GPU Accelerated CBCT Reconstruction from Few Views with SART and TV Regularization

Ping Liu[1], Lin Shi[1,2], Defeng Wang[2], Yu Guo[1], Jianying Li[1], Jing Qin[1]and Pheng-Ann Heng[1,2]

[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[2] The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

Abstract. Compressed sensing-based iterative algorithms can reconstruct high-quality CBCT from undersampled and noisy projection data. However, a practical implementation of these methods still remains a challenge due to the heavy computation. We implemented an algorithm by combining simultaneous algebraic reconstruction technique (SART) and total variation (TV) regularization for the CBCT reconstruction from few views. More importantly, we introduced approaches to fit the SART and TV into the GPU architecture. Experimental results showed that our GPU accelerated algorithm could obtain good reconstruction quality from 20 to 40 projections, as well as significant gain in time performance. It only took $29.1s$ for reconstruction from 120 projections with 40 iterations. The proposed method has potential to make iterative-based CBCT reconstruction more accessable for routine clinical applications.

Keywords: Cone-Beam Computed Tomography Reconstruction, Simultaneous Algebraic Reconstruction Technique, Total Variation, Compressed Sensing, Graphics Processing Units

1 Introduction

Cone-Beam Computed Tomography (CBCT) has been widely used in medical diagnosis and image guided radiation therapy. X-ray radiation dose is still a clinical concern for patients' safety, especially for pediatric patients. The imaging dose can be reduced by using less X-ray projections. However, this should result in the degradation of quality of reconstructed images. A new sampling theory, compressed sensing (CS), has demonstrated the feasibility of recovering signals from incomplete measurements through optimization methods in various mathematical situations [1]. Past studies have shown that iterative image reconstruction (IIR) based on a constrained L_1 or total-variation (TV) optimization has been particularly useful for CBCT reconstruction from few views and noisy projection data. Y. Sidky *et al.* introduced a TV method called adaptive steepest descent projection onto convex sets (ASD-POCS) for incomplete data reconstruction [2, 3]. Jia *et al.* presented a method by minimizing an energy function which consists of a data fidelity term and a TV regularization term [4].

These algorithms optimized the raw data and sparsity cost functions separately in an alternating manner. The performance of these methods relied on too many regularity parameters. Similarly, Ritschl *et al.* proposed an improved TV-based CT image reconstruction method by transferring the step-size determination of two optimization procedures into the raw data domain [5]. Choi *et al.* built a CS problem based on $L1$-norm minimization constrained by statistically weighted least-squares of CBCT projection data [6].

Although these methods perform well under certain circumstances, currently filtered back projection (FBP) remains the most popular method in clinical CT [7]. The reason is that a practical implementation of these methods still remains a challenge. The main problem is the iterative nature in solving the TV-based IIR, which requires multiple iterations of forward and backward projections of large datasets and cannot be completed in a clinically feasible time frame. General purpose graphic processing units (GPUs) have shown great power in increasing efficiencies of heavy duty tasks in medical image processing. Much of the resent work has explored it in CT reconstruction, reducing the computational time from several hours to few minutes. Our previous work applied CUDA to the $3D$ CBCT reconstruction using simultaneous algebraic reconstruction technique (SART) [8, 9]. Xu *et al.* studied the generalization of simultaneous iterative reconstruction technique (SIRT) into ordered subsets SIRT to find the optimal number of subsets as well as relaxation factor settings for optimizing the computational performance [10]. Furthermore, Xu and Mueller presented a framework targeting GPU acceleration of iterative reconstruction algorithms as well as exploiting GPUs to optimize their parameter settings for a given quality/speed performance objective [11].

We implemented an algorithm combining SART algorithm with TV regularization, and accelerated it with GPU. While each of these methods has been described individually in the literature, GPU accelerated SART with TV regularization is new for CBCT reconstruction. We introduced approaches to fit the SART and TV into the GPU architecture: ray-driven projection along with hardware built-in trilinear interpolation, voxel-driven back-projection and TV minimization to avoid redundant computation. Experimental results show that our GPU accelerated algorithm can obtain good reconstruction quality from 20 projections, as well as significant gain in time performance.

2 Methods

2.1 Imaging Model

Figure 1 illustrates the principle of CBCT imaging. We consider a volumetric image represented by a function $v(\mathbf{x})$ with $\mathbf{x} = (x, y, z) \in \mathcal{R}^3$. A projection operator P_ψ maps $v(\mathbf{x})$ onto another function on an x-ray image plane along a projection angle ψ. In the discrete setting, v is divided to a $3D$ grid, with a total number of N voxels and the projection image P is a $2D$ grid with $w \times h$ elements (w and h is the width and height of the projection image respectively). Each of the voxel values v_j is assumed uniform. Then, the objective of reconstruction

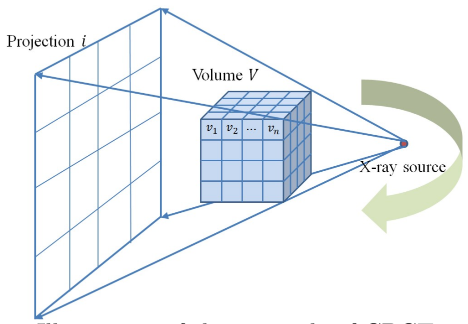

Fig. 1. Illustration of the principle of CBCT imaging

is to estimate all v_j from a set of projections $\boldsymbol{p}$ acquired from different X-ray source orientations along a circular trajectory. The cone-beam projection-data vector $\boldsymbol{p}$ can be written as weighted sums over the pixels traversed by the X-ray as:

$$p_i = \sum_{j=1}^{N} w_{i,j} v_j \tag{1}$$

where $i = 1, 2, \ldots, M$. The $M = w \times h \times proj_{num}$ is the total number of rays from all projections and $proj_{num}$ is the number of projection orientations. The weight component $w_{i,j}$ of the system matrix W is computed by the intersection length of the i-th ray through the j-th pixel. Thus the reconstruction problem can be transformed into an algebraic equation, which can be simply expressed as $\boldsymbol{W}\boldsymbol{v} = \boldsymbol{p}$, where $\boldsymbol{v}$ represents the unknown N voxels in the volume, which is a $N \times 1$ vector. $\boldsymbol{p}$ is a $M \times 1$ vector storing all pixels in the set of k projection images, and $\boldsymbol{W}$ is a $M \times N$ matrix.

2.2 The Proposed Algorithm

The use of constrained TV minimization for CT reconstruction is derived from recent theory in CS [1], where certain sparsely sampled linear systems can be inverted accurately when the underlying object has an approximately sparse gradient magnitude image. In tomographic imaging applications, images formed by taking the magnitude of its gradient could be approximately sparse. In Sidky's ASD-POCS [2], ART is used to update an image reconstructed for data discrepancy reduction first, and then the adaptive steepest descent technique is used in an iterative framework for TV minimization. These two steps are iteratively performed in an alternating manner. Algorithm 1 shows the pseudo-code of the proposed algorithm; we apply SART to enforce projection data consistency, and then adaptive gradient descent is employed to reduce total variation.

Algorithm 1: Pseudo-code of the proposed algorithm

```
$nd \leftarrow 4$, $\beta \leftarrow 0.25$, $\gamma_{max} \leftarrow 0.95$, $\gamma_{red} \leftarrow 0.95$, $\boldsymbol{v} \leftarrow 0$
$iter\_num \leftarrow 10$    //number of iterations of the main loop
for ($i = 0$; $i < iter\_num$; $i$++) do
    $\boldsymbol{v}_0 \leftarrow \boldsymbol{v}$
    for ($j = 0$; $j < proj\_num$; $j$++) do    //SART loop
        Projection: Compute line integrals $p_i$ for all rays of $P_\psi$
        Correction: Subtract the calculated line integral from projection
                    $p_i$ in the projection image, and normalize it
        Back-projection: Distribute corrections onto voxels
        Update: Update $\boldsymbol{v}$
    end for
    for ($j = 0$; $j < vol\_num$; $j$++) do    //enforce positivity
        if $\boldsymbol{v}_j < 0$ then $\boldsymbol{v}_j \leftarrow 0$ end if
    end for
    $dp \leftarrow |\boldsymbol{v} - \boldsymbol{v}_0|$
    if first iteration then $dtvg \leftarrow 0$ end if
    $\boldsymbol{v}_0 \leftarrow \boldsymbol{v}$
    for ($i = 0$; $i < nd$; $j$++) do    //TV-steepest descent loop
        $\mathrm{d}\boldsymbol{v} \leftarrow \nabla \boldsymbol{v} \|\boldsymbol{v}\|_{TV}$
        $\hat{\mathrm{d}\boldsymbol{v}} \leftarrow \mathrm{d}\boldsymbol{v} / |\mathrm{d}\boldsymbol{v}|$
        $\boldsymbol{v} \leftarrow \boldsymbol{v} - \hat{\mathrm{d}\boldsymbol{v}}$
    end for
    $dg \leftarrow |\boldsymbol{v} - \boldsymbol{v}_0|$
    if $dg > \gamma_{max}$ then $dtvg \leftarrow dtvg \cdot \gamma_{red}$ end if
return $\boldsymbol{v}$
```

2.3 GPU Implementation

The whole process starts with loading of the projection data P from CPU to global memory in GPU, together with an initialization of volume $\boldsymbol{v}$ in the texture memory. The volume $\boldsymbol{v}$ is initialized with all-zero. Then in each iteration of the main loop, for each viewing orientation in the SART loop, the volume $\boldsymbol{v}$ is bound to a texture memory and projected in the viewing orientation ψ consistent with the projection P_ψ, so that their difference forms the corrective image. Later, the corrective image is back-projected to the volume $\boldsymbol{v}$ in global memory to update the volume. After the process repeats for all the projections, the TV of the volume is reduced adaptively in the TV-steepest descent loop.

CUDA-accelerated SART Reconstruction. A large number of computationally intensive tasks involved in SART share a common feature, which means we can apply a single operation to different parts of the data elements. In the SART algorithm, for a viewing orientation ψ, a projection image P_ψ is computed. Then, each voxel is corrected by an accumulated correction that is due to all pixels in P_ψ. In the back-projection step, voxels are corrected by the accumulated correction. It can be divided into four steps: projection, correction, back-projection and update. We combine the first two steps to a projection kernel function for computing errors of the projection image, and combine the last two steps to a back-projection kernel for updating the volume on GPU.

For each viewing orientation, there are $w \times h$ (w and h is the width and height of the projection image respectively) rays emit simultaneously in the projection stage, so we can use a ray-driven method for the projection and error image computation, i.e., each thread computing a ray in the projection kernel. For a projection image with $w \times h$ pixels, we assign $w \times h$ threads. The corrective image, i.e., the errors of projection image p'_i computed and the true projection data p_i is calculated by the following equation: $\varepsilon_i = (p_i - p'_i)/\sum_{n=1}^{N} w_{in}$, where $i = 1, 2, \cdots, w \times h$, N and w_{in} are described in Subsection 2.1. The reason to design in this way is that parameters used in this equation are all independent along a ray. Hence, the ε_i can be computed in parallel, leading to the improvement of computation efficiency.

For the back-projection and updating kernel, we choose a voxel-driven method, i.e., each thread computing and updating a voxel. For a volume with N voxels, we assign at least N threads and each thread is used to compute a voxel according the equation:

$$v_j^{(k+1)} = v_j^{(k)} + \beta \frac{\sum\limits_{i \in I_\varphi} (\varepsilon_i w_{ij})}{\sum\limits_{i \in I_\varphi} w_{ij}} \tag{2}$$

where $v_j^{(k)}$ means the j-th voxel value in the k-th iteration; β is relaxation factor, range from 0 to 1 and is usually set to a small value. In the following experiments, β is set to 0.25.

As seen in the Eq. 2, each voxel can be computed independently. Here the volume data must be stored in global memory because texture memory is read only and can't be updated. Once a voxel's back-projection is completed, the voxel v_j can be updated based on ε_i. Readers can refer to [9] for more implementation details.

CUDA-accelerated TV Minimization. For each voxel $v_{x,y,z}$ of $\boldsymbol{v}$, its gradient can be computed by the following equation:

$$|\nabla v_{x,y,z}| = \sqrt{(v_{x+1,y,z} - v_{x,y,z})^2 + (v_{x,y+1,z} - v_{x,y,z})^2 + (v_{x,y,z+1} - v_{x,y,z})^2} \tag{3}$$

where x, y, z are the position of a voxel in the volume $\boldsymbol{v}$. Eq. 3 is valid only for non-border voxels. The gradient can also be thought of as a volume, where each voxel value is the partial derivative of the image TV with respect to that voxel. For parallel computing, we take the derivative with respect to each voxel resulting in a singular expression, which can be computed by a thread. The approximate derivative of each voxel in the gradient volume can be computed as follows:

$$\frac{\partial\|\boldsymbol{v}\|_{TV}}{\partial v_{x,y,z}} \approx \frac{3v_{x,y,z} - v_{x-1,y,z} - v_{x,y-1,z} - v_{x,y,z-1}}{\sqrt{(v_{x,y,z} - v_{x-1,y,z})^2 + (v_{x,y,z} - v_{x,y-1,z})^2 + (v_{x,y,z} - v_{x,y,z-1})^2 + \xi}}$$
$$- \frac{v_{x+1,y,z} - v_{x,y,z}}{\sqrt{\left(v_{x+1,y,z} - v_{x,y,z}\right)^2 + \left(v_{x+1,y,z} - v_{x+1,y-1,z}\right)^2 + \left(v_{x,y,z} - v_{x,y+1,z-1}\right)^2 + \xi}}$$
$$- \frac{v_{x,y+1,z} - v_{x,y,z}}{\sqrt{\left(v_{x,y+1,z} - v_{x-1,y+1,z}\right)^2 + \left(v_{x,y+1,z} - v_{x,y,z}\right)^2 + \left(v_{x,y+1,z} - v_{x,y+1,z-1}\right)^2 + \xi}}$$
$$- \frac{v_{x,y,z+1} - v_{x,y,z}}{\sqrt{\left(v_{x,y,z+1} - v_{x-1,y,z+1}\right)^2 + \left(v_{x,y,z+1} - v_{x,y-1,z+1}\right)^2 + \left(v_{x,y,z+1} - v_{x,y,z}\right)^2 + \xi}} \tag{4}$$

where ξ is a small positive number to avoid dividing zero. Here, we set $\xi = 10^{-8}$.

For the TV-steeptest descent loop, the most time consuming step is the computation of $\partial\|\boldsymbol{v}\|_{TV}/\partial v_{x,y,z}$. We also use a voxel-driven method, i.e., each thread computing the derivative of a voxel. With N voxels of the volume, we assign at least N threads. According to [2], in the actual algorithm, we also employ the normalized TV gradient. In this regard, we need to sum the derivatives of all the voxels. Here, we adapt the optimized reduction algorithm in the CUDA SDK.

3 Results

All experiments were conducted on a desktop PC equipped with a Interl(R) Xeon(R) CPU (W3505, 2.53GHz), 4 GB memory and an nVIDIA GTX680 GPU with 8 multiprocessors, 192 cores per multi-processor and CUDA 4.2. We first evaluated the quality of images reconstructed from various numbers of projections using our method and compared them with the images reconstructed using SART only. Then, we implemented a CPU version of our method and compared it with our GPU version.

We present the CBCT reconstruction results on a digital Shepp-Logan phantom[3]. This phantom is often used in evaluating tomographic reconstruction algorithms. The phantom was generated at head region with a size of $128 \times 128 \times 128$ voxels. The x-ray imager was modeled to be an array of 256×256 detectors. X-ray projection images of the phantom were generated along various equally spaced orientations and were then used as the input for the reconstruction.

3.1 Reconstructed Image Quality from Few Views

In case of CT reconstruction from few views, iterative reconstruction methods such as SART perform better than analytical method such as FDK, in spite of

[3] $3D$ Shepp-Logan phantom. http://tomography.o-x-t.com/

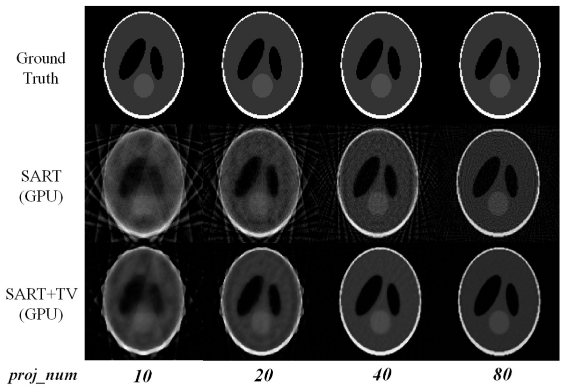

Fig. 2. Reconstructed images (70th Slice) from different projections with 40 iterations.

heavy computation [12]. The CS-based reconstruction algorithms such as our method are supposed to further reduce the number of projections. In Algorithm 1, the length of the measurement P and the number of rows of the system matrix W are linearly proportional to the number of projections. So the projection number has great effect on the problem size and hence the computation time per iteration. To improve the computational efficiency as well as to ensure acceptable reconstruction quality, we need to choose a proper number of projections $proj\_num$. Another key factor is the iterative number $iter\_num$.

The reconstruction results based on the $proj\_num$ = 10, 20, 40 and 80 projections with 40 iterations are drawn in Fig. 2 respectively (the 70th axial slices are presented representatively). For comparison, the image reconstructed from SART only as well as the ground-truth image are also given. From Fig. 2, it is observed that with 40 iterations, SART+TV algorithm can achieve much better results than SART algorithm when the $proj\_num$ = 10, 20 and 40, as it effectively reduced the artifacts due to incomplete projections. To show the effects of the iteration number of the main loop, axial slices of the reconstruction results based on $proj\_num$ is 20 with $iter\_num$ is 10, 20, 40 and 80 are presented in Fig. 3 respectively. Fig. 3 further demonstrates that the image quality of our method is always better than that of SART at various iteration numbers.

To compare the reconstruction results quantitatively, correlation coefficient (CC) of the reconstruction results and the ground truth are calculated according to the following equation [13]:

$$CC = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}(v_{ijk}-\bar{v})(v'_{ijk}-\bar{v'})}{\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}(v_{ijk}-\bar{v})^2\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}(v'_{ijk}-\bar{v'})^2}} \tag{5}$$

To fully evaluate the performance of the algorithms, CC of the reconstructed volumes and the ground truth from different projections with different iterations

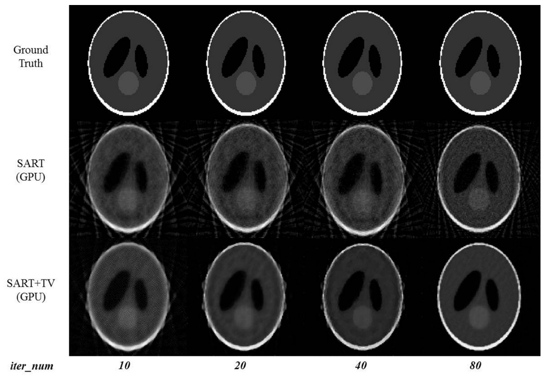

Fig. 3. Reconstructed images (70*th* Slice) from 20 projections with different iterations.

are shown in Fig. 4 (SART(20) means SART algorithm from 20 projections). From Fig. 4,we can see that TV minimization greatly improved the CC when $proj\_num = 20, 40$; and when reconstruction by SART+TV from 20 projections with more than 80 iterations, the CC gets closer to that obtained from 120 projections with more than 10 iterations by SART. This means we could use much less projections to get as accurate reconstructed images as those from 120 projections, leading to reduction of X-ray radiation. Fig. 4 also shows that when $proj\_num$ is more than 80, SART could reaches convergence far more quickly and it seems SART+TV has no advantages compared with SART.

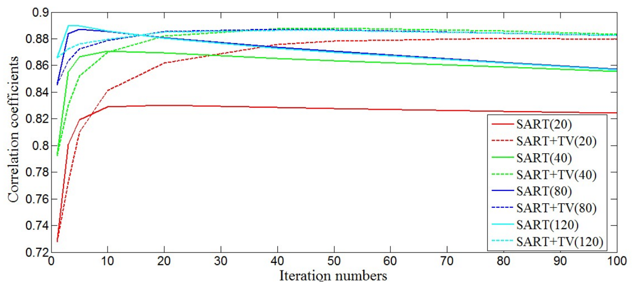

Fig. 4. Correlation coefficients of the volumes reconstructed and the ground truth from different projections with different iterations.

3.2 Time performance

Table. 1 shows the time statistics for reconstruction of the Shepp-Logan phantom sized $128 \times 128 \times 128$ from various numbers of projections sized 256×256 with different iterations of the main loop. The hardware configuration especially

Table 1. Execution time of reconstruction using various projections and 10/40 iterations, with CPU and CUDA-based implementation.

Implementation	CPU	GPU	GPU	CPU	GPU	GPU	GPU	GPU
	(10,10)	(10,10)	(10,40)	(20,10)	(20,40)	(40,40)	(80,40)	(120,40)
SART	3205.79	0.44	1.73	6449.91	3.42	6.82	13.4	24.42
SART+TV	3214.51	2.56	10.31	6514.31	11.94	15.23	21.86	29.1

Table 2. Execution time of reconstruction process using 40 projections with CUDA-based implementation,and with different projection resolutions

Projection size	256×256		512×512		768×768		1024×1024	
	(40,20)	(40,40)	(40,20)	(40,40)	(40,20)	(40,40)	(40,20)	(40,40)
SART	3.61	6.97	5.27	10.58	8.42	16.7	13.18	26.27
SART+TV	7.64	15.23	9.56	19.06	12.59	25.04	17.3	34.47

the GPU specification is much higher than that used in our previous work, and we used less projections to reconstruct the volume by combing the TV Regularization, so we obtained very good speedup values as presented in Table. 1. The time recorded was that the main loop took both for the CPU and GPU version of Algorithm 1. The data also included the overhead of copying data from the CPU memory to the GPU memory for the GPU version. From Table. 1 we can see that CUDA-based implementation greatly reduced the reconstruction time compared with CPU-based implementation. It only takes $29.1s$ for reconstruction from 120 projections with 40 iterations.

Table. 2 shows the time statistics for reconstruction of the Shepp-Logan phantom sized $128 \times 128 \times 128$ from 40 projections with different resolutions via the GPU version of SART and SART+TV, respectively. Compared with SART, SART+TV takes more time due to the TV minimization loop. However, the total time of SART+TV is still acceptable for practical applications.

4 Conclusions

SART and TV was combined and accelerated with CUDA-based GPU for CBCT reconstruction from few views. Experimental results showed that our CUDA-based implementation could obtain good reconstruction quality from 20 to 40 projections, as well as significant gain in time performance. It only took $29.1s$ for reconstruction from 120 projections with 40 iterations. In addition, TV minimization greatly improved the quality of the reconstructed image when $proj\_num$ = 20, 40; and when reconstruction by SART+TV from 40 projections with about 30 iterations, the quality of the reconstructed image was close to that obtained from 120 projections with more than 10 iterations by SART. When $proj\_num$ was more than 80, SART could reached convergence far more quickly and it seemed SART+TV had no advantages compared with SART. The proposed method has potential to make iterative-based CBCT reconstruction more accessible for routine clinical applications.

Acknowledgments

The work described in this paper was partially supported by a grant from Ministry of Science and Technology of the People's Republic of China under the Singapore-China 9th Joint Research Programme (Project No.: 2013DFG12900), grants from the National Natural Science Foundation of China (Project No. 81101111 and 61233012), and grants from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No.: CUHK 416712, CUHK 473012).

References

1. Candes, E.J., Romberg, J.K., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Communications on pure and applied mathematics **59**(8) (2006) 1207–1223
2. Sidky, E.Y., Pan, X.: Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. Physics in medicine and biology **53**(17) (2008) 4777–4807
3. Sidky, E.Y., Duchin, Y., Pan, X., Ullberg, C.: A constrained, total-variation minimization algorithm for low-intensity x-ray CT. Medical Physics **38** (2011) S117–S125
4. Jia, X., Lou, Y., Lewis, J., Li, R., Gu, X., Men, C., Song, W.Y., Jiang, S.B.: GPU-based fast low-dose cone beam ct reconstruction via total variation. Journal of X-Ray Science and Technology **19**(2) (2011) 139–154
5. Ritschl, L., Bergner, F., Fleischmann, C., Kachelrieß, M.: Improved total variation-based CT image reconstruction applied to clinical data. Physics in Medicine and Biology **56**(6) (2011) 1545–1561
6. Choi, K., Wang, J., Zhu, L., Suh, T.S., Boyd, S., Xing, L.: Compressed sensing based cone-beam computed tomography reconstruction with a first-order method. Medical physics **37** (2010) 5113–5125
7. Pan, X., Sidky, E.Y., Vannier, M.: Why do commercial CT scanners still employ traditional, filtered back-projection for image reconstruction? Inverse problems **25**(12) (2009) 1–36
8. Lu, Y., Wang, W., Chen, S., Xie, Y., Qin, J., Pang, W.M., Heng, P.A.: Accelerating algebraic reconstruction using CUDA-enabled GPU. In: CGIV. (2009) 480–485
9. Pang, W.M., Qin, J., Lu, Y., Xie, Y., Chui, C.K., Heng, P.A.: Accelerating simultaneous algebraic reconstruction technique with motion compensation using CUDA-enabled GPU. International journal of computer assisted radiology and surgery **6**(2) (2011) 187–199
10. Xu, F., Xu, W., Mueller, K., Jones, M., Keszthelyi, B., Sedat, J., Agard, D.: On the efficiency of iterative ordered subset reconstruction algorithms for acceleration on GPUs. Computer methods and programs in biomedicine **98**(3) (2010) 261–270
11. Xu, W., Mueller, K.: Using GPUs to learn effective parameter settings for GPU-accelerated iterative ct reconstruction algorithms. GPU Computing Gems Emerald Edition (2011) 693–708
12. Kak, A.C., Slaney, M.: Principles of computerized tomographic imaging. AC Kah and Malcolm Slaney (1999)
13. Zhuang, T.: Principle and algorithms of CT. Shanghai Jiaotong University Press (1992)