Hyperfast Perspective Cone–Beam Backprojection

Marc Kachelrieß, *Member, IEEE,* Michael Knaup, Olivier Bockenbach

***Abstract*— Cone–beam image reconstruction, such as the reconstruction of CT projection values, is computational very demanding. The most time–consuming step is the backprojection that is often limited by the memory bandwidth. Recently, a novel general purpose architecture optimized for distributed computing became available: the Cell Broadband Engine (CBE). Its eight synergistic processing elements (SPEs) currently allow for a theoretical performance of 192 GFlops (3 GHz, 8 units, 4 floats per vector, 2 instructions, multiply and add, per clock).**

Our aim is to maximize the image reconstruction speed for flat–panel–based cone–beam CT such as micro–CT or C–arm–CT. Therefore we implemented a highly optimized perspective cone–beam backprojection algorithm on the Cell processor. Data mining techniques and double buffering of source data were extensively used to optimally utilize both the memory bandwidth and the available local store of each SPE. The voxel–driven backprojection code uses 32 bit floating point arithmetic and bilinear interpolation between neighboring detector channels. The latter is performed in two stages by first upsampling the detector (this includes bilinear interpolation) to double the number of detector pixels followed by a nearest neighbor interpolation during backprojection.

Performance was measured by backprojecting simulated data with 512 cone–beam projections per full rotation and 1024 by 1024 detector elements. The data were backprojected into a volume of 512^3 voxels fully contained in the field of measurement using an optimized PC–based (CPU–based) approach and the new Cell–based (CBE–based) algorithm. Both the PC and the CBE were clocked at 3 GHz.

PC–based backprojection takes 3.2 min whereas the CBE version finishes within 13.6 s. Using both CBEs of our dual Cell–based blade (Mercury Computer Systems) one can do the cone–beam backprojection in 6.8 s.

I. Introduction

CELL processors are general purpose processors that combine a PowerPC element (PPE, manager) with eight synergistic processor elements (SPE, worker) [1], [2], [3]. A single chip contains eight SPEs, each with an synergistic processing unit (SPU), a memory flow controller (MFC), and 256 kB of SRAM that are used as local store (LS) memory. The LS runs in its own address space at the full 3.2 GHz clock frequency. An SPU uses 128 bit vector operations and can execute up to eight floating point instructions per clock cycle. We focus on backprojecting floating point values (4 bytes each). Hence the data vector consists of four floats. (See our related submission [4] for details.)

Prof. Dr. Marc Kachelrieß: Institute of Medical Physics (IMP), University of Erlangen–Nürnberg, Henkestraße 91, 91052 Erlangen. E–mail: marc.kachelriess@imp.uni-erlangen.de.

Dr. Michael Knaup: Institute of Medical Physics (IMP), University of Erlangen–Nürnberg, Henkestraße 91, 91052 Erlangen. E–mail: michael.knaup@imp.uni-erlangen.de.

Olivier Bockenbach: Mercury Computer Systems, Lepsiusstr. 70, 12163 Berlin. E–mail:olivier@mc.com

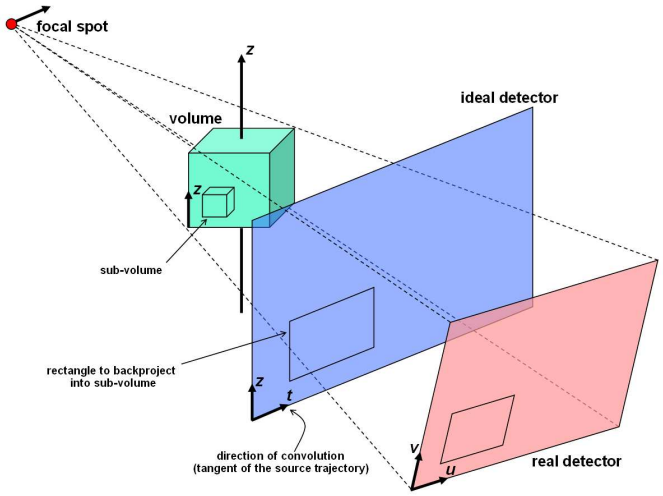

Fig. 1. Illustration of the perspective geometry, of the real–to–ideal rebinning and of the sub–volume and tiling strategy.

The aim of this investigation is to implement a 3D cone–beam backprojection algorithm, as it is used for Feldkamp–type image reconstruction, for example, for the Cell processor and to benchmark its performance against PC–based implementations.

II. Method

We consider a backprojection of type

$$f(\boldsymbol{r}) = \int d\alpha\, w^2(\alpha,\boldsymbol{r})\, p\big(\alpha, u(\alpha,\boldsymbol{r}), v(\alpha,\boldsymbol{r})\big)$$

with

$$\begin{aligned} u(\alpha,\boldsymbol{r}) &= (c_{00}x + c_{01}y + c_{02}z + c_{03})w(\alpha,\boldsymbol{r}) \\ v(\alpha,\boldsymbol{r}) &= (c_{10}x + c_{11}y + c_{12}z + c_{13})w(\alpha,\boldsymbol{r}) \\ w(\alpha,\boldsymbol{r}) &= (c_{20}x + c_{21}y + c_{22}z + c_{23})^{-1} \end{aligned}$$

and $c_{ij} = c_{ij}(\alpha)$. Here, f is the image, p are the (usually preweighted and convolved) rawdata and α, u, and v are the projection index, the detector's axial and the detector's longitudinal coordinate, respectively (figure 1). The coefficients $c_{ij} = c_{ij}(\alpha)$ define the perspective transform from the detector into the volume and are arbitrary functions of the projection angle, in general. The distance weight $w(\boldsymbol{r})$ is required for cone–beam filtered backprojection, the square results from the fact that the kernel is a homogeneous function of degree -2.

A. Implementation

The backprojection integral is usually realized in a discretized version called voxel–driven backprojection. Our reference code that backprojects one projection is shown in

```
void PerBackProjRefLI(int const I, int const J, int const K,
                      int const L, int const M,
                      float const c00, ..., float const c23,
                      float const * const Raw,
                      float       *       Vol)
    {
    for(int i=0; i<I; i++) // slow voxel index
    for(int j=0; j<J; j++)
    for(int k=0; k<K; k++) // fast voxel index
        {
        float const     w=1/(c20*i+c21*j+c22*k+c23);
        float const lreal=w*(c10*i+c11*j+c12*k+c13);
        float const mreal=w*(c00*i+c01*j+c02*k+c03);
        int const l=int(lreal); float const wl=lreal-l;
        int const m=int(mreal); float const wm=mreal-m;

        (*Vol++)+=w*w*((1-wl)*((1-wm)*Raw[l*M  +m  ]
                               +wm *Raw[l*M  +m+1])
                       +wl *((1-wm)*Raw[l*M+M+m  ]
                               +wm *Raw[l*M+M+m+1]));
        }
    }
```

Listing 1: Reference code. The pixel indices i, j and k correspond to x, y and z and the detector indices l and m correspond to u and v.

listing 1. For a complete image reconstruction the function must be called N times where N is the number of projections acquired. This reference code is not optimized, however.

An optimized version that was coded in 2001 is available to our group and is used for comparisons to the Cell–based algorithm. This PC–based implementation is purely C++ and makes no explicit use of assembler segments or CPU intrinsics. It achieves its high performance by using a hierarchical memory layout and loop unrolling techniques. It further replaces the linear interpolation by a hybrid technique that first performs a detector upsampling based on linear interpolation (similar to [21]). The upsampled detector has very fine pixels and for the subsequent backprojection step it is sufficient to carry out a nearest neighbor interpolation without impairing image quality. We call this rebinning from the real physical detector data onto an ideal virtual detector that is properly aligned to one of the volume axes real–to–ideal rebinning (figure 1).

For the Cell one is forced to use a hierarchical memory layout since each SPE's local store is limited to 256 kB. Hence, we divide the volume into small sub–volumes, typically 32^3 voxels. Each worker takes care of backprojecting into his sub–volume. Projection patches — these are portions of the full projection that are actually required by the worker — are passed from the manager to the workers using DMA (figure 1). Double buffering of these patches ensures that the DMA can run while the worker is busy backprojecting the previous projection.

Care was taken during implementation to obtain a high rate of dual issues. A dual issue occurs when a group of fetched instructions has two issueable instructions, one of which is executed by a unit on the even pipeline and the other executed by a unit on the odd pipeline. For example multiplies and adds run on the even, load and store issues run on the odd pipeline. Since there is a larger number of loads and stores than multiplies or adds it is possible to completely hide the multiplies and adds behind the loads and stores.

To achieve the dual issue as often as possible 8–fold loop unrolling of the innermost loop was done manually. Thereby instructions were rescheduled to hide the instruction latencies and to enforce the dual issue case.

Two backprojection versions were implemented for the Cell. A direct version that is equivalent to the reference code and a hybrid method that uses an intermediate upsampled detector just as our optimized PC–based algorithm does.

	C/O	T^{-1}	T	$512 \cdot T$
PC, reference	309	0.07 fps	13.6 s	1.93 h
PC, hybrid	8.58	2.66 fps	376 ms	3.21 min
Cell, direct	1.19	18.8 fps	53.1 ms	27.2 s
Cell, hybrid	0.59	37.6 fps	26.6 ms	13.6 s

TABLE I
PERSPECTIVE BACKPROJECTION PERFORMANCE ACHIEVED WITH OUR APPROACHES.

B. Performance Assessment

The code was implemented to cope with any number of voxels (also non–cubic volumes), any number of projections and any number of pixels per projection (also non–square detectors). We assessed the performance of backprojecting 512 cone–beam projections into a volume of size $512\times512\times512$. The complexity of the code is $O = 512^4$ operations. All voxels were fully contained in the field of measurement. The fact that each projection consisted of 1024×1024 detector elements is irrelevant to our timing measurement.

The standard and the optimized code ran on a single 3.06 GHz Xeon processor with 533 MHz front side bus while the Cell–based implementation uses a 3.2 GHz Cell processor running on a dual Cell blade (Mercury Computer Systems). All timing values were linearly scaled to 3.0 GHz, for convenience. The time T per 512×512 slice was measured as an average of the time $512 \cdot T$ required to backproject the whole volume. Care was taken that no other significant CPU or SPU workload impaired our measurements (the PC operating system is Windows XP and the Cell operating system is Linux and hence multitasking may occur for both processors).

Additionally, we compute the number of CPU clock cycles per operation as C/O with $C = FT$ being the number of clock cycles per reconstructed image and F being the clock frequency that equals 3.06 GHz for our PC, 3.2 GHz for the Cell system, and 3.0 GHz for our scaled values.

III. RESULTS

The timing results are shown in table I. The reference algorithm is the code provided in listing 1. It is PC–based but not optimized. The PC–based hybrid code is the said optimized implementation. The Cell–based codes are both highly optimized as detailed earlier in this paper.

It should be noted that we found no performance increase when taking advantage of the fact that the distance weight is proportional to the denominator of the perspective transform

and our optimized implementations actually allow for a more flexible choice of distance weighting.

Apparently, the CBE achieves to backproject all 512 projections into the 512^3 volume in 13.6 s. Considering that two Cells are available per blade one may finish a complete cone–beam backprojection in 6.8 s using a dual Cell board.

DMA Latency

One of the most prominent feature of the CBE is its fast DMA between the main memory and the worker local store. Since Cell DMA works in parallel to the SPU's command execution pipeline, the DMA latency may be completely hidden for some CPU-limited problems.

To measure the DMA latency for our implementations, we performed dummy reconstructions without DMA transfers and calculated the differences of the total backprojection times to that of real backprojections. The backprojection times were measured with clock–cycle precision via the so–called worker decrementer. The decrementer is a counter on each SPU that is decremented on a clock cycle base. Statistical errors were estimated by repeating all measurements five times.

Table II shows the results for the direct perspective backprojection of a 512^3 volume using linear interpolation. It turned out that the DMA fraction of the total reconstruction time is about about 0.37% and thus negligible.

IV. Other Attempts

Other groups have made lots of efforts to speed up CT image reconstruction. Although a fair and quantitative comparison is not always possible, table III lists those performance figures that have been published in this millennium, including those published in this paper. Benchmarks found in older literature are considered obsolete due to the ongoing developments in computer technology.

To allow for some comparison we scale the values found in the literature to the case of backprojecting 512 projections into a volume of $512 \times 512 \times 512$ voxels that are fully contained in the field of measurement (no empty voxels). The projection size itself is considered irrelevant. For PC–based implementations the CPU clock rate is scaled to 3.0 GHz. This assumption is quite optimistic since backprojection is usually limited by memory latency and memory speed has not increased that significantly during the last years. Especially for older experiments that have been carried out on slow CPUs this scaling will overestimate the actual performance that could be achieved with the same algorithm on modern CPUs. Note that in most cases comparing the cost–to–performance ratio would be more adequate than just comparing performance. However, there are no reliable cost figures available to us.

A further complication regarding the cone–beam backprojection algorithms is given by the fact that the underlying assumptions are different from publication to publication and it is not always clear whether all assumptions are precisely stated in the paper. One example are assumptions about the detector alignment. Another assumption that is sometimes made is that the scanner performs an exact rotation. In this case the perspective coefficients are not independent but can be transformed into each other using a rotation matrix. This allows to use the resulting symmetries and thereby speed up the reconstruction process.

We further want to point to the fact that there are significant differences whether the reconstructed FOV is cuboid or cylindrical (or even spherical). A cylindrical FOV contains only $\pi/4 \approx 79\%$ of the voxels that are contained in the enclosing cuboid. This adds another 21% uncertainty to the values found in the literature if the FOV shape is not disclosed or if voxels outside the FOM are not backprojected. Similarly, the volume ratio between a spherical FOV and its enclosing cube is $\pi/6 \approx 52\%$.

Divide–and–conquer–type backprojection, such as Fourier–based image reconstruction [5], [6], [7], hierarchical backprojection [8] or the link–method [9], for example, is of completely different type than the standard backprojection algorithms discussed here and therefore not included in our comparison. It should be noted that these methods have the potential to increase reconstruction speed by a factor $cN/\ln N$ with c being some (sometimes rather small) constant. Except maybe for Fourier reconstruction there is no highly optimized implementation that can really compete with the standard backprojection performance values listed here. Further, some of these divide–and–conquer concepts work well in 2D but become difficult or impossible in the cone–beam case. E.g. Fourier reconstruction in 3D only works when the complete Radon data are available [10]. Last but not least they often suffer from a trade–off between reconstruction speed and reconstruction accuracy, except for the Fourier–based algorithms.

We also did not include the interesting distance–driven backprojection algorithm proposed in reference [11]. Although the authors claim significant speed–ups relative to their pixel–driven backprojection implementation their approach is not fully optimized. Hence the achievable timing cannot be reliably determined from the paper.

Wiesent et al. use a dual Pentium III Xeon 550 MHz CPU [12]. They reconstruct 256^3 voxels from 100 projections in about 40 s. In terms of the 512^4 operations at 3 GHz and a single CPU this scales to 10.0 min.

Yu et al. provide a PC–based implementation [13]. On a 500 MHz Pentium III CPU they can reconstruct a 512^3 volume from 288 projections within 15.03 min. They use a spherical FOV and do not backproject voxels outside this sphere. Scaled to 3 GHz, to 512 projections and to a cubic FOV this becomes 8.51 min whereby we believe that this scaling yields a far too optimistic value since memory speed did not improve the same way as the CPU clock rates did. Their code utilizes single instruction multiple data (SIMD) instructions.

Goddard and Trepanier present an FPGA–driven reconstruction (which includes convolution) that can reconstruct a 512^3 volume from 300 projections between 15 s and 38.7 s [14], [15], [16]. The range of values corresponds to using one or more FPGAs. Since the convolution process was completely hidden behind the backprojection the reconstruction times also correspond to the backprojection performance. Scaling the 38.7 s (one FPGA) to the 512 projections used here we obtain a performance of 66.0 s. Among other assumptions

	without DMA	DMA get	DMA put	total
`PerBackProj`, direct	(27096 ± 1) ms	(94 ± 10) ms	(6 ± 1) ms	(27196 ± 12) ms

TABLE II

DMA LATENCIES FOR THE DIRECT PERSPECTIVE BACKPROJECTION OF A 512^3 VOLUME USING LINEAR INTERPOLATION. DMA GET: RAWDATA FLOW FROM MANAGER TO WORKER. DMA PUT: VOLUME FLOW FROM WORKER TO MANAGER.

	Type	Hardware	Time	Comment
Wiesent et al. [12]	LI / f32	CPU	10.0 min	includes convolution
Yu et al. [13]	? / ?	CPU	8.51 min	includes convolution
Goddard, Trepanier [14], [15], [16]	LI / i16	FPGA	66.0 s	detector $\parallel$ rotation axis
Xu and Mueller [17]	LI / f32	CPU	7.57 h	
	LI / f32	GPU	34 min	
Kole and Beekman [18]	NN / ?	GPU	17.3 min	
	LI / ?	GPU	25.8 min	
Hornegger [19]	NN / f32	CBE	1.99 min	simulation
Mueller and Xu [20]	? / f32	CPU	1.28 h	includes convolution
	? / f32	GPU	17.9 min	includes convolution
	? / i16	GPU	3.84 min	includes convolution
Riddell and Trousset [21]	LI / ?	CPU	9.15 min	hybrid
Kachelrieß et al. [this]	LI / f32	CPU	3.21 min	hybrid
	LI / f32	CBE	27.2 s	direct
	LI / f32	CBE	13.6 s	hybrid

TABLE III

BACKPROJECTION PERFORMANCE. ALL VALUES HAVE BEEN SCALED TO 512 PROJECTIONS AND 512^3 VOXELS. ALL VALUES WERE FURTHER SCALED TO A SINGLE PROCESSING UNIT, I.E. TO ONE CPU, ONE FPGA, ONE GPU AND TO ONE CBE, RESPECTIVELY, AND TO 3.0 GHZ IN THE CASE OF CPU– AND CELL–BASED ALGORITHMS. THE TYPE COLUMN SPECIFIES THE INTERPOLATION TYPE, NEAREST NEIGHBOR (NN) OR BI–LINEAR INTERPOLATION (LI), AND THE TYPE OF ARITHMETHIC USED: F+NUMBER OF BITS DENOTES FLOATING POINT ARITHMETHICS WHILE I+NUMBER OF BITS STANDS FOR INTEGER (FIXED POINT) ARITHMETICS.

the algorithm assumes one detector axis to be parallel to the rotation axis, the center of rotation to be the center of the cubic volume and the distances of the focal spot to the isocenter and to the detector to be constant. The first assumption implies that their backprojection matrix is of the same type as for our hybrid approach. The real–to–ideal rebinning is not mentioned and probably not included in their experiment. The other assumptions imply that the perspective coefficients c_{ij} are generated by a rotation matrix.

Xu and Mueller published on GPU–based image reconstruction [17]. They compare a "fairly optimized CPU implementation" with the GPU–based approach they propose. The PC runs on 2.66 GHz and the GPU is an Nvidia FX 5900. Their backprojection (LI) requires 75 s for the fairly optimized CPU algorithm and 5 s for the GPU code when a volume of 128^3 and 80 projections are used. In terms of our 512^4 problem at 3.0 GHz these values become 7.57 h for the PC and 34 min for the GPU, respectively.

Kole and Beekman recently optimized a statistical image reconstruction algorithm to run on a graphical processing unit (GPU) [18]. Each iteration consists of one forward and two backprojection steps. Since the forward projection is of about the same speed as the backprojection we may divide their performance values by three to estimate the GPU performance of a perspective backprojection. They cite a speed of 195 s for NN and of 290 s for LI for one iteration consisting of 256 projections and a 256^3 volume. The time needed for a backprojection of the 512^4 problem will be about 17.3 min for the nearest neighbor backprojection and about 25.8 min for the linear interpolation version.

Hornegger recently presented a backprojection code for the Cell processor that was tested on a Cell simulator and not on a real Cell system [19]. They show that 6 projections per second can be backprojected (NN) on a 512^3 volume using a dual Cell (16 SPUs) running with 2.1 GHz and speculate that the code can be further sped up by a factor of five. Scaling their value to 512 projections, 3.0 GHz and 1 CBE yields 1.99 min for the complete volume.

Lately, Mueller and Xu published new results on GPU–based CT image reconstruction [20]. Since the problem of floating point arithmetics on GPUs seems not yet to be solved they find integer arithmetics very useful to speed up the process although image quality becomes inferior. Depending on what arithmetic is used the timing for a 256^3 volume and 160 projections achieved on an Nvidia 7800 FX GPU ranges from everything between 1.9 to 42 s. Adequate images are

provided by their "dual–pass" approach that allows for 16 bit accuracy and finishes in 9 s. Full floating point accuracy requires 42 s on the GPU. Their PC–based implementation needs 180 s for the same task in full floating point accuracy (CPU and bus clock frequencies are not stated). Normalizing their values to 512^4 yields 3.84 min (16 bit integer) and 17.9 min (single precision float) for the GPU and 1.28 h for the CPU. It should be noted that these values include the convolution step which typically makes up about 10% of the reconstruction time if it cannot hidden behind the backprojection by using a parallel thread.

Riddell and Trousset implemented a rectification–based perspective backprojection on a 3.4 GHz Pentium 4 CPU [21]. Their "rectification" is similar to our real–to–ideal rebinning and therefore their algorithm is a hybrid approach. In contrast to our hybrid algorithm that performs alignment only along one volume axis their backprojection requires the ideal detector to be aligned parallel to one of the volume faces (i.e. aligned along two volume axes). The authors state that backprojecting 148 projections into a cylinder of 512 voxels height and diameter takes 110 s. Scaling this to our 512^4 problem and to 3.0 GHz we find that their code takes 9.15 min.

V. Conclusion

The Cell Broadband Engine allows for hyperfast backprojection on a general purpose hardware. Our implementation greatly outperforms other existing hard- or software by one order of magnitude. The best performance found in the literature is FPGA–based, has 16 bit fixed point accuracy, and needs 66 s to finish the $O = 512^4$ backprojection task [14], [16]. The dual Cell blade can backproject 512 projections of size 1024^2 into a volume of 512^3 voxels in 6.8 s with full 32 bit floating point accuracy and linear interpolation.

Note that convolution of 512 projections of size 1024×1024 with a 2047–element kernel runs in 0.2 s on the dual Cell blade and is therefore negligible compared to the backprojection step.

Considering that typical scan times are in the same order (at least for flat–panel detector–based CT) one can potentially achieve real–time imaging at full spatial resolution. Besides its very high performance probably the most significant advantage of the CBE over other hardware–based acceleration approaches is its versatility. FPGA–, ASIC– or GPU–based solutions are usually limited to certain functionality. The Cell processor, in contrast, is a general purpose hardware that can be used for all kinds of tasks ranging from data preprocessing, image reconstruction, image display, volume rendering to more complicated issues such as dose and scatter calculation. Its high performance may even leverage completely new applications or may help to bring other, low performant approaches into clinical routine, such as iterative or statistical CT image reconstruction, for example [22].

References

[1] H. P. Hofstee, "Power efficient processor architecture and the Cell processor," *Proceedings of the 11th International Symposium on High–Performance Computer Architecture*, Feb. 2005.

[2] D. Pham, S. Asano, M. Bolliger, M. N. Day, H. P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki, and K. Yazawa, "The design and implementation of a first–generation Cell processor," *IEEE International Solid–State Circuits Conference*, pp. 184–185, Feb. 2005.

[3] B. Flachs, S. Asano, S. H. Dhong, H. P. Hofstee, G. Gervais, R. Kim, T. Le, P. Liu, J. Leenstra, J. Liberty, B. Michael, H. Oh, S. M. Mueller, O. Takahashi, A. Hatakeyama, Y. Watanabe, and N. Yano, "A streaming processing unit for a Cell processor," *IEEE International Solid–State Circuits Conference*, pp. 134–135, Feb. 2005.

[4] M. Kachelrieß, M. Knaup, and O. Bockenbach, "Hyperfast parallel–beam backprojection," *IEEE Medical Imaging Conference Record*, pp. M14–168, 2006.

[5] H. Stark, J. W. Woods, I. Paul, and R. Hingorani, "An investigation of computerized tomography by direct Fourier inversion and optimum interpolation," *IEEE Transactions on Biomedical Engineering*, vol. BME–28, no. 7, pp. 496–505, July 1981.

[6] H. Schomberg and J. Timmer, "The gridding method for image reconstruction by Fourier transformation," *IEEE Transactions on Medical Imaging*, vol. 14, no. 3, pp. 596–607, Sept. 1995.

[7] S. Schaller, T. Flohr, and P. Steffen, "An efficient Fourier method in 3D reconstruction from cone–beam data," *IEEE Transactions on Medical Imaging*, vol. 17, pp. 244–250, Feb. 1998.

[8] S. Basu and Y. Bresler, "An $O(N^2 \log N)$ filtered backprojection reconstruction algorithm for tomography," *IEEE Transactions on Medical Imaging*, vol. 9, no. 10, pp. 1760–1773, Oct. 2000.

[9] P.-E. Danielsson and M. Ingerhed, "Backprojection in $O(N^2 \log N)$ time," *IEEE Nuclear Science Symposium Record*, vol. 2, pp. 1279–1283, 1998.

[10] C. Axelsson and P.-E. Danielsson, "Three–dimensional reconstruction from cone–beam data in $O(N^3 \log N)$ time," *Phys. Med. Biol.*, vol. 39, no. 3, pp. 447–491, 1994.

[11] B. De Man and S. Basu, "Distance–driven projection and backprojection in three dimensions," *Phys. Med. Biol.*, vol. 49, pp. 2463–2475, 2004.

[12] K. Wiesent, K. Barth, N. Navab, P. Durlak, T. Brunner, O. Schuetz, and W. Seissler, "Enhanced 3–D–reconstruction algorithm for C–arm systems suitable for interventional procedures," *IEEE Transactions on Medical Imaging*, vol. 19, no. 5, pp. 391–403, May 2000.

[13] R. Yu, R. Ning, and B. Chen, "High–speed cone–beam reconstruction on PC," *SPIE Medical Imaging Proc.*, vol. 4322, pp. 964–973, 2001.

[14] M. Trepanier and I. Goddard, "Adjunct processors in embedded medical imaging systems," *SPIE Medical Imaging Proc.*, vol. 4681, pp. 416–424, 2002.

[15] I. Goddard and M. Trepanier, "High–speed cone–beam reconstruction: An embedded systems approach," *SPIE Medical Imaging Proc.*, vol. 4681, pp. 483–491, 2002.

[16] ——, "The role of FPGA–based processing in medical imaging," *VMEbus Systems*, Apr. 2003.

[17] F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," *IEEE Transactions on Nuclear Science*, no. 3, pp. 654–663, 2005.

[18] J. Kole and F. J. Beekman, "Evaluation of accelerated iterative x–ray CT image reconstruction using floating point graphics hardware," *Phys. Med. Biol.*, vol. 51, pp. 875–889, Jan. 2006.

[19] J. Hornegger, "Moscow–Bavarian joint advanced student school — MB-JASS'2006," *www5.informatik.uni-erlangen.de/Lehre/WS0506/MB-JASS06/*.

[20] K. Mueller and F. Xu, "Practical considerations for GPU–accelerated CT," *IEEE International Symposium on Biomedical Imaging*, pp. SU–AM–OS3.3, Apr. 2006.

[21] C. Riddell and Y. Trousset, "Rectification for cone–beam projection and backprojection," *IEEE Transactions on Medical Imaging*, vol. 25, no. 7, pp. 950–962, July 2006.

[22] M. Knaup, W. Kalender, and M. Kachelrieß, "Statistical cone–beam CT image reconstruction using the Cell broadband engine," *IEEE Medical Imaging Conference Record*, pp. M11–422, 2006.