High Resolution Iterative CT Reconstruction using Graphics Hardware

Benjamin Keck, Hannes G. Hofmann, Holger Scherl, Markus Kowarschik, and Joachim Hornegger

***Abstract*—Particular applications of computed tomography require high slice resolutions. The fastest iterative implementations on graphics cards use 3-D textures to exploit hardware-accelerated trilinear interpolation. However, the size of 3-D textures is subject to technical limitations, which makes them inapplicable here. Alternatively a 2-D texture array can be used instead of the 3-D texture as in early graphics implementations. The additional memory synchronizations cause a significant loss of performance.**

We utilize new features of the recently released CUDA 2.2 framework to improve the performance of the Simultaneous Algebraic Reconstruction Technique (SART). In this paper we present an enhanced version of our efficient implementation of the most time-consuming parts of the iterative reconstruction algorithm: forward- and back-projection.

We explain the required strategy to adapt the algorithm for the CUDA 2.2 features, in particular the usage of 2-D texture lookups from pitchlinear memory. Finally, we compare the result to our previous ones with respect to both reconstruction speed and technical limitations. The proposed strategy is a new balance between performance and limitations in resolution.

I. Introduction

In the field of medical imaging two major classes of CT reconstruction algorithms exist. First, the well known and wide-spread class of analytical methods, e.g., filtered back-projection (FBP) [1]. These algorithms are primarily used in commercial CT and C-arm CT due to the fact that a typical medical environment requires fast reconstructions in order to save valuable time.

The second class is characterized by an iterative update process, e.g., the Simultaneous Algebraic Reconstruction Technique (SART) [2]. Statistical iterative reconstruction algorithms, e.g., MLEM [3] or OSEM [4], have been used in molecular imaging scanners for a few years. They incorporate the modeling of physical effects and thus provide improved image quality for noisy data. However, the complexity of iterative methods is a multiple of the complexity of analytical methods.

Modern graphics cards offer sufficient compute power to overcome this drawback. We previously demonstrated that they can be used to accelerate analytical reconstruction algorithms [5]. Iterative reconstruction performance on graphics accelerators has also been evaluated since 1998 [6]. The first approaches employed shading languages (e.g., OpenGL) [7], [8]. In 2007, NVIDIA introduced the Compute Unified Device Architecture (CUDA), which simplifies the usage of GPUs for general-purpose computing tasks. CUDA popularized GPU-accelerated applications and is widely used in research, in particular for the acceleration of compute-intensive parts of iterative reconstruction algorithms [9], [10]. For this, 3-D textures are usually used in the forward-projection step of the SART algorithm [10] to benefit from the available hardware-accelerated interpolation. Further, Yan et al. [11] showed how to use 3-D textures in the back-projection step to accelerate the voxel access.

In some applications such as nondestructive testing, 3-D breast imaging and in research, the desirable slice resolution may exceed 2000×2000 pixels. In the past, memory limitations on the GPU have been the main reason to inhibit their application in these fields. Current high-end graphics cards offer sufficient memory, e.g., $4\,\text{GB}$ on the NVIDIA QuadroFX 5800 or NVIDIA Tesla C1060. However, the usage of 3-D textures for reconstruction of high resolution volumes is then still technically limited. While this constraint allows reconstruction of cubic volumes of up to $32\,\text{GB}$, slice resolutions larger than 2048 voxels are impossible, even if the volume fits into memory (e.g., $3072 \times 2048 \times 50 \approx 1.2\,\text{GB}$).

In this paper we present an approach that makes use of 2-D texture lookups from pitchlinear memory, introduced in CUDA 2.2. Compared to 3-D textures they are allowed to contain $32\ k \times 16\ k$ float elements. We use a modified version of our original approach [10] and compare the performance. Further, we will show an example of high resolution iterative reconstruction.

II. Method & Implementation

This work is based on our implementation of the SART method previously described in [10]. We applied technical changes, which will be also explained in detail after introduction of the method.

Originating from ART, SART is an iterative method to reconstruct a volumetric object from a sequence of alternating volume projections and corrective back-projections [6]. By measuring the difference between the current volume's forward-projection and the projections acquired by the scanner, a corrective image can be computed and distributed onto the volume grid in the back-projection step. Repeating this correction procedure makes the volume fit to almost all projections or at least minimizes the error in the case of convergence. Differing from the original ART, where the volume is corrected ray by ray, SART performs projection-wise corrections.

In theory, the system matrix for the SART defines the forward- and back-projection such that the methods are transposed to each other. In our case, we have used an unmatched forward-/back-projector pair for the iterative reconstruction, which has been investigated by Zeng et al. [12]. The au-

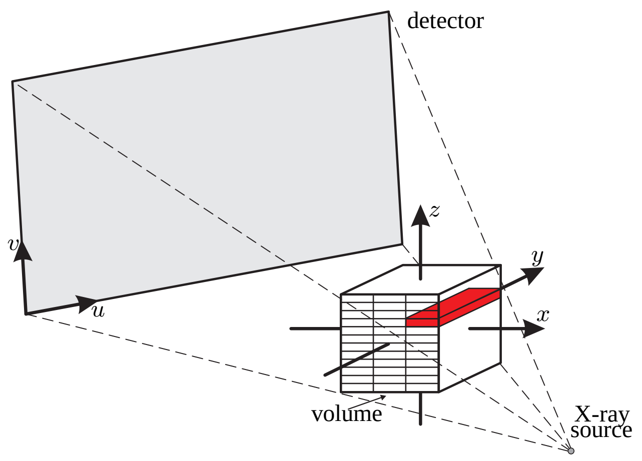

Fig. 1: General perspective geometry of an acquisition device (the v-axis and z-axis are not necessarily parallel) together with the parallelization strategy of our back-projection implementation on the GPU using CUDA. Tthe x-z plane is divided in several blocks to specify a grid configuration, and each thread of a corresponding block processes all voxels in y-direction.

thors demonstrate that an unmatched pair, where the forward-projector is ray-driven while the back-projector is voxel-driven, can effectively remove ring artifacts compared to a matched pair.

In general, iterative reconstruction algorithms consist of a forward-projection and a corrective back-projection step. We took the SART method for the performance evaluation of iterative reconstruction, using normal and high resolution reconstruction tasks.

A. Back-projection

The back-projection step is voxel driven. This requires the calculation of a matrix-vector product for each voxel in order to determine the corresponding corrective projection value. We invoke our back-projection kernel on the graphics device, where each thread of the kernel computes the back-projection of a certain column and volume slice (see figure 1). The algorithm is shown in algorithm 1.

Algorithm 1 Back-projection

```
//Host:
for all projections $P_j$ , $j = [0 \ldots N_p[$ do
  Call back-projection kernel;
  //Device:
  Compute voxel x and z coordinate
  for all voxels $(x, y, z)$, $y = [0 \ldots N_y[$ do
    Compute the coordinates $(u, v)$ of voxel $(x, y, z)$ in projection $P_j$
    Get the projection value at position (2-D texfetch)
    Add the weighted value to voxel
  end for
end for
```

B. Forward-projection / Corrective image computation

As introduced, SART performs projection-wise corrections of the current volume estimate. The corrective image is computed from the difference between the original projection and the appropriate simulated X-ray image of the current reconstruction estimate. All values in the corrective image are finally multiplied by the relaxation factor [2] before the back-projection step.

The previous implementation principles for CUDA 1.1 and CUDA 2.0 are illustrated in figures 2 and 3. The proposed CUDA 2.2 technique is shown in figure 5.

A realistic simulation of the X-ray imaging process can be achieved by a ray cast based forward-projection. Research on this grid-interpolated scheme, where the interpolation is performed using a trilinear filter and the integration according to the trapezoidal rule, showed that the root mean square (RMS) error is comparable to other popular interpolation and integration methods used in computed tomography [13]. This scheme is our first choice, because it can be ideally mapped to the GPU hardware including hardware-accelerated texture access.

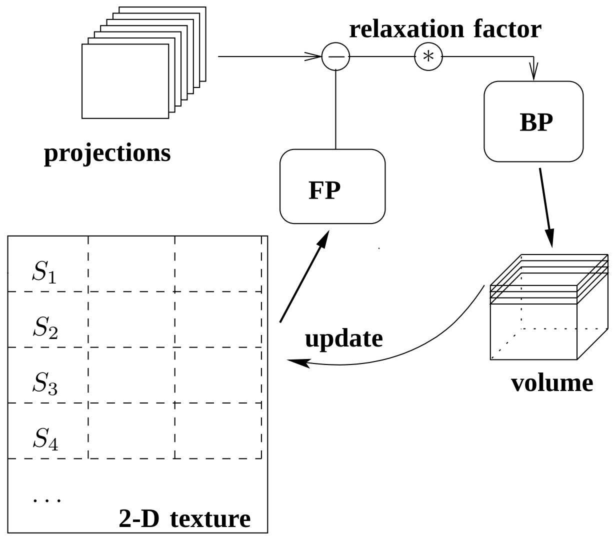

Fig. 2: CUDA 1.1 - GPU implementation principle: Volume represented in a 2-D texture by slices S_i is forward-projected (FP). After computing the corrective image and scaling with the relaxation factor, the back-projection (BP) distributes the result onto the volume. After performing an update the 2-D texture representation of the volume is equal to the volume.

The volumetric ray casting principle for the forward-projection step is illustrated in figure 4 and the algorithm is shown in algorithm 2. To determine the attenuation value of a certain pixel on the detector plane, a ray is drawn pointing from the X-ray source towards the detector pixel position. Afterwards, voxel attenuation coefficients inside the volume are sampled equidistantly along the ray. These sampling values add up to the respective attenuation value in the simulated projection. Similar to the back-projection step we use projection matrices to compute the resulting perspective projection.

To parallelize the forward-projection step, each thread of the kernel computes one corrective pixel of a projection. Analogous to the back-projection step we chose the grid configuration experimentally due to our results [14]. In the implemented kernel we compute the direction vector for a specific ray, which is the first step in the inner for loop in algorithm 2. Therefore we take the source position vector and the 3-D coordinate of the pixel position, compute the difference vector, and normalize it. The source position for all rays of a projection is obtained from the homogeneous projection matrix which is designed to project a 3-D point to the image plane. Galigekere et al. [15] have shown how to reproject using projection matrices.

Algorithm 2 Forward-projection with a ray casting algorithm

```
//Host:
for all projections $P_j$ , $j = [0 \dots N_p[$ do
  Compute source position out of projection matrix
  Compute inverted projection matrix
  Call back-projection kernel;
  //Device:
  for all rays $(u, v)$ inside the projection $P_j$ do
    Compute normalized ray direction
    //RAY CASTING
    Compute entrance and exit point of the ray
    Initialize the pixel value to zero
    if ray hits the volume then
      Set sample point to the entrance point
      while sample point is inside the volume do
        Get the sample value at current position {depending on CUDA 1.1, 2.0 and 2.2 strategy}
        Add up this value to the pixel value
        Compute new sample point for given step size
      end while
    end if
    Normalize pixel value to world coordinate system units
  end for
end for
```

In the kernel code, the inverse of the projection matrix is used to get the ray direction out of the pixel position in the projection image. The entrance and exit positions of the specific ray into the volume are calculated and stored as entrance and exit distances with respect to the source position. Between those points the volume is then sampled equidistantly. To get one sampling position, we take the entrance vector and add the product of direction vector, step size and the counter variable. The following sampling step itself proves to be crucial for the algorithm's efficiency. In order to get satisfying results, a sub-voxel sampling is required, which introduces a trilinear interpolation.

The global memory offers write access and thus has a higher latency. In contrast read-only texture memory has conspicuous low latency due to caching mechanisms and further offers hardware-accelerated interpolation. In CUDA 1.1 the computation of each sample point intensity is a critical issue since support for 3-D textures is not provided. In consequence, a workaround had to be applied that used just the bilinear interpolation capability of the GPU. The kernel computes a linear interpolation between stacked 2-D texture slices (S_i) (see figure 2). Therefore, two values are fetched from proximate stack slices with hardware-accelerated bilinear interpolation and afterwards linearly interpolated in software. These sampling steps are substituted by only one hardware-accelerated 3-D texture fetch in CUDA 2.0. Since texture memory is read-only, the back-projection updates the original volume data kept in global memory. The volume-representing texture has to be synchronized with the updated estimate (figure 2 and 3). Such a synchronization is referred to as a texture update.

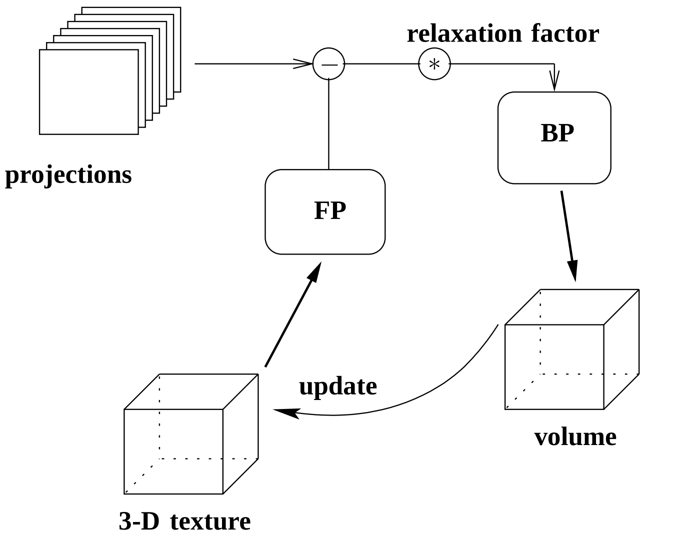

Fig. 3: CUDA 2.0 - GPU implementation principle: Volume represented in a 3-D texture is forward-projected (FP). After computing the corrective image and scaling with the relaxation factor, the back-projection (BP) distributes the result onto the volume. After performing an update the 3-D texture representation of the volume is equal to the volume.

So far, the difference in volume representation for the corrective image computation led to two main principles of SART implementation [10] using CUDA shown in figure 2 for CUDA 1.1 and figure 3 using CUDA 2.0. After all corrective images have been computed and back-projected for all iterations the reconstruction finishes by transferring the volume to the host system memory.

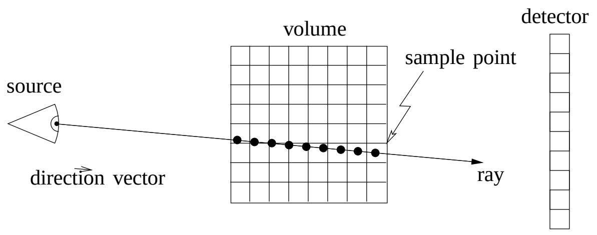

Fig. 4: Ray casting principle with an equidistant sample step size.

C. High resolution adaption

Due to the size constraints of 3-D textures they are unsuitable for high resolution reconstructions. In April 2009, NVIDIA updated the CUDA framework to version 2.2 and introduced 2-D texture lookups from pitchlinear memory. They support hardware-accelerated bilinear interpolation from writable global memory. However, the memory consistency during a kernel execution is not ensured. A 2-D texture is also limited in size, but the restriction is relaxed: $32\ k \times 16\ k$ float elements (overall $2\,\mathrm{GB}$).

Therefore we adapt the voxel-driven back-projection (see section II-A). To use 2-D textures we use a method similar to our approach shown in figure 2. The implementation is identical, except for an adapted voxel memory address computation due to the new memory layout, shown in figure 5.

Using writable 2-D texture lookups from pitchlinear memory for the volume representation, we get rid of the time consuming synchronization of the volume copies in writable global memory and the read-only texture arrays.

Accessing the sample value in the innermost loop of algorithm 2 was replaced by a linear interpolation between two 2-D texture values that are both bilinearly interpolated by hardware. The volume is represented as a stack of slices (S) organized in the 2-D texture, as is illustrated in figure 5.

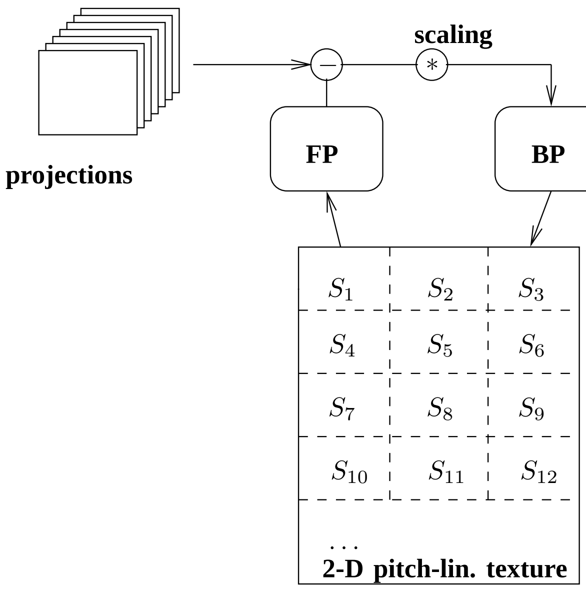

Fig. 5: CUDA 2.2 - GPU implementation principle: Volume represented in a 2-D texture by slices S_i is forward-projected (FP). After computing the corrective image, the back-projection (BP) distributes the result onto the volume-representing 2-D pitch-linear texture.

III. Results

In order to evaluate the performance of the different implementations we extend our experiment from [10], where we used 228 projections representing a short-scan from a C-arm CT system to perform iterative reconstruction with a projection size of 256×128 pixels. The reconstruction yields a $512 \times 512 \times 350$ volume. Table I shows the achieved performance for the different approaches of GPU-based SART reconstruction, including the benefits and technical limitations. Using NVIDIA's most recent GPU generation, Tesla C1060, the reconstruction time for our new approach is further reduced from 1488 to 955 seconds[1].

Volume	$512 \times 512 \times 350$ voxels		
Hardware	QuadroFX 5600		
volume representation (FP)	2-D texturearray	3-D texturearray	2-D pitch-linear texture
volume representation (BP)	global memory (linear)	global memory (linear)	global memory (spec. arrangem.)
device memory required [MB]	700	700	350
volume synch. needed	YES	YES	NO
required CUDA version	$\geq$ CUDA 1.1	$\geq$ CUDA 2.0	$\geq$ CUDA 2.2
SART performance in [s][1]	4234	844	1488

TABLE I: Comparison of iterative reconstruction times in seconds (for 20 iterations each).

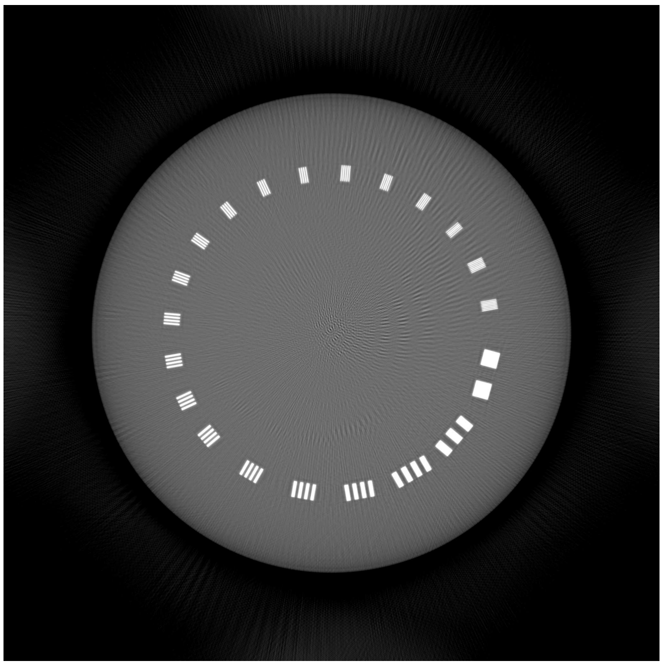

Fig. 6: Iterative reconstruction of the Catphan CTP528 phantom using simulated projections.

In order to assess image quality of high resolution iterative CT reconstruction, we utilized simulated phantom projections of the Catphan CTP528 phantom, generated with DRASIM. We generated 400 projections of 1024×128 pixels with a pixel size of 0.3×0.7 millimeters. An iterative reconstruction of the phantom is shown in figure 6, where all 21 line pairs are

[1]preliminary results

Hardware	Tesla C1060			
volume resolution	512×512 $\times 100$	1024×1024 $\times 100$	2048×2048 $\times 100$	3072×2048 $\times 50$
voxel size in mm	0.4×0.4 $\times 0.1$	0.2×0.2 $\times 0.1$	0.1×0.1 $\times 0.1$	0.075×0.1 $\times 0.1$
device memory required [MB]	100	400	1600	1200
SART performance in [s][1]	1166	2407	11353	4951

TABLE II: Iterative reconstruction setups for the Catphan CTP528 phantom using NVIDIA's Tesla C1060 and our new approach. The performance is measured for 20 iterations each.

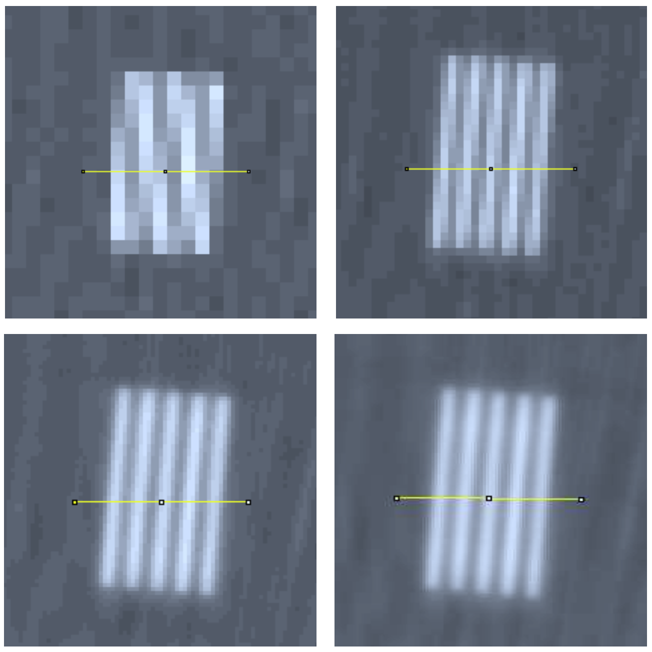

Fig. 7: Iterative reconstructions details of the 16-th line pair of the Catphan CTP528 for different slice resolutions: top-left 512^2 pixels, top-right 1024^2 pixels, bottom-left 2048^2 pixels, the highest resolution 3072×2048 on the bottom right. The visualized yellow line across the elements illustrates the line used for the line profiles.

visible. The different resolution and reconstruction setups are detailed in table II including the performance measurements. Due to the 2 GB memory limitation of 2-D texture lookups from pitchlinear memory, the number of slices for the highest resolution of 3072×2048 pixels is reduced to 50 slices.

To proof the expectable increase in image quality, the reconstruction results of the 16-th line pair of the phantom is illustrated for each setup in figure 7. Finally, an attenuation coefficient comparison for the lines (marked yellow) is shown in the line profiles (see figure 8).

IV. Conclusions

We have presented an enhanced GPU-accelerated SART reconstruction for high resolution volumes. We have shown the advantage of using 2-D texture lookups from pitchlinear memory to overcome the 3-D texture size limitation for volume representation. The missing hardware-accelerated trilinear interpolation is substituted by linear interpolation in software between two hardware-accelerated bilinear interpolations. Although this approach decreases reconstruction speed as anticipated, no volume synchronization procedure and only half the memory (due to writable texture) on the graphics card are necessary compared to our previous implementations [10]. The expectable increase in image quality for a high resolution reconstruction was validated by the reconstruction and comparison of the high resolution phantom Catphan CTP528.

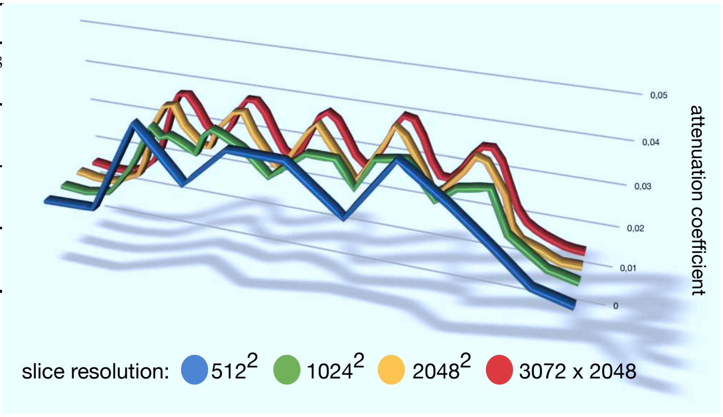

Fig. 8: Compared line profiles of the line across the 16-th line pair of the different reconstructions of the Catphan CTP528 phantom.

Acknowledgments

This work was supported by Siemens AG, Imaging & IT Division, Medical Electronics & Imaging Solutions.

References

[1] H. Turbell, "Cone-beam reconstruction using filtered backprojection," Ph.D. dissertation, Dept. Elect. Eng., Linköping University, Sweden, Mar. 2001.

[2] A. Andersen and A. Kak, "Simultaneous Algebraic Reconstruction Technique (SART): A superior implementation of the ART algorithm," *Ultrasonic Imaging*, vol. 6, no. 1, pp. 81–94, January 1984. [Online]. Available: http://www.sciencedirect.com/science/article/B6WXM-4C52HCN-25/1/a52bef081f65f9da0e626f2ec5cd00ba

[3] L. A. Shepp and Y. Vardi, "Maximum likelihood reconstruction for emission tomography," *Medical Imaging, IEEE Transactions on*, vol. 1, no. 2, pp. 113–122, Oct. 1982.

[4] H. Hudson and R. Larkin, "Accelerated image reconstruction using ordered subsets of projection data," *Medical Imaging, IEEE Transactions on*, vol. 13, no. 4, pp. 601–609, Dec 1994.

[5] H. Scherl, B. Keck, M. Kowarschik, and J. Hornegger, "Fast GPU-Based CT Reconstruction using the Common Unified Device Architecture (CUDA)," in *Nuclear Science Symposium, Medical Imaging Conference 2007*, E. C. Frey, Ed., vol. 6, Honolulu, Oct. 2007, pp. 4464–4466.

[6] K. Mueller and R. Yagel, "Rapid 3D cone-beam reconstruction with the Algebraic Reconstruction Technique (ART) by utilizing texture mapping graphics hardware," *Nuclear Science Symposium, 1998. Conference Record.*, vol. 3, pp. 1552–1559, Nov. 1998.

[7] K. Mueller, F. Xu, and N. Neophytou, "Why do Commodity Graphics Hardware Boards (GPUs) work so well for acceleration of Computed Tomography?" in *SPIE Electronic Imaging Conference*, vol. 6498, San Diego, Feb. 2007, (Keynote, Computational Imaging V).

[8] M. Churchill, "Hardware-accelerated cone-beam reconstruction on a mobile C-arm," in *Proceedings of SPIE*, J. Hsieh and M. Flynn, Eds., vol. 6510, San Diego, Feb. 2007, p. 65105S.

[9] M. Knaup, S. Steckmann, and M. Kachelriess, “Gpu-based parallel-beam and cone-beam forward- and backprojection using cuda,” in *Nuclear Science Symposium Conference Record, 2008. NSS ’08. IEEE*, Oct. 2008, pp. 5153–5157.

[10] B. Keck, H. Hofmann, H. Scherl, M. Kowarschik, and J. Hornegger, “GPU-accelerated SART reconstruction using the CUDA programming environment,” in *Proceedings of SPIE*, E. Samei and J. Hsieh, Eds., vol. 7258, Lake Buena Vista, 2009.

[11] G. Yan, S. Zhu, Y. Dai, and C. Qin, “Fast cone-beam CT image reconstruction using GPU hardware,” *Journal of X-Ray Science and Technology*, vol. 16, pp. 225–234, Jul. 2008.

[12] G. Zeng and G. Gullberg, “Unmatched projector/backprojector pairs in an iterative reconstruction algorithm,” *IEEE Transactions on Medical Imaging*, vol. 19, no. 5, pp. 548–555, May 2000.

[13] F. Xu and K. Mueller, “A comparative study of popular interpolation and integration methods for use in computed tomography,” *Biomedical Imaging: Nano to Macro, 2006. 3rd IEEE International Symposium on*, pp. 1252–1255, April 2006.

[14] A. Weinlich, B. Keck, H. Scherl, M. Korwarschik, and J. Hornegger, “Comparison of High-Speed Ray Casting on GPU using CUDA and OpenGL,” in *High-performance and Hardware-aware Computing (HipHaC 2008)*, R. Buchty and J.-P. Weiss, Eds., Como (Italy), 2008, pp. 25–30.

[15] D. H. R. Galigekere, K. Wiesent, “Cone-Beam Reprojection Using Projection-Matrices,” *IEEE Transactions on Medical Imaging*, vol. 22, no. 10, pp. 1202–1213, 2003.