Additionally we evaluate our algorithm on two different types of CBCT data, head phantom (Figure 3) and animal study data. For both types of data, 106 projections of size $1024^2$ with an angular separation of $2^\circ$ were acquired. The gantry has a source-to-image distance of 110 cm, a source-to-isocenter distance of 75 cm, a pixel size of 0.019 cm, and an angular speed of 50 dps. Distortion correction of the projections is not necessary since our system is equipped with a flat panel detector. We compare the intensity profile across the horizontal medial axes of the center slices from both reconstruction methods. The intensity profile is compared for one animal case and the head phantom.

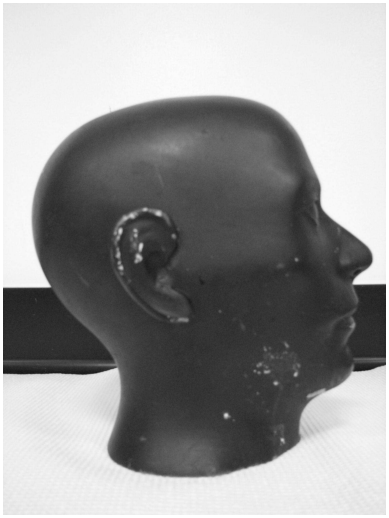

**Fig. 3.** The head phantom

For the ten animal case, we calculate for each one the average difference between the volumes as:

$$\Delta = 1/N * \sum ||vol_{CPU}(x, y, x) - vol_{GPU}(x, y, z)||, \tag{7}$$

where, N is the total number of voxels within the volume. Note, the volumetric difference is calculated for volumes of dimension $256^3$ since larger data sets would be extremely time consuming to reconstruct on the CPU.

## 4 Results

Table 1 shows the reconstruction time of three different volume sizes ($256^3$, $512^3$, $1024^3$). The total reconstruction time is substantially reduced compared to the standard CPU times, while providing reconstructed volumes of a higher resolution. The overall performance of our approach with existing techniques is

comparable. For a $512^3$ volume, Mueller, et al., [4] achieves a reconstruction time of 8.9 seconds for a GPU approach, and Yang, et al., [5] a reconstruction time of 8.7 seconds for their GPU approach. Note these times for these techniques are reported for reconstructions using 360 projections, where our approach only uses 106 projections. Also it seems Parker weights have not been implemented in both techniques, and in the second publication bilinear interpolation is not reported to be used. For these techniques, the reported GPU results are executed on a GPU with a high performance profile, making it difficult to directly compare the results of the other approaches with our results.

**Table 1.** Results for different volume sizes

| Volume Size | total [s] | convolution [s] | [fps] | backprojection [s] | [Mvps] |
|---|---|---|---|---|---|
| $256^3$ | 5.07 | 3.58 | 29.6 | 1.50 | 11.22 |
| $512^3$ | 12.61 | 3.58 | 29.6 | 9.03 | 14.86 |
| $1024^3$ | 61.29 | 3.58 | 29.6 | 57.71 | 18.61 |

**Head Phantom.** In Figure 4, we present the center slices from the GPU and CPU reconstructions and the intensity profile across the horizontal central axis. Visually the intensity profiles for both approaches identically in shape with small variations.

**Animal Study.** In Figure 5, we show the centerslices from the GPU and the CPU reconstruction, and the intensity profile across the horizontal median axis. Visually, the intensity profiles for both approaches identically in shape with small variations.

Figure 6 illustrates the differences between the volumetric data as calculated from Equation 7. The average difference over all cases is 26.7 Hu, which is about 2% considering a Hounsfield range of 1400. The maximal error over all cases is 15.3%. These errors are caused by different factors, which could be as a result of the varition from the IEEE 754 standard, the bilinear interpolation provided by CUDA, or implementational differences. Since the differences between the data are minor, we can assume that the variation is not significant for the computed tomography problem.

Finally, a 3-D rendering of one rabbit head is presented in Figure 7, showing that high resolution renderings like this are now achievable in short amount of time, using a low cost standard system.

## 5 Discussion

In this paper, we presented an efficient, clinically orientated algorithm to reconstruct computed tomography data in almost real-time, demonstrating the power of GPUs in the field of medical imaging. For future work, implementations of other medical imaging problems using a GPU should be considered. In the field

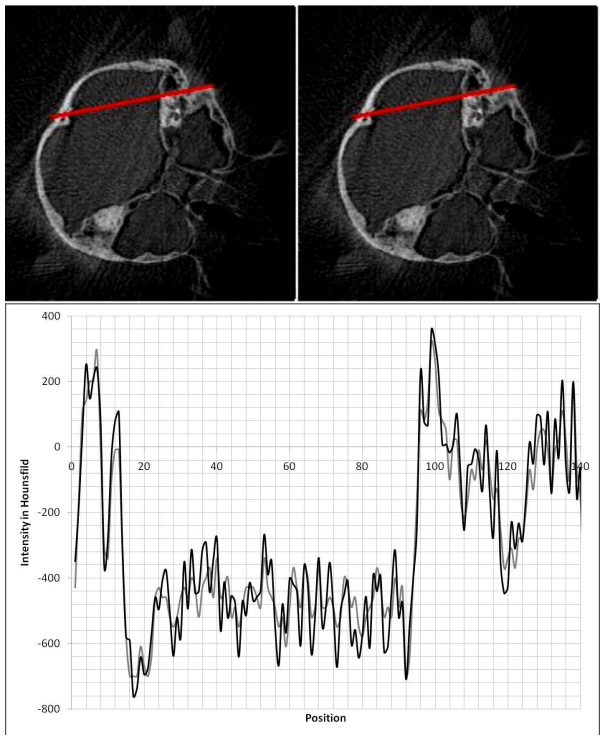


**Fig. 4.** Reconstruction from CPU(left), GPU(right), and intensity profiles from the GPU-CT in gray and for the CPU-CT in black(lower)

of computed tomography, there exists other more efficient reconstruction algorithms whose running time may benefit by using a similar approach.

In our evaluations, we report promising results on two different types of data sets. The amount of time needed for reconstruction is significantly reduced. For a three dimensional volume with dimensions $512^3$, our algorithm completes after 11.3 seconds. Compared to implementations which use a shader (CG, GLSL) our implementation is slightly slower. This slowdown is caused by the fact that with CUDA, the graphics subsystem ASIC hardware cannot be exploited for some of the operations. Nevertheless, CUDA is the latest platform provided by NVIDIA and therefore it is likely that these issues will be improved in time, allowing us to make improvements in our implementation, and therefore our results. Another limitation of programming on a GPU, from which most implementations will suffer, is the fact there exists a bottleneck between the CPU and GPU memory when transferring large datasets. Improvements in the hardware bandwidth between the CPU and GPU will further improve reconstruction times.

In the future, higher resolution volumes could become standard since they provide more information for diagnostic and treatment purposes. In our implementation, our kernel function allows reconstruction of all volume sizes which

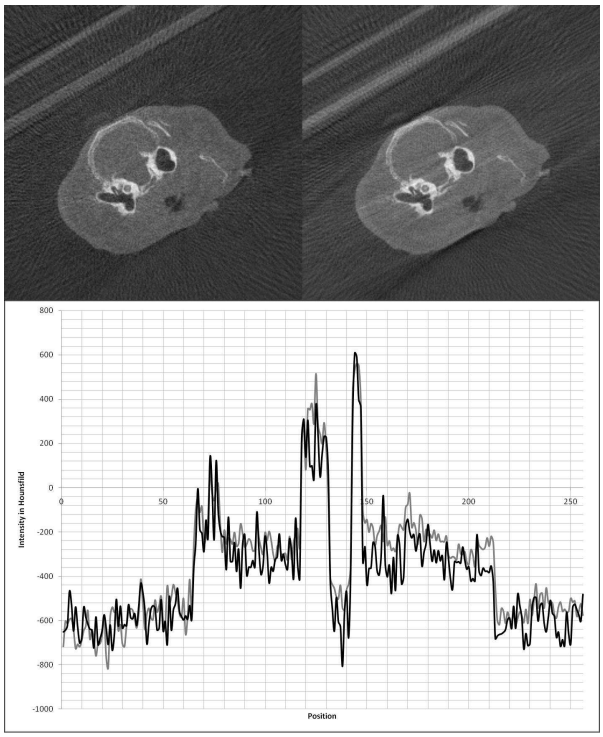


**Fig. 5.** Reconstruction from CPU(left), GPU(right), and intensity profiles from the GPU-CT in grapy and for the CPU-CT in black(lower)

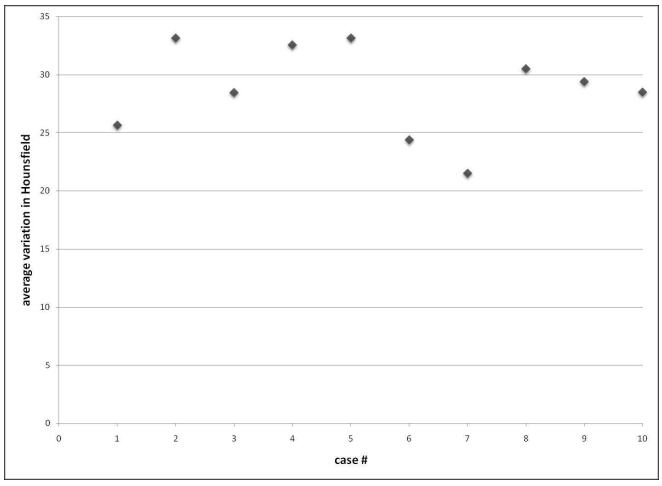


**Fig. 6.** The average gray value variation for the 10 clinical cases.

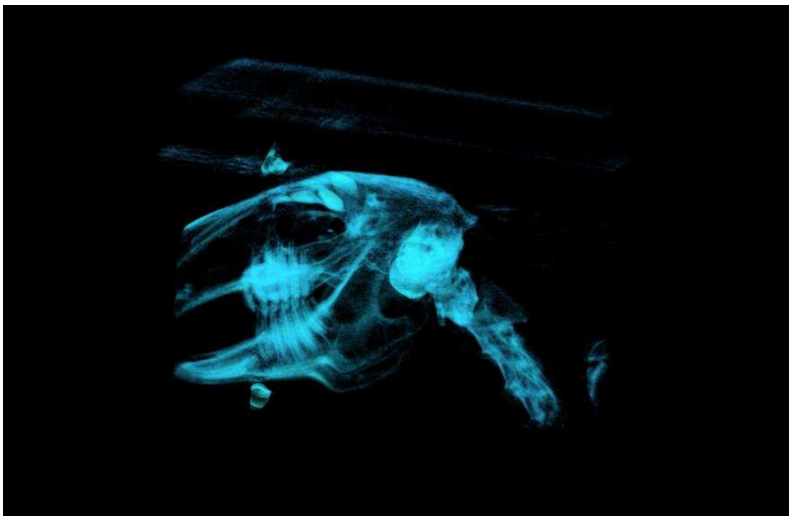

**Fig. 7.** A 3-D rendering of the rabbit data reconstructed using GPUs.

are a multiples of 256. The additional weighting functions make the approach a little slower but results in a mathematically correct reconstruction. The time and relatively simple implementation by using CUDA makes our approach attractive compared to other CPU based techniques.

## 6 Acknowledgements

This work was partly supported by NIH Grant EB002916, NIH Grant HL52567, NSF grant IIS-0713489, NSF CAREER Award CCF-0546509, and the Toshiba Medical Systems Corporation.

## References

1. Hough, D.: Applications of the proposed IEEE-754 standard for floating point arithmetic. Computer **14**(3) (March 1981) 70–74
2. Bockenbach, O., Schuberth, S., Knaup, M., Kachelriess, M.: High Performance 3D Image Reconstruction Platforms; State of the Art, Implications and Compromises. In: 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine. Volume 9. (2007) 17–20
3. Xu, F., Mueller, K.: Real-time 3D computed tomographic reconstruction using commodity graphics hardware. Phys Med Biol **52** (Jun 2007) 3405–3419
4. Mueller, K., Xu, F., Neophytou, N.: Why do commodity graphics hardware boards (GPUs) work so well for acceleration of computed tomography? In: Medical Imaging 2007: Keynote, Computational Imaging V. Edited by Hsieh, Jiang; Flynn, Michael J.. Proceedings of the SPIE, Volume 6510, (2007). Volume 6510 of Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference. (2007)

5. Yang, H., Li, M., Koizumi, K., Kudo, H.: Accelerating Backprojections via CUDA Architecture. In: 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine. Volume 9. (2007) 52–55
6. Churchill, M., Pope, G., Penman, J., Riabkov, D., Xue, X., Cheryauka, A.: Hardware-accelerated cone-beam reconstruction on a mobile C-arm. In: Medical Imaging 2007: Physics of Medical Imaging. Edited by Hsieh, Jiang; Flynn, Michael J.. Proceedings of the SPIE, Volume 6510, pp. 65105S (2007). Volume 6510 of Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference. (March 2007)
7. Scherl, H., Keck, B., Kowarschik, M., Hornegger, J.: Fast gpu-based ct reconstruction using the common unified device architecture (cuda). Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE **6** (Oct. 26 2007-Nov. 3 2007) 4464–4466
8. Feldkamp, L.A., Davis, L.C., Kress, J.W.: Practical cone-beam algorithm. J. Opt. Soc. Am. A **1**(6) (1984) 612
9. NVIDA Corporation, Santa Clara, C.: Nvidia cuda compute unified device architecture, programming guide (2008) [Online; accessed 26-April-2008].
10. Parker, D.: Optimal short scan convolution reconstruction for fanbeam CT. Med Phys **9** (1982) 254–257