COMPUTER METHODS AND PROGRAMS IN BIOMEDICINE XXX (2009) XXX–XXX

journal homepage: www.intl.elsevierhealth.com/journals/cmpb

GPU-based cone beam computed tomography

Peter B. Noël[a,b,*], *Alan M. Walczak*[a,c], *Jinhui Xu*[a,b], *Jason J. Corso*[a,b], *Kenneth R. Hoffmann*[a,c], *Sebastian Schafer*[a,c]

[a] *The State University of New York at Buffalo, USA*
[b] *Department of Computer Science and Engineering, USA*
[c] *Toshiba Stroke Research Center, USA*

ARTICLE INFO

Article history:
Received 20 February 2009
Received in revised form
9 August 2009
Accepted 14 August 2009

Keywords:
Graphics Processing Unit
Cone beam computed tomography
Filtered backprojection

ABSTRACT

The use of cone beam computed tomography (CBCT) is growing in the clinical arena due to its ability to provide 3D information during interventions, its high diagnostic quality (sub-millimeter resolution), and its short scanning times (60 s). In many situations, the short scanning time of CBCT is followed by a time-consuming 3D reconstruction. The standard reconstruction algorithm for CBCT data is the filtered backprojection, which for a volume of size 256^3 takes up to 25 min 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 their use in many scientific problems. We have implemented an algorithm for 3D reconstruction of CBCT data using the Compute Unified Device Architecture (CUDA) provided by NVIDIA (NVIDIA Corporation, Santa Clara, California), which was executed on a NVIDIA GeForce GTX 280. Our implementation results in improved reconstruction times from minutes, and perhaps hours, to a matter of seconds, while also giving the clinician the ability to view 3D volumetric data at higher resolutions. We evaluated our implementation on ten clinical data sets and one phantom data set to observe if differences occur between CPU and GPU-based reconstructions. By using our approach, the computation time for 256^3 is reduced from 25 min on the CPU to 3.2 s on the GPU. The GPU reconstruction time for 512^3 volumes is 8.5 s.

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 turn-around 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 GTX 280 which provides high performance for a relatively low cost (US$ 350). The advantage of a NVIDIA product is that a C-like programming environ-

* *Corresponding author at*: The State University of New York at Buffalo, USA. Tel.: +1 716 400 7535.
E-mail address: pbnoel@buffalo.edu (P.B. Noël).
0169-2607/$ – see front matter © 2009 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.cmpb.2009.08.006

ment, 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 (16 kB in size) that can be shared amongst threads, and it utilizes faster downloads and readbacks to and from the GPU. However, in comparison to shader-based languages, CUDA-based implementations are slightly slower. Compared to traditional CPU calculations, the GPU computations have some disadvantages. These include no support for recursive functions, bottlenecks 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 NaNs.

Since computed tomographic reconstruction is computationally very demanding, several approaches to speed up the process have been developed in recent years. The main achievements have been made using Cell Broadband Engines [2], Field Programmable Gate Arrays (FPGAs) [3,4], and GPU [5,6]. A comprehensive summary of the different approaches is given in [7], where four different approaches are compared (PC Reference, FPGAs, GPU and Cell). 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 s, FPGA 25 s, GPU 37 s, and Cell 17 s. 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 GPUs. Over the last decade, the main contributions in accelerated CT have been made by Mueller and coworkers [5,8], where different implementations and programming platforms are used to show the ability of the graphic accelerator. In [8], a streaming-shader-based 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 Yang et al. [6]. Reconstruction of CBCT data from mobile C-arm units by using NVIDIA devices is presented in [9,10].

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 backprojecting in parallel. We investigate the limitation and differences between the reconstruction on GPUs and on CPUs, which most likely primarily occur as a result of the deviation from the IEEE 754 standard. Since our hardware allows different GPU architectures, we evaluate our algorithm using two different architectures, i.e., sm_10 (basic) and sm_13 (double floating point precision). 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.

[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].

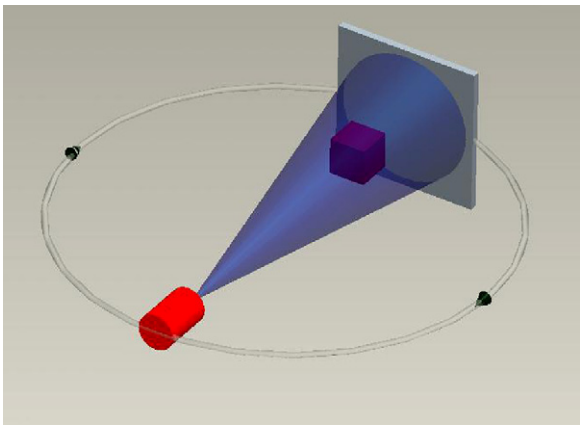

Fig. 1 – Systematic drawing of a cone beam system..

2. Method

2.1. Cone beam computed tomography

In this section, we revisit a reconstruction method for CBCT data as introduced by Feldkamp et al. [11]. Since we use a rotational angiographic system (Toshiba Infinix VSI/02), which is equipped with a flat panel detector, we only discuss the case of equally spaced planar detectors.

In Fig. 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 plane lies perpendicular to the central axis of the X-ray beam.

The projection image $P(\cdot)$ at angular position Θ 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, Θ range is about 210° with angular separations of 2°. 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 3D volume is reconstructed by a backprojection. Let $r = [x, y, z]$ be the 3D 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 given by:

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

where

$$u = \frac{SID \times x}{ISO - z}, \tag{2}$$

Please cite this article in press as: P.B. Noël, et al., GPU-based cone beam computed tomography, Comput. Methods Programs Biomed. (2009), doi:10.1016/j.cmpb.2009.08.006

$$v = \frac{SID \times y}{ISO - z}, \tag{3}$$

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 calculate the 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

2.2.1. GPUs

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 [12].

2.2.2. GPU-based 3D 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 algorithm 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 1024 Mb, 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° 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 2D 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 Fig. 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:

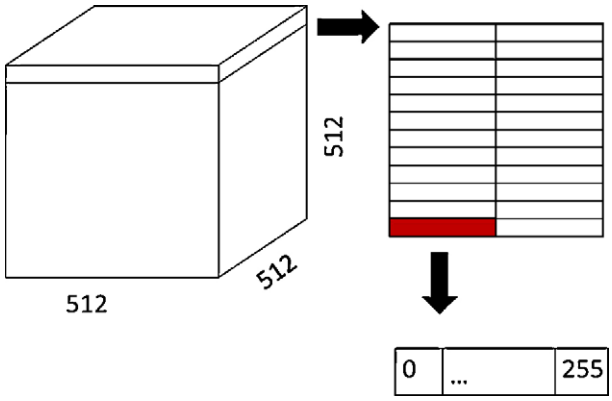

Fig. 2 – Systematic drawing defining a sub-row. Different sub-rows are backprojected in parallel.

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

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

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

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

```
1: copy all projection into GPU memory as textures
2: for each slice in volume do
3:   initialize voxel intensities of the current sub-row to zero in shared memory
4:   calculate coordinates of voxel in sub-row into shared memory
5:   copy all rotation and translation matrices into shared memory
6:   for each projection image do
```

Table 1 – Results for different volume sizes, where GPU-A is the GPU architecture.

Volume size	GPU-A	Total [s]	Convolution [s]	Backprojection [s]
256^3	sm_10	3.2	2.2	1.0
256^3	sm_13	5.1	2.2	2.9
512^3	sm_10	8.5	2.2	6.3
512^3	sm_13	21.7	2.2	21.7

```
7:        apply rotation and translation matrices to voxel coordinates
8:        project voxel into the image
9:        add pixel intensity in image to voxel intensity
10:     end for
11:     write sub-row back into volume on host
12:   end for
```

Due to mechanical limitations of the gantry for our cone beam computed tomography unit, the rotation range is 210°. 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 Π plus the opening angle of the fan [13]. 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 (Intel Core2 Quad, 2.83 GHz, 4 GB of RAM) equipped with a NVIDIA GeForce GTX 280. The performance profile of the GPU is: 240 Stream Processors with 1296 MHz Shader Clock which equals a peak performance of almost 1 TFlops.[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. All results are generated for both architectures (basic and double floating point precision) to see if there are numerical or running time differences.

Additionally, we evaluate our algorithm on two different types of CBCT data, head phantom (Fig. 3) and animal study data. For both types of data, 106 projections of size 1024^2 with an angular separation of 2° 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°/s (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.

For the ten animal cases, we calculate the absolute difference volume between CPU data, vol_{CPU}, and the GPU data, vol_{GPU}, as:

$$\Delta = ||vol_{CPU} - vol_{GPU}|| \quad (7)$$

Next, we determine the arithmetic mean and the standard deviation for all gray values within Δ.

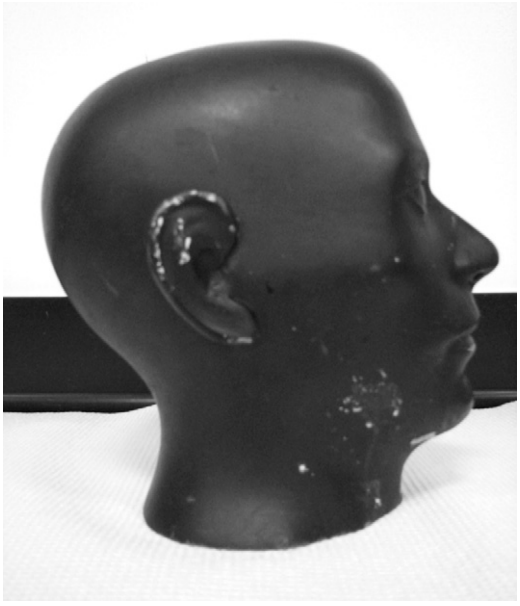

Fig. 3 – The head phantom used in our evaluation.

4. Results

Table 1 shows the reconstruction time of two different volume sizes (256^3, 512^3) and for the two different GPU architectures. Note, additionally we calculated the transfer time between the main memory and the GPU memory. The total transfer time for a 256^3 volume and 106 projection images was 0.75 s, i.e., one fourth of the total reconstruction is devoted to the transfering of data, which is significant. However, the total reconstruction time is substantially reduced compared to the standard CPU times while providing reconstructed volumes of a higher resolution. Compared to the double precision floating point architecture (sm_13), the basic architecture (sm_10) results in a faster reconstruction by up to 60 percent. The overall performance of our approach is comparable to existing techniques. Different types of input data and the frequent release of new GPUs make it difficult to directly compare the results of the other approaches with our results.

[2] In high performance computing, Flops is an acronym meaning floating point operations per second, which is a measure of a computer's performance.

Please cite this article in press as: P.B. Noël, et al., GPU-based cone beam computed tomography, Comput. Methods Programs Biomed. (2009), doi:10.1016/j.cmpb.2009.08.006

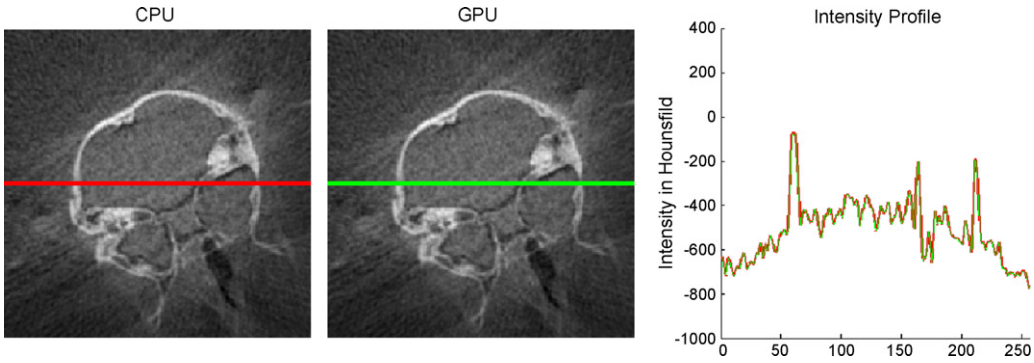

Fig. 4 – Reconstruction from CPU (left), GPU (middle), and intensity profiles from the CPU-CT in red and from the GPU-CT in green (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

4.1. *Head phantom*

In Fig. 4, we present the centerslices from the CPU and GPU reconstructions and the intensity profile across the horizontal central axis. Both reconstructions are performed in double floating point precision. Visually, the intensity profiles for both approaches are identical in shape and comparable numerically.

4.2. *Animal study*

In Fig. 5, we show the centerslices from the GPU and the CPU reconstruction and the intensity profile across the central horizontal axis. Both reconstructions are performed in double floating point precision. Visually, the intensity profiles for both approaches are identical in shape and position.

Fig. 6 illustrates the arithmetic mean difference for the ten different cases. The average arithmetic mean difference over all cases is 0.0137 Hounsfield units (HU). In Fig. 7, we present the RMS difference for all cases. The average RMS difference over all cases is 0.3495 HU. All results are for a Hounsfield range of 1400 HU. These errors are caused by different factors, which could be a result of the variation from the IEEE 754 standard, the bilinear interpolation provided by CUDA, or implementational differences. Since the earlier results are all generated by using double floating point precision, we performed one reconstruction with both architectures sm_10 and sm_13. Further, we calculated the arithmetic mean difference and the RMS difference between the absolute difference of the two volumes. The arithmetic mean is 6.92×10^{-4} HU and the standard deviation is 2.73×10^{-5} HU. Since the differences are minor, we can assume that the differences are not significant for the computed tomography problem.

Finally, a 3D rendering of one rabbit head is presented in Fig. 8, showing that high resolution renderings like this are now achievable in a short amount of time, using a low-cost standard system.

5. Discussion

In this paper, we presented an efficient and 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 of computed tomography, there exists other and 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. We looked at the numerical differences between CPU and GPU implementation,

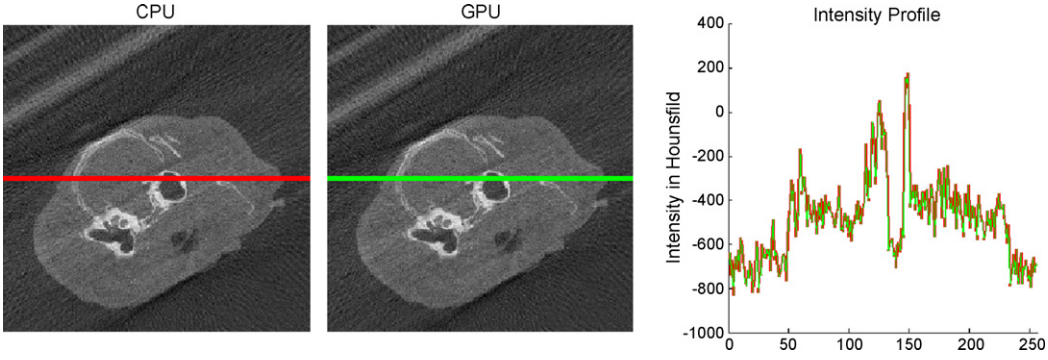

Fig. 5 – Reconstruction from CPU (left), GPU (middle), and intensity profiles from the CPU-CT in red and from the GPU-CT in green (right). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

Please cite this article in press as: P.B. Noël, et al., GPU-based cone beam computed tomography, Comput. Methods Programs Biomed. (2009), doi:10.1016/j.cmpb.2009.08.006

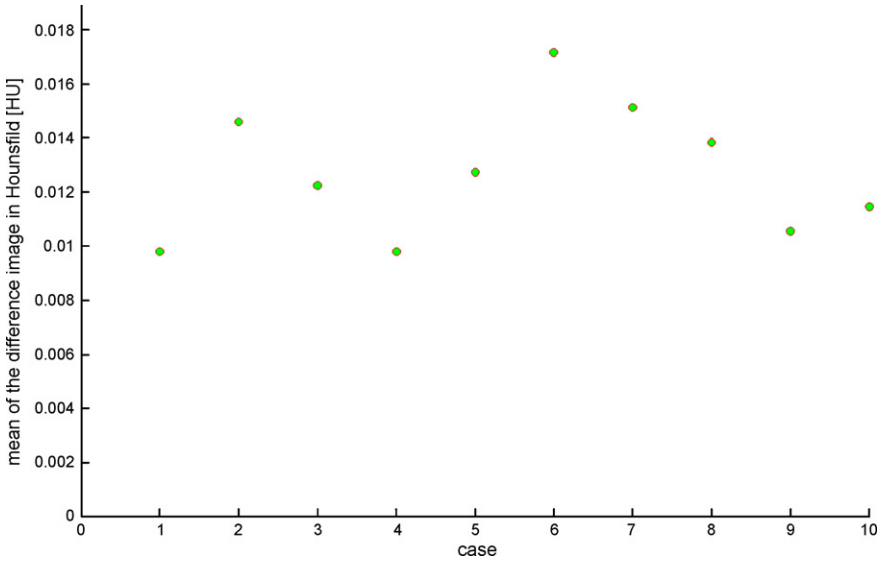

Fig. 6 – The arithmetic mean gray value variation for the 10 clinical cases.

and the differences are not significant. Further, by using the standard GPU architecture (not double floating point precisions), the differences are not significant.

As mentioned in the introduction, 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 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 that a bottleneck exists 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.

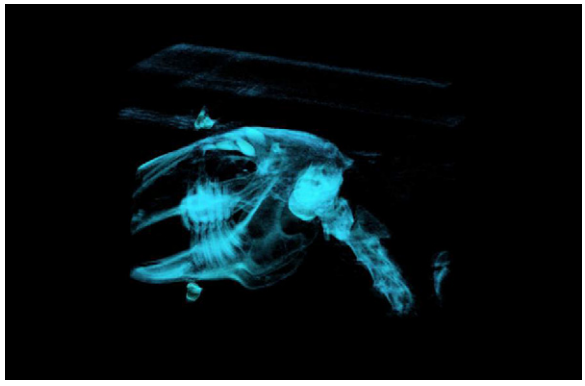

Fig. 8 – A 3D rendering of the rabbit data reconstructed using GPUs.

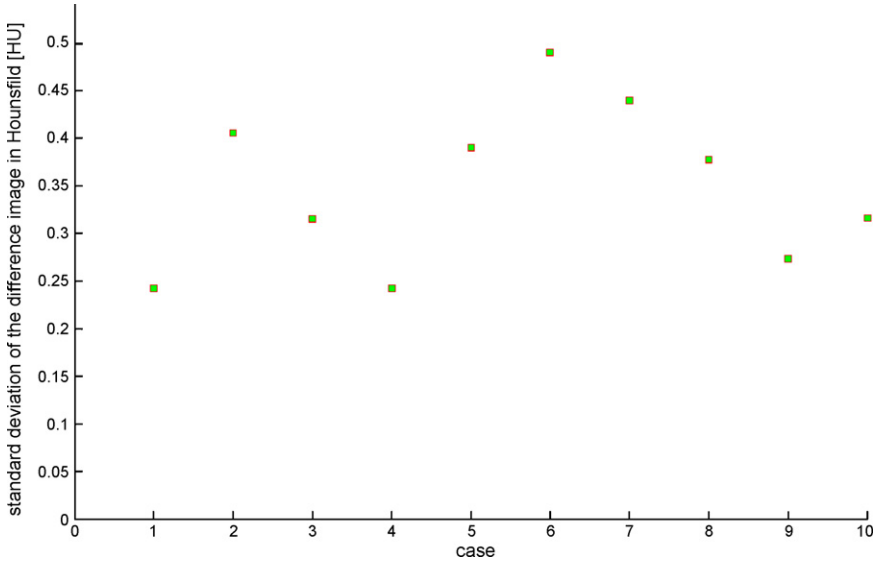

Fig. 7 – The standard deviation of the difference volume for the 10 clinical cases.

Please cite this article in press as: P.B. Noël, et al., GPU-based cone beam computed tomography, Comput. Methods Programs Biomed. (2009), doi:10.1016/j.cmpb.2009.08.006

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 are 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.

Conflict of interest statement

None declared.

Acknowledgements

This work was partly supported by The State University of New York at Buffalo IRD Fund, NSF grant IIS-0713489, NSF CAREER Award CCF-0546509, and the Toshiba Medical Systems Corporation.

REFERENCES

[1] D. Hough, Applications of the proposed IEEE-754 standard for floating point arithmetic, Computer 14 (3) (1981) 70–74.
[2] M. Kachelriess, M. Knaup, O. Bockenbach, Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware, Med. Phys. 34 (2007) 1474–1486.
[3] D. Brasse, B. Humbert, C. Mathelin, M.C. Rio, J.L. Guyonnet, Towards an inline reconstruction architecture for micro-CT systems, Phys. Med. Biol. 50 (2005) 5799–5811.
[4] S. Coric, M. Leeser, E. Miller, M. Trepanier, Parallel-beam backprojection: an FPGA implementation optimized for medical imaging, in: FPGA '02: Proceedings of the 2002 ACM/SIGDA Tenth International Symposium on Field-programmable Gate Arrays, ACM, New York, NY, USA, 2002, pp. 217–226, ISBN 1-58113-452-5, doi:acm.org/10.1145/503048.503080.
[5] F. Xu, K. Mueller, Real-time 3D computed tomographic reconstruction using commodity graphics hardware, Phys. Med. Biol. 52 (2007) 3405–3419.
[6] H. Yang, M. Li, K. Koizumi, H. Kudo, Accelerating backprojections via CUDA architecture, in: 9th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine, vol. 9, 2007, pp. 52–55.
[7] O. Bockenbach, S. Schuberth, M. Knaup, M. Kachelriess, 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, vol. 9, 2007, pp. 17–20.
[8] K. Mueller, F. Xu, N. Neophytou, Why do commodity graphics hardware boards (GPUs) work so well for acceleration of computed tomography?, in: Jiang Hsieh, Michael J. Flynn (Eds.), Medical Imaging 2007: Keynote, Computational Imaging V, Proceedings of the SPIE, vol. 6510, Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, vol. 6510, 2007.
[9] M. Churchill, G. Pope, J. Penman, D. Riabkov, X. Xue, A. Cheryauka, Hardware-accelerated cone-beam reconstruction on a mobile C-arm, in: Jiang Hsieh, Michael J. Flynn (Eds.), Medical Imaging 2007: Physics of Medical Imaging, Proceedings of the SPIE, vol. 6510, Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, vol. 6510, 2007, pp. 65105S, doi:10.1117/12.711797.
[10] H. Scherl, B. Keck, M. Kowarschik, J. Hornegger, Fast GPU-Based CT Reconstruction using the Common Unified Device Architecture (CUDA), Nuclear Science Symposium Conference Record, 2007. NSS '07. IEEE 6, 26 October 2007–3 November 2007, pp. 4464–4466, ISSN: 1082–3654, doi:10.1109/NSSMIC.2007.4437102.
[11] L.A. Feldkamp, L.C. Davis, J.W. Kress, Practical cone-beam algorithm, J. Opt. Soc. Am. A1 (6) (1984) 612, http://josaa.osa.org/abstract.cfm?URI=josaa-1-6-612.
[12] C. NVIDA Corporation, Santa Clara, NVIDIA CUDA Compute Unified Device Architecture, Programming Guide, 2008, http://developer.download.nvidia.com/compute/cuda/1_1/NVIDIA_CUDA_Programming_Guide_1.1.pdf [Online; accessed 26 April 2008].
[13] D. Parker, Optimal short scan convolution reconstruction for fanbeam CT, Med. Phys. 9 (1982) 254–257.