GPU-based cone-beam CT reconstruction with extended FOV

Tamás Huszár[1], Gábor Jakab[1 2], and Attila Rácz[1]

[1] Mediso Medical Equipment Developing and Service Ltd.
Budapest, 1022 Hungary
http://www.mediso.hu, research@mediso.hu
[2] Budapest University of Technology and Economics
Department of Control Engineering and Information Technology
Budapest, 1117 Hungary
http://www.iit.bme.hu

Abstract. Modern graphics processing units (GPUs) offer high level of parallelism and huge computational power which makes them suitable for complex calculations and processing of voxel data, including Computed Tomography (CT) image reconstruction. 3D reconstruction has higher computational cost but superior image quality than 2D methods. Using a fast GPU-based algorithm, these 3D algorithms became fast enough to be used in commercial CT equipment.
In our paper we present our extended CT reconstruction framework with 3D reconstruction. We implement the Tang and xFDK algorithms. These are based on the Feldkamp-Davis-Kress (FDK) method and extend the field of view as much as possible. We evaluate speed and image quality using real measurements with Mediso CT equipment.

1 Introduction

The 2D filtered backprojection (FBP) [1][2] is still the industry standard for CT image reconstruction used in human cameras. FBP provides acceptable image quality with well-known artifacts, including streaks and rings [3][4]. However with the advancement of processing hardware, 3D reconstruction is getting more attention and becomes an alternative to 2D methods.

In our previous paper we presented our real-time, GPU based reconstruction framework capable of 2D FBP and MLTR[1] reconstruction for the Mediso AnyScan CT (Mediso Ltd.) [5]. Our GPU implementations use CUDA[2] and support NVIDIA[3] graphics hardware. We also developed a GPU based cone-beam CT reconstruction for in vivo imaging, used in Mediso NanoPET/CT (Mediso Ltd.) [6][7][8].

We have extended our existing reconstruction framework to accommodate the native 3D cone-beam geometry of the AnyScan CT. This enabled us to

[1] Maximum Likelihood for Transmission tomography
[2] Compute Unified Device Architecture - http://www.nvidia.com/object/cuda_home_new.html
[3] http://www.nvidia.com

remove 2D rebinning from the reconstruction pipeline. The whole reconstruction is running on the GPU, minimizing unnecessary data transfers between system and device memory.

However, using 3D geometry introduces new problems and artifacts which result uneven image quality along the axial direction. Reconstructed slices further away from the x-ray source suffer increased quality loss. The original FDK algorithm [9] suffers greatly from these artifacts.

In human CT development, one desirable goal is to minimize the dose usage while maintaining high image quality. To achieve this we extend the reconstruction area by reducing cone-beam artifacts. We implemented the extended FDK (xFDK) [10] and Tangs method [11], compared them to the original FDK and evaluated their practical use in human CT.

2 Cone-beam CT reconstruction

2.1 Overview

Third generation CT equipment have large detectors with multiple rows. For example the Anyscan CT has 16 rows and it is able to reconstruct 16 slices at once. Its detector has a cylindrical surface, hence we refer to it as cylindrical cone-beam geometry. In many applications, the detector has a flat surface, but this yields only minor differences in most reconstruction algorithms.

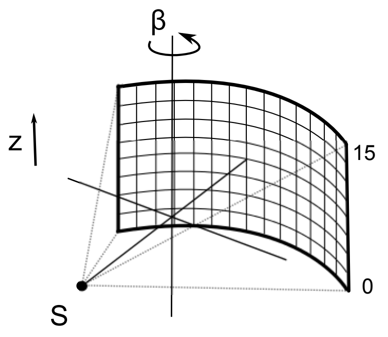


Fig. 1. Cylindrical cone-beam geometry. S is the position of the source, β is the view or gantry angle. The numbers next to the detector indicate the indexing of the 16 detector rows (the indices start from 0)

We further differentiate 3D reconstruction based on source trajectory. The two simple trajectories used in commercial CT are called circular and helical. In our article, we discuss the circular (or axial) case only. The gantry is rotated with a constant speed and the patient table remains stationary. In a coordinate system fixed to the table, the trajectory is a circle.

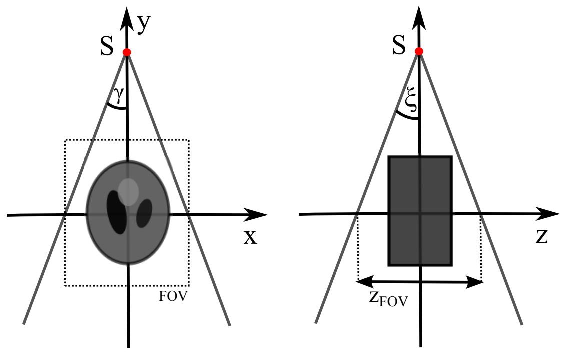


Fig. 2. Axial and sagittal view of cone-beam geometry. The coordinate system is fixed to the gantry. S is the position of the source, FOV is the axial, Z_{FOV} is the transaxial field of view. γ and ξ mark the fan and cone angles.

The axial cone-beam geometry is shown in Figure 1. The position of the gantry is measured as the view angle β. The gantry rotates along the z axis. Axial and sagittal views are shown in Figure 2. Reconstructed slices correspond to detector rows marked 1..16. The source position on the z axis is between slice 8 and 9. Slices 1 and 16 are the furthest away from the center. The fan and the cone angles are marked as γ and ξ, respectively. Please note that when we talk about the fan or cone angle of a CT system, we usually mean the angle between the two outermost detector elements along the corresponding axis. This is usually the double of the maximal γ or ξ angles used in this paper.

2.2 FDK method

Exact reconstruction means, that the difference between the reconstructed and the original object can be arbitrarily small by increasing detector and voxel resolution and the number of views. The theoretical background of 3D cone-beam reconstruction is derived from the 3D radon transform. Plane integrals are calculated along parallel rays passing through the object. According to the Tuy-Smith sufficiency condition [12], exact reconstruction is only possible if all planes going through the object intersect the source trajectory at least once. For circular reconstruction, this condition is obviously not satisfied, except in the central plane.

This conclusion leads us to approximate methods, namely the FDK method by Feldkamp, Davis and Kress [9]. This method originates from the 2D FBP algorithm for fan-beam geometry. It basically regards the 3D projection data as a series of 2D fan-beam projections. The projections are filtered line by line in 2D and backprojected onto the volume.

Similar to fan-beam geometry, rays with a higher cone angle traverse a longer path in the object. This is approximated by multiplying the projection data with

$\cos(\xi)$, where ξ is the cone angle of the given detector element. For an object which is homogeneous in the z direction, this correction gives exact results.

The FDK method was originally intended for flat detector, however a slightly modified version called C-FDK exists for cylindrical detectors [13].

Algorithm 1. The main steps of the FDK method are the following.
Step 1. Calculate filtered and corrected sinogram data:

$$\hat{p}(\beta, \gamma, q) = (\cos(\gamma)\cos(\xi)p(\beta, \gamma, q)) * g(\gamma)) \tag{1}$$

Step 2. Calculate voxel values (backprojection):

$$f(x, y, z) = \int_0^{2\pi} \frac{R^2}{L(x, y, \beta)^2} \hat{p}(\beta, \gamma, q)\delta\beta \tag{2}$$

Algorithm 1 shows the filtering and the backprojection steps of FDK. The input projections are marked as $p(\beta, \gamma, q)$, where β and γ are the view and fan angles respectively. Parameter q addresses the detector along the z direction (see Fig. 1). The filtered sinogram is noted as $\hat{p}$. The filter operator is $f(\gamma)$.

The voxel values $f(x, y, z)$ are calculated using equation (2). In practice, the integral is approximated as a summation for all available views. Parameter R is the distance of the source from the rotation center. L is the distance of the voxel from the center.

2.3 Extending reconstruction area

An important practical question is the maximal reconstructible area using a given detector and source trajectory. In parallel geometry, according to the original FBP algorithm, a slice is reconstructible if there are projections available from at least 180°.

In fan-beam geometry, this value is

$$\pi + \gamma/2. \tag{3}$$

Equation 3 is derived from the the rebinning equations between the fan and parallel geometries:

$$\theta = \beta + \gamma, \qquad t = R \sin\gamma \tag{4}$$

The area where voxels are reconstructed is called the field of view (FOV). The axial and transaxial field of view is shown as FOV and Z_{FOV} in Figure 2. This area is usually selected by the user, but limited by the detector geometry. Inside the FOV reside voxels projected onto the detector in every view. We also say these voxels are fully illuminated or visible under 360°. In the axial plane, this area is a circle around the center of rotation.

A full-scan is called when view data from 360° are available. If voxels are visible from less than 360°, we talk about a short-scan.

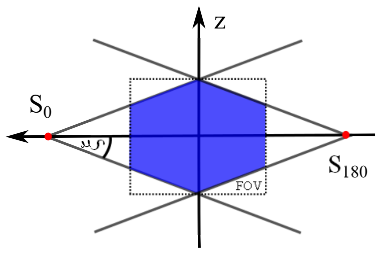
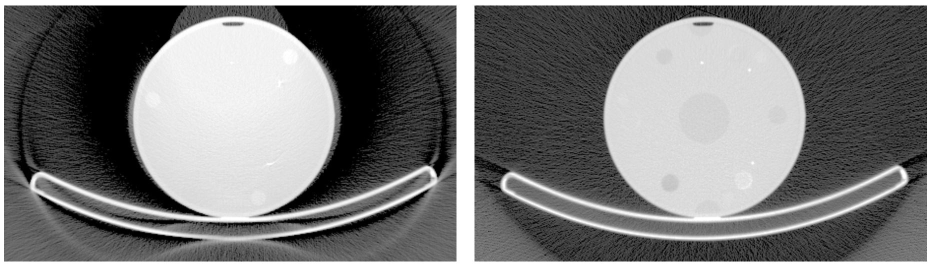


Fig. 3. The double cone in 2D (the highlighted area) where data under 360° are available. S and S_{180} are the source positions for view angles 0° and 180°. The dotted rectangle represent the field of view. Reconstructed slices are located evenly in this area.

For each ray going through a given voxel, there is a complementary (or conjugate) ray intersecting that same voxel. Complementary ray pairs lie on the same line in the axial plane but have opposite direction. They belong to different detector positions. In parallel geometry, the angle between a ray and its complementary are exactly 180°.

Short-scan algorithms do not have data from all views. This missing data can be recovered from complementary rays. If all view data can be recovered, than the short-scan yields the same result as a full-scan. Otherwise artifacts appear due to missing data. In a full-scan, each voxel is projected to each view. Short-scan algorithms must use weighting to compensate for the different number of rays going through each voxel.



(a) slice #1 (b) slice #8

Fig. 4. Reconstructed slices of the Catphan phantom using FDK with normalization. Figure 4a shows the outermost slice (#1), 4b shows the slice closest to the center ((#8)). Heavy cone-beam artifacts are visible on the first image. Notice the artifacts below the patient table and the dark aura around the phantom.

The original FDK algorithm is a full-scan algorithm. In Figure 3 the highlighted area contains voxels visible from all source positions. If the source moves on a circle, we get a double cone, which is the intersection of all rotated view cones. Voxels are fully illuminated inside this volume. Outside of the cone, various artifacts appear. This means, that the reconstructible area is less than the depth of the detector along the z axis. In case of human cameras, one may need a detector twice as large as the desired FOV.

However, given the above observations, it is obvious that less than 360° view coverage is necessary for successful reconstruction. Using partial scan weighting, voxels visible from partial view coverage can be utilized.

A naïve weighting is achieved by counting the detector hits for every voxel and divide the voxel by this value. The idea of this normalization comes from equation (2). For voxels fully illuminated the integration goes from $0..2\pi$. When we discretize this integral, the voxel values are summed for every view and divided by 2π and the number of views. However, when view data is only available for a limited range, we only divide by the number of views contributing to that voxel.

2.4 Tang's method

Another problem of FDK is related to large cone angles. Consider a slice on the edge of the transaxial field of view. A ray, with a relatively large cone angle is travelling through multiple image slices. When a voxel inside our slice is backprojected along the aforementioned ray, the value read from the detector originates from a larger z range than our slice. This makes the image blurred along the z axis.

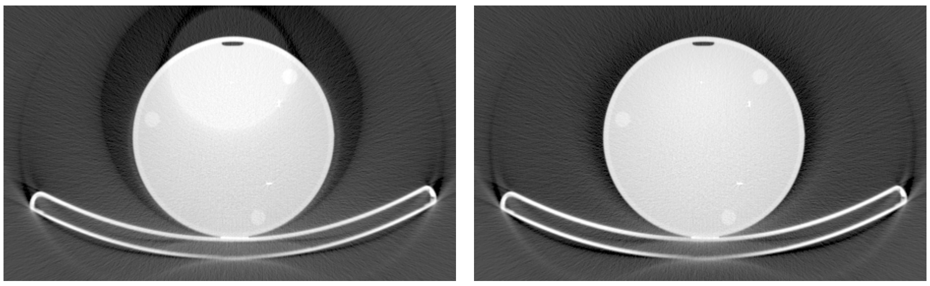

(a) $k = 0.5$ (b) $k = 2$

Fig. 5. Reconstructed slices using Tang´s method. Figure 5a shows $k = 0.5$, where most of the artifacts are visible. Figure 5b shows $k = 2$ with suppressed artifacts in the center of the image.

The Tang´s method [11] examines complementary rays. It works with cone-parallel geometry. The rays are first rebinned according to the 2D rebinning equations (4), while the z coordinates remain untouched. If we disregard the z axis, complementary and normal rays intersecting a voxel bear exactly the same information. However, in 3D, these rays have a different cone angle, depending of the voxel´s position. For example, a voxel, on the center of rotation, has rays with the same cone angle. A voxel further from the center of rotation has a ray with smaller cone angle and a ray with a larger cone angle. It is clear, that the voxel with the larger cone angle carry more invalid information, originated from the outside of the slice.

The idea behind Tang´s method is to create a weighing based on the cone angle of the ray. The method favors rays with small cone angles. If a ray has no valid complementary ray (because the complementary ray would fall outside of the detector) then it has a weight of 1. Otherwise the weight goes from 0 to 1 according to equation below. The method discards projection values so it increases noise.

$$w_{Tang}(\xi, l) = \frac{\tan^{k|l|}(|\xi_c|)}{\tan^{k|l|}(|\xi_c|) + \tan^{k|l|}(|\xi|)} \tag{5}$$

The k is a user defined parameter, l is the distance from the source along the z axis, ξ is the cone angle of the given voxel. ξ_c is the cone angle of the complementary ray.

The value $k|l|$ defines the sharpness of the transition between the full and zero weights. The $k|l| = 0$ translates back to the original FDK as all the weights are 0.5. When $k|l| = 1$ we get a smooth transition and good balance between artifact reduction and noise. Larger values suppress more artifacts but increase noise further.

2.5 xFDK

The extended FDK (xFDK) method [10] is very similar to Tang´s method. It uses weighing and discards projection values, however the weighting is done in voxel space. The idea behind the method is that voxels seen by less than 360° must be weighted by a partial scan weight.

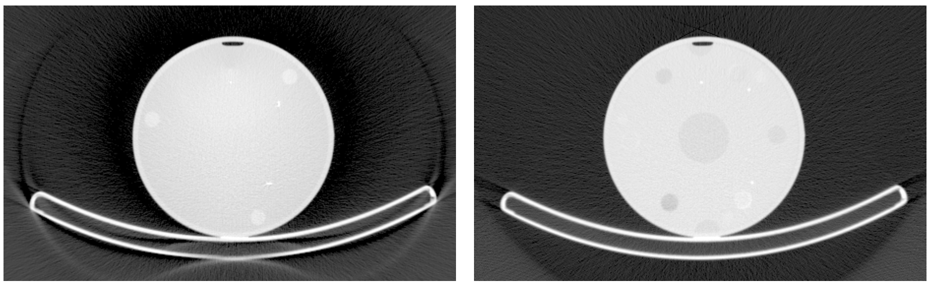

(a) slice #1 (b) slice #8

Fig. 6. Reconstructed slice using xFDK method. Artifacts inside the phantom are suppressed. However they are still visible near the edge of the slice (e.g. at the patient bed).

The algorithm finds the illumination range of each voxel. The data is first rebinned to cone-parallel geometry. The illumination range is the interval of paralell view angles, where the voxel´s projection is visible on the detector. The voxels are transformed to polar coordinates, ϕ is the polar angle of the voxel. According to [10], the voxel is fully illuminated when equation (6) and (7) are true.

$$|\beta - \phi| \leq \frac{\pi}{2} + \arcsin \frac{R_F - cz}{r} \tag{6}$$

$$c = \cot \gamma_{max} \tag{7}$$

In equation (6), R_F is the radius of the axial field of view. The value cz is defined in eq. (7) and called demarcation line, which separates the illuminated and not illuminated areas of the volume.

Now we can calculate the γ fan angle for a given β and ϕ:

$$\gamma = -\arctan \frac{r \sin(\beta - \phi)}{R_f + r\cos(\beta - \phi)]} \tag{8}$$

By using equations (4)(6)(7)(8), we can derive a condition for θ, which gives the illumination range:

$$[\phi - \pi/2 - \varDelta\theta, \phi + \pi/2 + \varDelta\theta] \tag{9}$$

For θ values outside this range, a weight of 0 is used. Inside the range the weight should be 2. This means that visible voxels are double weighted and voxels not visible are ignored. Now for voxels at the edge of the valid θ range, a smooth sinusoid transition is introduced (see eq. 10). The transition is $\varDelta\theta$ wide. For the exact method of weighting see table 1.

$$s(t) = \sin(\pi t/2) \tag{10}$$

The weighting uses the observation, that voxels at the end of the illumination range have larger cone angles as they are located closer to the detector and their projections are towards the edge of the detector. Voxels in the middle of the θ range are further away from the detector and have smaller cone angles.

Table 1. xFDK partial scan weights. $s(t)$ is the smoothing function.

w_{PS}	Condition
0	$\theta < \phi - \pi/2 - \varDelta\theta$
$1 + s(\frac{\theta-\phi-\pi/2}{\varDelta\theta})$	$\theta < \phi - \pi/2 + \varDelta\theta$
2	$\theta < \phi + \pi/2 - \varDelta\theta$
$1 - s(\frac{\theta-\phi-\pi/2}{\varDelta\theta})$	$\theta < \phi + \pi/2 + \varDelta\theta$
0	otherwise

Note, that the above weighting is only applied to voxels outside the fully illuminated range. Inside that range, a constant 0.5 full-scan weight is applied. Between the two weights, another smooth transition is introduced based on the voxels location in the axial plane.

One of the advantages of the xFDK method is that no user dependent parameter is introduced. However one can replace the $s()$ function which has a similar effect as the k parameter of the Tang´s algorithm.

3 Implementation

The reconstruction framework is written in C++ and CUDA. We use CUDA 4.2 and our target architecture is CM 2.0. The framework consists of device data structures, samplers, error handling, wrappers for kernel launches and a collection of reconstruction classes and kernels.

The input projections are wrapped into a custom datastructure containing geometry descriptors and view headers. This data is dumped from the CT device or generated by our forward projector. The forward projector itself works with raw phantom data generated using Matlab[4].

Frequency domain filtering is done on the device. FFT is calculated using the NVIDIA CUDA Fast Fourier Transform library[5] (cuFFT). The framework supports classical filters like ramp or cosine and is able to load custom filters from an external file.

The voxel driven backprojector uses a matrix transformation to calculate the voxels position on the detector. It calculates the cone and fan angle of the given voxel, which is then transformed to texture coordinates. The detector model is represented by a class with device members and passed to the reconstruction engine as a template parameter, so it can be easily replaced.

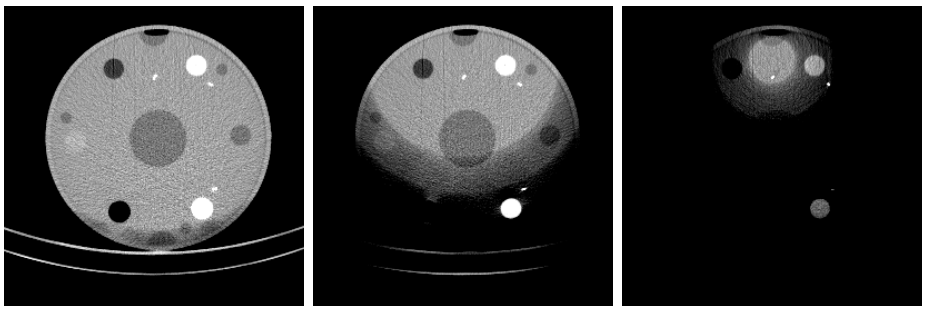
The pixel driven forward and backprojector use a simple 3D line drawing algorithm. It calculates ray-voxel intersection lengths and weights voxel or detector values along the ray. The ray directions are calculated by transforming the detector pixels to a virtual detector in voxel space. The whole projection algorithm runs on the GPU.

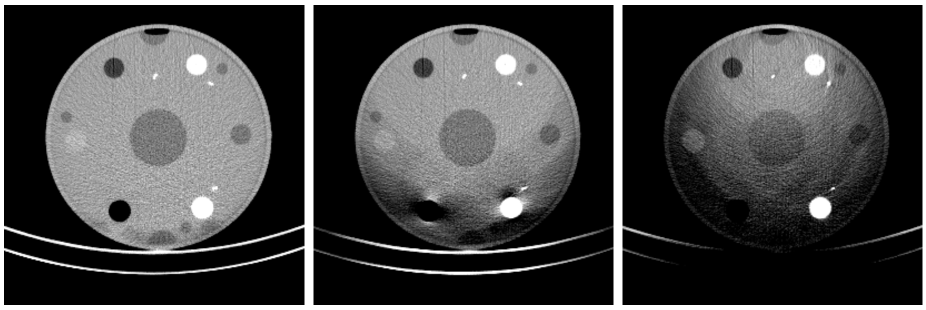
4 Measurements

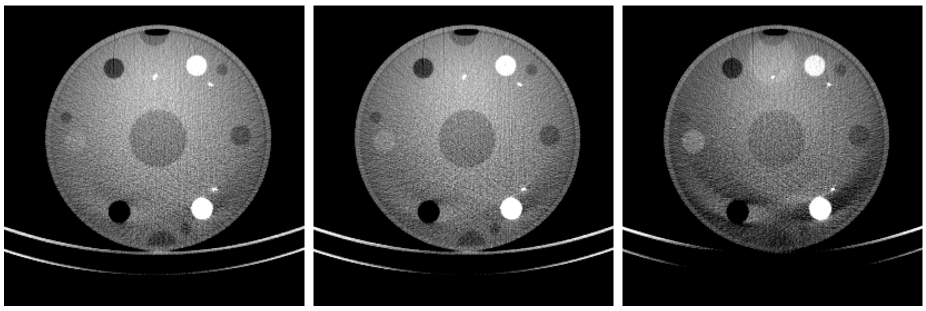
4.1 Materials and methods

To evaluate the accuracy of a CT system, a comprehensive test study is needed. We used mathematical and physical phantoms along with real clinical scans for testing.

[4] http://www.mathworks.com/products/matlab/
[5] https://developer.nvidia.com/cufft


(a) FDK #14 (b) FDK #15 (c) FDK #16


(d) Norm. FDK #14 (e) Norm. FDK #15 (f) Norm. FDK #16


(g) Tang #14 (h) Tang #15 (i) Tang #16

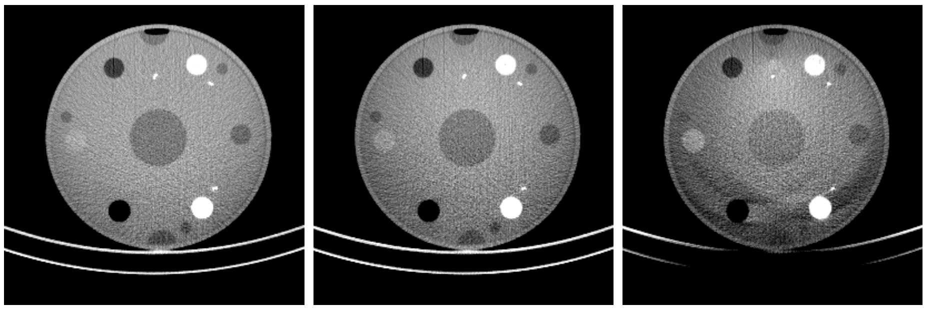
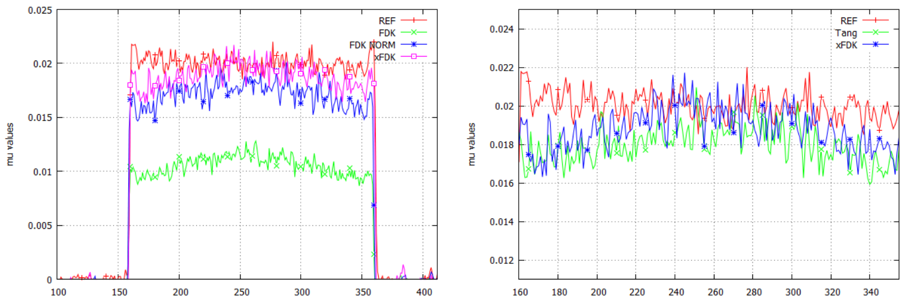

(j) xFDK #14 (k) xFDK #15 (l) xFDK #16

Fig. 7. Comparison of FDK, normalized FDK, Tang and xFDK methods (from top to bottom). The slices presented in each row mostly contain partially illuminated voxels. All images have the same window and level setting.

We created simulated phantoms to ensure the consistency between our back and forward projector. These phantoms were generated with Matlab and forward projected with our GPU based forward projector. We also used real measurements of the The Catphan 600[6] phantom, which is a widely accepted tool, designed for characterizing CT performance.

4.2 Artifact analysis

We used the Catphan 600 phantom to evaluate artifacts. The original FDK method has artifacts on slices outside of the fully illuminated area. These artifacts include object breaking up (patient table in Figure 4a), blurring and different high and low intensity patches (see Figure 7b,c and Figure 7e) and rapid intensity drop near the edges (Figure 8a). The fully illuminated area shows no visible artifacts (Fig. 4b).



(a) (b)

Fig. 8. Intensity drop near the edge of the reconstructed phantom using various methods. Fig. 8a: FDK, normalized FDK, xFDK Fig. 8b: Tang's method, xFDK. The algorithms are compared to a reference slice inside the fully illuminated area. In Fig. 8b the values are truncated to the center of the phantom for better visibility.

Figure 7 shows a comparison of the different algorithms. The first row (in Fig. 7a-c) shows the original FDK algorithm. A huge intensity drop is visible on the two outermost slices (Fig. 7b,c). The second row (in Fig. 7d-f) shows the effect of the normalization. The intensity drop is less prominent. The last two rows show the results of the Tang (g-i) and xFDK (j-l) methods.

Extended algorithms eliminates all artifacts on slice 14 as seen in Figure 7a,g,j. The xFDK method has the less visible artifacts on slice 15 (Fig. 7k). Slice 16, which is the furthest from the center shows visible artifacts using all methods, however they are less prominent using the extended methods.

The intensity drop is visualized as a line plot across the outermost slice in Figure 8a. By FDK the attenuation (μ) values drop by 50%, from 0.02 to 0.01.

[6] The Phantom Laboratories Inc.

Using normalization, the drop is less, the values stay above 0.015. xFDK eliminates the drop near the center of the image. The xFDK and Tang's method are compared in Figure 8b. It is seen that the xFDK method is better in eliminating the intensity drop artifacts. Note that the drop almost completely disappeared on all but the two outermost slices.

Both of the extended algorithms suppress artifacts near the center of the slice (see the last two rows in Figure 7). This is happening because more views are available for voxels on that area. Near the edge of the slice, the algorithms are unable to suppress artifacts as the ray illumination range of the voxels are less than specified in equation (3).

As a conclusion to this section we saw that both extended methods efficiently reduce artifacts on all but the two outermost slices. The fully illuminated area covers the 8 middle slices out of the total 16 (Figure 3). These 8 slices are reconstructible using FDK. Using the extended methods, the artifact free reconstruction of 3 more slices on each side is possible, resulting a total number of 14 slices.

4.3 Speed and running time

For slices inside the fully illumination range it is not necessary to use the extended algorithms. It is possible to use the original FDK method for these slices. The extended weights only need to be calculated outside the double cone shown in Figure 3, which reduces reconstruction time.

Table 2. Total running time of the backprojection kernel for different methods. The tests were executed on a Geforce GTX 580 graphics card.

method	1 sample (s)	8 samples (s)
FDK	1.42	5.33
Tang	1.94	9.74
xFDK	2.45	12.61
Tang 1-4,13-16	1.75	7.85
xFDK 1-4,13-16	1.98	9.44
Tang (double))	10.29	79.27

The execution time of various methods are compared in Table 2. The table shows the summed amount of backprojection kernel execution time for 960 views. Using 8 samples per voxel the Tang and the xFDK methods are respectively 1.8 and 2.5 times slower than FDK. By calculating weights only for slices $1-4$ and $13-16$, these numbers are reduced to about 1.5 and 1.8.

Note the last row in Table 2 which shows Tang's method with parameter $k|l| = 5$. Equation 5 requires double precision when the $k|l|$ parameter goes above 2.2. Normally, all calculations in this article are implemented using single float precision, because of the performance impact of doubles on the GPU. The use of doubles in the Tang algorithm introduces a serious 5× performance penalty, making the method unfavorable. If performance is an issue, then Tang's method with $k|l| = 2$ can be used, which is faster than xFDK. However the artifact suppression is less sufficient in this case. Where not noted differently, all results of Tang's method have parameter $k|l| = 2$.

4.4 Noise measurements

We made noise measurements using simulated water phantoms and scans of the Catphan phantom. In this section, we present our measurements using the Catphan phantom. We made measurements on multiple slices of the phantom to observe the effect of different artifacts. We used 8 samples per voxel for all results in this section.

Table 3. Noise measurements on slice #1, which is the outermost slice.

Slice 1	Air		Water	
Method	Mean (HU)	Noise (HU)	Mean (HU)	Noise (HU)
FDK	-1050.95	44.20	-351.57	103.84
FDK NORM	-1081.48	62.21	-75.40	57.34
xFDK	-1045.66	22.45	-11.55	46.80
Tang (k=2)	-1035.76	17.60	-49.72	42.84
Tang (k=5)	-1041.15	18.19	-27.78	47.08
Tang (k=10)	-1042.22	18.63	-21.71	47.24

Table 3 shows our measurement of slice 1. It is clearly seen that all algorithms suffer from the intensity drop. Noise is also high due to various artifacts. These results show that even the extended methods are not able to suppress all artifacts in the outermost slice.

Table 4. Noise measurements on the third slice.

Slice 3	Air		Water	
Method	Mean (HU)	Noise (HU)	Mean (HU)	Noise (HU)
FDK	-1016.98	28.17	-8.74	29.50
FDK NORM	-1013.90	24.40	-7.71	29.82
xFDK	-1042.79	17.47	-12.61	30.60
Tang (k=2)	-1026.42	19.33	-15.12	41.38
Tang (k=5)	-1023.93	18.80	-15.98	45.17
Tang (k=10)	-1023.74	18.91	-12.29	46.27

On slice 3 most of the artifacts fall outside of the measured water area which fills the middle of the slice (see Table 4). The intensity drop is clearly less, but still visible. Interestingly, the drop is higher for the extended methods. Noise levels are nearly the same. In the case of FDK, the noise in the air area is larger than expected due to the artifacts still prominent near the edge of the slice.

Table 5. Noise measurements on the fifth slice.

Slice 5	Air		Water	
Method	Mean (HU)	Noise (HU)	Mean (HU)	Noise (HU)
FDK	-1006.70	14.80	-8.01	29.83
FDK NORM	-1006.80	14.84	-7.02	29.78
xFDK	-1006.41	14.66	-8.15	29.71
Tang (k=2)	-1026.38	17.45	-14.67	38.29
Tang (k=5)	-1023.83	18.51	-13.32	44.17
Tang (k=10)	-1022.95	18.51	-10.41	45.99

Table 5 shows measurement data from slice 5. This slice is closer to the source, so artifacts are less prominent and only appear near the very edge of the slice. As expected the intensity drop of FDK is not noticeable and nearly the same for all methods. The noise is overall small as well. The extended methods (especially the Tang´s method) show increased noise compared to FDK. This is because the dropped views based on cone angle. To counter this effect, the extended methods should only be used in the region where artifacts are prominent (see 4.2).

Note that regardless of the $k|l|$ parameter, the mean values of Tang's method are lower compared to other methods. On the fifth slice this effect is still present. By increasing the $k|l|$ parameter more artifacts are suppressed and the mean values are getting closer to the expected value with a slight increase of noise.

5 Conclusion

We presented our GPU-based 3D CT reconstruction framework. We implemented the FDK, xFDK and Tang´s algorithm and compared them, regarding visible artifacts and noise.

The original FDK method has a limited field of view in the z direction. Using xFDK or Tang´s method it was possible to extend the reconstruction range from 8 to 14 slices. However visible artifacts remain in off-center areas on the outer slices and they need to be further reduced on the two outermost slice to fully utilize the coverage of the detector.

Extended methods require longer reconstruction time. In our framework, we only calculate partial scan weights for slices where it is necessary. With this optimization the increase of execution time is justified by the extended FOV offered by the algorithms. A further optimization is possible by only calculating the extended weights in each voxel when the voxel is outside of the fully illuminated double cone. This would increase reconstruction speed of partially illuminated slices.

An important question is which method to use in a practical implementation. In our study, the xFDK method proved several advantages over Tang's method in terms of speed and usability. The xFDK method is more robust as it has no adjustable parameter and it is more efficient when implemented on the GPU using float precision.

Acknowledgements

This work is supported by TAMOP-4.2.2.B-10/1-2010-0009 and OTKA K-104476.

References

1. Kak A. C. and Slaney M. Principles of Computerized Tomographic Imaging *IEEE Press USA*
2. Xiaochuan Pan, Emil Y Sidky, and Michael Vannier. Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction? *Inverse Problems*, 25(12):1230009, 2009.
3. B. De Man. Iterative Reconstruction for Reduction of Metal Artifacts in Computed Tomography, 2001. ISBN 90-5682-300-0.
4. B. De Man, J. Nuyts, P. Dupont, G. Marchal, and P. Suetens. Reduction of metal streak artifacts in X-ray computed tomography using a transmission maximum a posteriori algorithm. *Nuclear Science, IEEE Transactions* on, 47(3):977-981, jun 2000.
5. Jakab, G., Huszár T., Csébfalvi, B.: Iterative CT Reconstruction on the GPU, *Sixth Hungarian Conference on Computer Graphics and Geometry*, Budapest, 2012
6. Jakab, G. and Domonkos, B.: Valósidejű helikális cone-beam CT rekonstrukció a Mediso NanoPET/CT készülékben. Képfeldolgozók és Alakfelismerők (KÉPÁF) 7. Konferenciája. Budapest, Hungary (2009)
7. Jakab, G., Rácz, A., Németh, G., and Bükki, T.: Fully GPU Based Real Time Corrections and Reconstruction for Cone Beam Micro CT. IEEE - Science Symposium Conference Record (NSS/MIC), pp. 4068–4071. Orlando, FL, USA (2009)
8. Jakab, G., Rácz, A., Nagy, K.: High Quality Cone-beam CT Reconstruction on the GPU. Hungarian Association for Image Processing and Pattern Recognition. Budapest, Hungary (2011)
9. L. A. Feldkamp, L. C. Davis, and J. W. Kress. Practical cone-beam algorithm. *Journal of the Optical Society of America A*, 1:612-619, June 1984.
10. Grimmer R, Oelhafen M, Elstrøm U, Kachelriess M. Cone-beam CT image reconstruction with extended z range. *Med Phys.* 2009 Jul;36(7):3363-70.
11. Xiangyang Tang et al. A three-dimensional weighted cone beam filtered backprojection (CB-FBP) algorithm for image reconstruction in volumetric CT under a circular source trajectory. *Phys. Med. Biol.* 2005; 50 3889
12. Tuy, H. (1983). An inversion formula for cone-beam reconstruction. *SIAM Journal of Applied Mathematics 43*, 546552.
13. Schaller, S., T. Flohr, H. Wolf, and W. Kalender (1998). Evaluation of a Spiral Reconstruction Algorithm for Multirow-CT. In *RSNA ´98, Nov. 29 Dec 4, Chicago, USA, supplement to Radiology*, volume 209 (P).