Journal of Advanced Engineering Trends
ISSN : 2682 - 2091

Vol. 40, No. 2. July 2021

http://jaet.journals.ekb.eg

AN ACCELERATION STRATEGYFOR GENERATING CONE-BEAM CT IMAGES BASEDON "MULTI-CORE" SYSTEMS

Marwa Mohamed Abbas*[1], Ammar Mostafa Hassan[2], Hassan Ali Younes[1]

[1]*Computers and Systems Engineering, Faculty of Engineering, Minia University, Minia, Egypt*

[2]*Arab Academy for Science, Technology and Maritime Transport, South Valley Branch, Aswan, Egypt*

**corresponding author E-mail:marwa.abas@mu.edu.eg*

Abstract

This paper displays a speeding up system for a cone-beam generation. We proposed a strategy to accelerate the Feld Kamp, Davis, and Kress (FDK) calculation based on a tiling process which is a compiler enhancement procedure. The test was the way by which to improve the use of the cache by strategies for loop blocking. Wiping out the cache misses as conceivable as possible is the guideline; reason/motivation behind the loop tiling.The method is represented and indicated by which method can be used as part of the recreation code, and in case of recognizing the reproduction output on the CPU "Multicore" condition; a specific model is also used. Exploratory results show that the intended Strategy achieves estimated 8.98 throughput per second (PPS) for reconstructing 512^3-voxel volume from 360 512^2 Pixel projections. This execution undertakes 3.37 speedup and 8.9 throughputs over than naive CPU based strategy that represents the infrastructure for parallelization code. Also, error measures results show that the proposed method did not affect the image quality reconstructed by the naïve method.

Keywords—*FDK, Loop tiling, Cone-beam, Filtered back projection*

Introduction

Computerized Tomography (CT) is a technique for imaging the cross-sections of an object using a series of x-ray measurements taken from different angles around objects [1, 2]. The computerized term refers to using the computer for processing combinations of many X-ray measurements taken from different views or angles to produce cross-sectional tomographic images or virtual "slices" for specific areas of a scanned object, let the user seeing inside the object without cutting. The tomographic reconstruction algorithm is responsible for producing those imaging slices, where it depends on mathematical reconstruction algorithms [3].

Most reconstruction algorithms can be classified as Analytical algorithms that are based on the filtered back-projection principle [4] and Iterative algorithms [5, 6]. Feldkamp-Davis-Kress (FDK) algorithm [7] is the most convenient reconstruction method and most CT scanners based on cone-beam used it as the standard reconstruction algorithm for reconstruction. FDK reconstruction composed of three stages: weighting, Filtering, and back-projection. Yet, the real-time imaging applications used FDK acts as a challenge due to its intensive computation and the massive amount of data through the reconstruction process.

As shown in Fig. 1[8], the patient or entity is positioned between the X- beam source and a detector.The majority of prior research has applied to accelerate FDK focused on optimizing reconstruction via different accelerators such as Cell-board Engine [9], Field-programmable-array FPGA [10, 11] has been widely studied and surveyed [12]. Graphics Processing Units (GPUs) which are considered as massively parallel engines for computation play an important role in reconstruction acceleration [13, 14].

Here in the paper, we will focus on the most intensively computing part of FDK, which is the stage of back-projection. The approach we used

Received:31 October, 2020, Accepted:3 Novamber, 2020

to encourage FDK technique acceleration is focused on arranging accesses to the data memory to avoid bottleneck memory bandwidth. This method/technique is called cache blocking. Since cache memory impacts the performance of the program when there are a lot of matrices. By using cache blocking to manipulate temporal localization, we create a method to optimize the FDK technique. Besides, this approach is evaluated using a computational model [15].

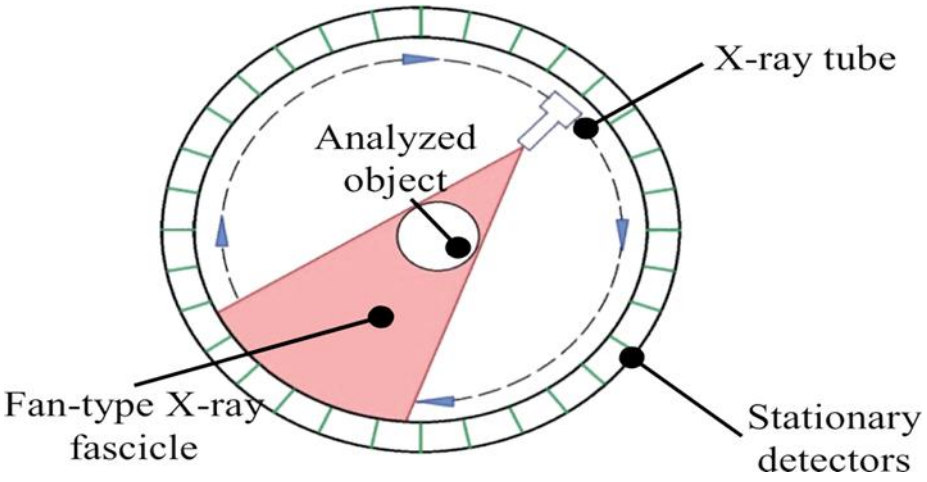


Fig.1 Principle of Cone Beam CT work reconstruction[8].

Whatever remains of the paper is sorted out as tails: we begin by presenting the related research in Section 2. Segment 3 includes a brief description of the strategy and Section 4 shows the implementation of the program for executing the scheme. Section 5 describes the outcomes and debates. Section 6 finishes the paper ultimately as a summary.

I. Related Work

Many methods have been used to quicken/ promote the reconstruction algorithms, where the efficient approach is the parallel technique.
Wang et al.[15] draw our attention to minimizing data transfer between the CPU and GPU during the execution of algorithms using OpenCLlanguage capabilities. They used page locked memory to store image slices, instead ofpageable memory. Where the pinned memory stores data to a contiguous memory page and eliminates the need for a page swapping operating system.

Sakamoto et al.[16]presented the FELDKAMP algorithm parallelization method on Cell Broadband Engine Architecture (CBEA), which has a programming scheme that differs from single-core architecture. They improved the execution time, but they had predicted that the program is insufficient to promote broadly speaking multiple successes.

Also [17]suggested an open-source toolbox for rapid cone-beam CT reconstruction, where the Reconstruction Tool Kit (RTK) allowed reconstruction on CPU processors/GPU cores. This independent software implemented to save time on the CUDA software / OpenCL application.

Others [18]provided an assessment of the reconstruction of the FDK cone beams on different architectures and their findings showed that the reconstruction on the Advanced Micro Devices (AMD) GPUs outperforms parallel C with the Open-multi processing system.

Pérez, et al. [19]used ideal contrasted GPU programming models and successive calculations in which the reconstruction time was reduced by up to 3 sets of size while preserving image quality.

A recent study by [20]concluded that proposed and tested an iterative algorithm for rapid CBCT image reconstruction using under sampled measurement data and reduced machine memory. And the result upon using the reduced number of machine memories, the proposed algorithm performed effective and fast computation of CBCT reconstruction.

Donghyun et al.[21] reconstructed volumetric 3D images through the back-projection algorithm that accelerated by custom GPU software, where it was 40 times faster than the CPU.

Authors[22] introduced a review article which systematically analyzed aspects of their settings, object models, imaging geometries, and algorithms in the currently publicly accessible CT image reconstruction open source toolkits. Deep learning reconstruction networks and open codes are also reviewed as the third group of reconstruction algorithms, in addition to analytic and iterative algorithms. This comprehensive review of the software platforms available to the public will help promote and improve CT research.

II. Material and methods

The second step was applying the Shepp-Logan filter [25]for minimizing noise propagation at the back-projection stage to the weighted projection images row-wise in Eq. (2), where R represents filter size to produce filtered projection Q_n, where $1 \leq n \leq N$ and N is the number of projections.

The last step was applying back-projection for filtered projections $\{Q_1, Q_2, \dots \dots Q_n\}$to obtain reconstructed volume F(x,y,z) in Eq. (3) for each angle θ_n from $1 \leq n \leq N$, where θ_n is the projection angel. Also, the second weighting factor W_2 during the back-projection step can be computed from Eq. (4). As u,v are discrete coordinates, and a bilinear interpolation is needed to be computed. This computation can be accomplished by Eq. (5) and Eq. (6) respectively.

$$W_1(r,v) = \frac{D}{\sqrt{D^2+r^2+v^2}} \quad (1)$$

$$Q_n(u,v) = \sum_{r=-R}^{R} \frac{2}{\pi^2(1-4r^2)} W_1(r,v)\, p_n(r,v) \quad (2)$$

$$F(x,y,z) = \frac{1}{2\pi N} \sum_{n=1}^{N} W_2(x,y,n) Q_n\big(u(x,y,n), v(x,y,z,n)\big) \quad (3)$$

$$W_2(x,y,n) = \left(\frac{d_n}{d_n - x\cos\theta_n - y\sin\theta_n}\right)^2 \quad (4)$$

$$u(x,y,n) = \frac{D(-x\sin\theta_n + y\cos\theta_n)}{d_n - x\cos\theta_n - y\sin\theta_n} \quad (5)$$

$$v(x,y,z,n) = \frac{DZ}{d_n - x\cos\theta_n - y\sin\theta_n} \quad (6)$$

B.Cache Blocking technique

Several ways to optimize cache efficiency. This is by raising the rate of miss. Miss rate decrease can be accomplished by the use of common methods such as increased cache space, higher associativity, and optimization of compilers.

We used a compiler optimization technique in our work that has several methods like loop optimization, but we used one of them called loop tiling/cache blocking.Cache blocking/loop tiling is a strategy designed to increase the reuse of information that decreases the temporal localization.

To demonstrate how to cache block has been used to boost overall performance, first address some important questions such as, what is the memory layout for matrices and whether the compiler does all the optimizations to work well on matrices. The matrices can be stored in two separate ways; row-major order/column-major order, where the 2D array can be stored in the main memory as a 1D array and accessed as a 1D array as seen in Figure 3. A matrix's row-major layout places the key push that is the 1st row in adjacent memory, the second line after it at that level, the third one too, etc.

By default, Fortran uses the major order column vs. C, which uses the major order row. But the use of loop tiling also poses several problems, where there are two essential ways to block nested loop by determining which loop will be tiled and which block size will fit well for the cache. The reason for using loop tiling has been the large size of matrices used to recreate the final volume in the process of back projection.

A. FDK Algorithm and Mathematical model

FDK is the first practical algorithm introduced by Feld Kamp-Davis-Kress[7] for CBCT reconstruction in 1984. This was a successfully reputable algorithm for a 3D volume reconstruction from multiple 2D projections. As shown in Fig.2 [23,24] full rotation around object or patient is taken by a scanner to obtain U×V pixel raw projections data before any processing p $\{p_1, p_2 \dots p_n\}$,, U and V represent u-axis and v-axis values for p data for a given voxel location and projection angle respectively.

The FDK scheme can be performed in three steps: pre-weight projection data according to weighting value in Eq. (1) where D is a distance source to the detector, and dn is the distance from the source to the center of the object.

Cache lacks capacity happens when blocks are disposed of from the cache because it can not accommodate all the work set needed for program execution, which means that less data is available in cache capability.

between a slow and fast memory.

- *tm* = time/slow memory operation.
- *f* = the number of arithmetic operations.
- *tf* = time/arithmetic operation << *tm.*

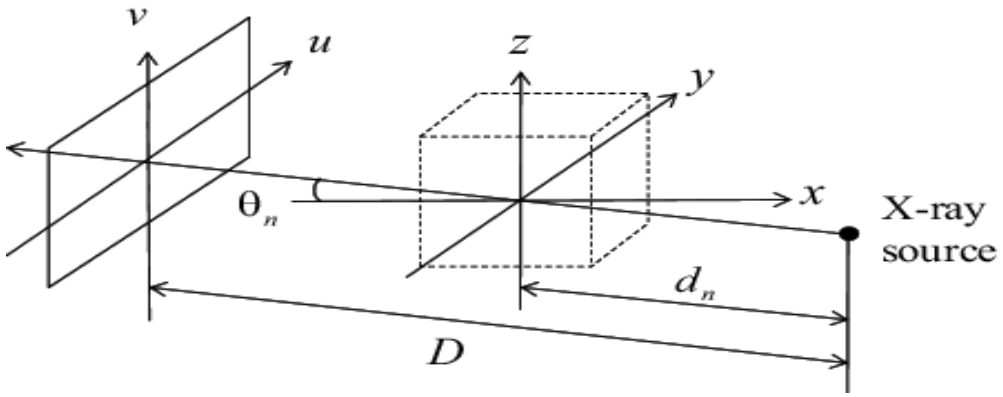


Fig.2 System coordinates for reconstructed volume xyzwhile u, v are detector coordinates [23,24]

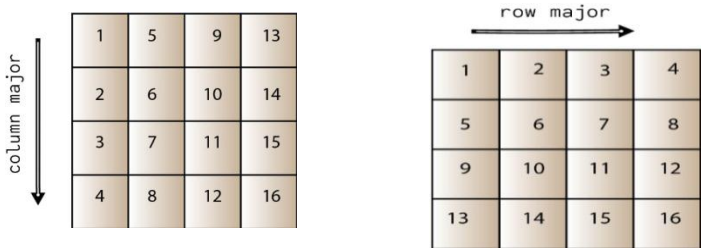


Fig.3 Row major order vs column-major order

C. Acceleration based cache blocking back-projection

We'll explain our proposed method of optimization in this section. As described above, loop tiling is used to remove cache mistakes and decrease overhead from loops. Now we used a simple model that will rely on some assumptions to evaluate the quality of implementation in terms of efficiency.

To simplify, assume that there are two memories, the hierarchy of fast memory (cache) and the slowmemory (main memory), taking into account the following parameters:

- *q = f / m* average flops/slow memory access.

When all data in the cache is established, then the minimum time needed time = f * tf. The actual time can be defined by : *f * tf + m * tm = f * tf * (1 + tm/tf * 1/q).* So, the larger *q*, the time close to ideal minimum *f * tf*.As shown in Algorithm 1 Fig. 4 (a),which represents standard back-projection algorithm without optimizations, the first loop iterates over theta, defining angles to the degree of the projection, and every projection is read weighted in this loop, and then filtered.

Then the projection detector co-ordinates U and V will be computed for each slice and determined then, the use of the bilinear the interpolation method is needed as an essential part of the FBP algorithm takes place in the back

projection phase.

Two last loops repeat over projection coordinates for final volume computation. This will createslices accumulation.

We used the tiling technique to boost the last intensive step back-projection of computation, which includes two inner loops. For performance analysis we presume the number of memory words to be referenced is calculated by $m=($read new slice array once n^2+read previous volume array n^2+previous volume array $n^2 = 3n^2$.

After arithmetic operation writes back volume array to slow main memory f =n^2. So the average no.of flops / slow memory access can be determined by $q=f/m=n^2/(3n^2)=1/3$.

After loop blocking usage, as shown in Algorithm 2. Matrices(slices) are blocked by B×B to fit in cache.m=. (read new slice array once n^2/B+ read previous volume array n^2/B + previous volume array $n^2/B=3\ n^2/B$.

So, the cache miss rate decreased to 3 n^2/B and the total number of arithmetic operations $f=n^2$and q the average number of flops per slow memory access can be calculated as:

f/m=(n^2/3n^2)/B=B/3.

It is clear that q is increased after loop tiling from this equation which defines the total actual time for accessing data is

$f * tf + m * tm = f * tf * (1 + tm/tf * 1/q)$. It is clear from the equation that the greater value of q the total actual time becomes close to optimal value $f * tf$.

That implies the cache contains all the data required for arithmetic operation. There will also be another gain in reducing memory flow.A load of memory traffic can be reduced by a greater *B*. This means that of block size *B* will achieve better efficiency, but there is a limit to reading two matrices of the old slice *(i,j)* by the size of *B*×*B*and a new slice of *(i,j)* size of B×B, so total size must fit in *2B^2* cache.

Algorithm 1: Unoptimized back-projection kernel

```
1.  FOR theta 1:360 //for each projection
2.      Readingprojection from sinogram
3.          Projection Weigting
4.              Projection Filtering
5.  s, t, u Computation
6.  FOR Z slices
7.  Compute v
8.  Bilinear interpolation
9.                  FOR 1: n 1st projection coordinate //back
    projection
10.                 FOR 1: n 2nd projection
    coordinate
11.     New volume=old volume+new_slice
12. END
13.             END
14.         END
15.             END
```

(a)

Algorithm 2: Tiled back-projection Algorithm

```
1.  FOR theta 1:360 //for each projection
2.      Readingprojection from sinogram
3.          Projection Weigting
4.              Projection Filtering
5.  s, t, u Computation
6.  FOR Z slices
7.  Compute v
8.  Bilinear interpolation
9.              FOR 1: n 1st projection coordinate //back
    projection
10.         Tiled an inner loop with B blocking factor
11.             FOR 1: n 2nd projection
    coordinate
12.     New volume=old volume+new_slice
13. END
14. END
15. END
16.             END
```

(b)

Fig 4. Pseudocode of the proposed method. (a) unoptimized back-projection code. (b) tiled back-projection code

Suppose the size of the memory cache is *Msize* so there is a limitation for *B* as $Msize \leq 2B^2$. This means that is $B \leq (Msize\ /2)^{1/2}$. So, Choosing the blocking factor B greatly affects tiling efficiency. The option of the optimal blocking factor is known to be multiple factors. Such as the size of the cache, and the number of matrices. While several researchers find different relationships that relate factor blocking and cache size.

In [26] they evaluated various methods for selecting optimal B, where they note that for several cases B can be √c as good as possible, but the rate of cache miss rate is 10 times than the optimal. After the blocking factor has been expressed as a fraction of the cache size (8 percent, 6 percent, 12 percent), the output is better. It's also difficult to continuously select

the blocking factor.

Yet, possibly, trial-and-error is quicker. Also in [27], the researchers reported that a block size selection algorithm calculates a set of optimal block sizes for L1, L2 cache rates are taken into consideration.

But in our application, it was found that experimentally using a block size equal to 50 percent of L2 cache size results in minimum execution time on a multicore device for the application of acceleration cone beams.

III. Implementation

To illustrate the Feldkamp-Devis-Kress algorithm for CT according to cone-beam geometry, the basic main stages are implemented in a simple shepp-Logan phantom model.FDK 's aim is to recreate cross-section images from a variety of projections. To demonstrate how to form projections from a sample image model and then reconstruct the image from the projections, a sinogram is generated for a head shepp-Logan phantom as shown in Fig.5 [28] by using MATLAB.

After the generation of Sinogram as shown in Fig. 5 which represents the forward projection, and stores each observed event corresponding to each angle, each projection weighted.The filter was conceived in the frequency domain and used the principle of convolution. The Fast Fourier Transform (FFT) has been accomplished with the use of the FFTW library package[29]for projection convolution. Convolve each row of projections by multiplying each row by its FFT. Defines the real elements and the imaginary components, and then uses FFTW to apply IFFT, separate extract the real parts.

Bilinear interpolation is subsequently done to get a realistic picture. We've tuned B blocking factor size with cache size and matrices size for using the tiling method. Using different methods to estimate B size such as $(Msize / 2)^{1/2}$ described above and using different cache size fractions of the form. Trial and error are the most effective for improved performance.The most powerful ratio was for our application about matrices scale blocking factor with scale 0.5 L2.

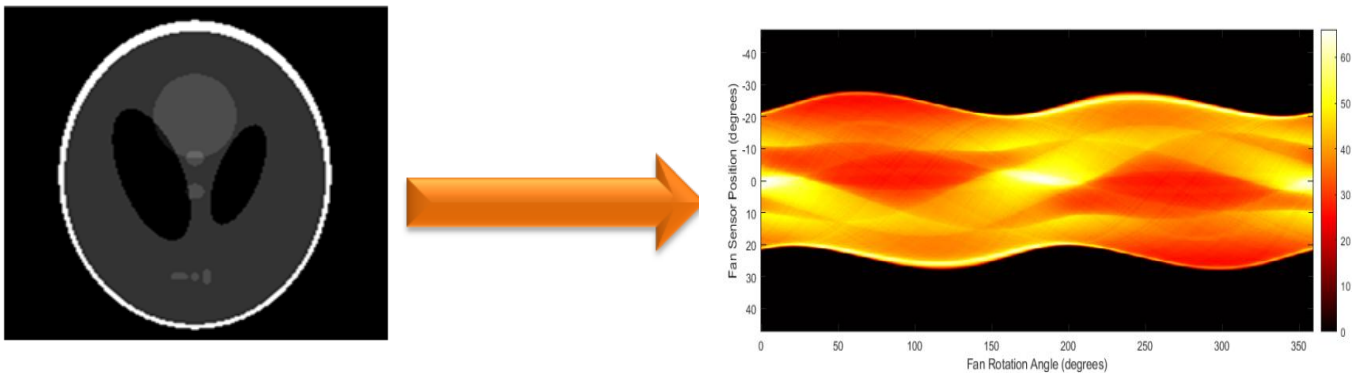


Fig. 5 Sinogram generated for Shepp –Logan –phantom from MATLAB[28]

V. Results and Discussion

We demonstrate FDK algorithm implementation on multicore "CPU" architecture. We use a desktop PC fitted with Intel core2due corei75500U,8 GB DDRAM, and 32 KB L1 cache. Our implementation operates on an operating system of 64-bit Windows 7.

We use the Shepp-Logan phantom[6] for evaluation, this is a computational phantom widely used to validate tomographic reconstruction algorithms,

The data size was U = V = 512, where U, V represents the coordinate of the detector. Projection number K = 360, and R = 256. The filtering stage was accomplished using the library package FFTW[29].

Fig.6 iullatrates the discrepancy between the initial and rebuilt accelerated one. Fig.6 (b) represents the reconstructed image from only 360 projection which most closely resembles the original image in Fig.6 (a). And the difference attributed to some artifacts from back-projection.

As demonstrated in Table.1, the experimental results show that the execution time on the multicore machine "CPU" to recreate 512^3 voxel volume from 360

512^3projections were 40.2s. Which achieves speed-up factor 9.97,3.34 and factor 3.37 over[30][24],[31]. Our proposed method achieves 8.955 PPS (Projection per second) in terms of the throughput.

To quantitatively test our proposed method, three error measures were used for our proposed method evaluation in terms of image quality as used in work[30, 32].Mean-squared-error (MSE) between two images the original phantom *p(x,y)* (output) and reconstructed image $\hat{p}(x,y)$(input). The MSE is given by :

$$MSE=\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{p}(x,y)-p(x,y)]^2 \quad (7)$$

Where M and N are images dimension in pixels. The second measure is Root –mean-squared error (RMSE) is given by :

$$RMSE=\frac{1}{MN}\sqrt{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{p}(x,y)-p(x,y)]^2} \quad (8)$$

Finally, Normalized-mean-square error (NMSE) is given by:

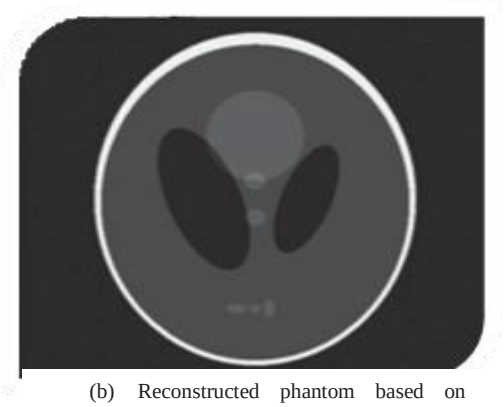
$$NMSE=\frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{p}(x,y)-p(x,y)]^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{p}(x,y)]^2} \quad (9)$$

Moreover, the findings of the reconstruction as shown in Table 2 also show that the proposed method will achieve the same accuracy as the conventional method.

Table1. Comparison of the proposed method with others in case of performance in multicore systems

Method	Hardware	Reconstruction time (s)	Throughput (PPS)	Speed up
A method based CPU[30]	4.2 GHz 4-core Intel Core™ i7-7700	**400.97*S***	**0.897**	**9.97**
A method based CPU[24]	Core 2 Quad Q6700 CPU	**134.4 *S***	**2.67**	**3.34**
A method based CPU[31]	Xeon 3.06 GHz	**135.4 *S***	**2.65**	**3.37**
Proposed method	Core2due-Intel corei75500 U	**40.2 *S***	**8.955**	

(a) Original Shepp-Loganphantom



(b) Reconstructed phantom based on accelerationFDK

Fig. 6 (a) Original Shepp-Logan phantom vs.(b) reconstructed after acceleration

Table2. Error comparison among naive method and optimized method on multicore.

	method	method
MSE	0.0611	0.0611
RMSE	0.2471	0.2471
NMSE	0.2471	0.2471

VI.Conclusion

We presented an efficient model for cone-beam reconstruction. This model is based on one of the familiar reconstruction algorithm called Feld-Kamp-Davis. We designed the main stages for filtered back-projection algorithm weighting, filtering, and back projection.

This program is often bounded in memory because of the large number of matrices that will be passed between the CPU and the memory. So, our contribution has been to speed up the projection step back by writing a cache-friendly code. This approach is achieved using an important technique of compiler optimization **.** This optimization is cache blocking or a loop tiling scheme which is one of the essential loop transformations to minimize overhead loops. Blocking factor size is a challenge in loop tiling. Experimentally, 50 percent of L2 cache size was found to have achieved minimum execution time. We defined how the overall execution time improves with this process.

Future work should focus on the blocking factor selection algorithmwhich will lead us to this result. Also improvement would have been accomplished by using GPUs which give an approach to speeding up intensive computational jobs such as the reconstruction of CT images.Overall we will present an efficient method of using memory.

References

[1] W. C. Scarfe and A. G. Farman.What is cone-beam CT and how does it work?*Dental Clinics of North America* 2008;vol. 52.

[2] S. Patel, C. Durack, F. Abella, H. Shemesh, M. Roig, and K. Lemberg.Cone beam computed tomography in endodontics–a review.*International endodontic journal* 2015;vol. 48.

[3] F. Natterer and F. Wübbeling, *Mathematical methods in image reconstruction* vol. 5: Siam, 2001.

[4] G. Zeng.Image reconstruction—a tutorial.*Computerized medical imaging and graphics* 2001;vol. 25.

[5] P. P. Bruyant.Analytic and iterative reconstruction algorithms in SPECT.*Journal of Nuclear Medicine* 2002;vol. 43.

[6] G. L. Zeng, "Revisit of the ramp filter," in *2014 IEEE Nuclear Science Symposium and Medical Imaging Conference (NSS/MIC)*, 2014, pp. 1-6.

[7] L. Feldkamp, L. Davis, and J. Kress.Practical cone-beam algorithm.*JOSA A* 1984;vol. 1.

[8] D. I. Tudor, S. Pastrama, and A. Hadar.The use of computed tomography and ultrasonic imaging for assessment of defects in plates made of a polyesteric resin.*Engineering Transactions* 01/01 2014;vol. 62.

[9] H. Scherl, M. Koerner, H. Hofmann, W. Eckert, M. Kowarschik, and J. Hornegger, "Implementation of the FDK algorithm for cone-beam CT on the cell broadband engine architecture," in *Medical Imaging 2007: Physics of Medical Imaging*, 2007, p. 651058.

[10] D. Held.Analysis of 3D Cone-Beam CT Image Reconstruction Performance on a FPGA.2016;

[11] J. Li, C. Papachristou, and R. Shekhar.An FPGA-based computing platform for real-time 3D medical imaging and its application to cone-beam CT reconstruction.*Journal of Imaging Science and Technology* 2005;vol. 49.

[12] H. Scherl, M. Kowarschik, H. G. Hofmann, B. Keck, and J. Hornegger.Evaluation of state-of-the-art hardware architectures for fast cone-beam CT reconstruction.*Parallel computing* 2012;vol. 38.

[13] L. Shi, W. Liu, H. Zhang, Y. Xie, and D. Wang.A survey of GPU-based medical image computing techniques.*Quantitative imaging in medicine and surgery* 2012;vol. 2.

[14] P. Despres and X. Jia.A review of GPU-based medical image reconstruction.*Physica Medica* 2017;vol. 42.

[15] B. Wang, L. Zhu, K. Jia, and J. Zheng, "Accelerated cone beam CT reconstruction based on OpenCL," in *Image Analysis and Signal Processing (IASP), 2010 International Conference on*, 2010, pp. 291-295.

[16] M. Sakamoto, H. Nishiyama, H. Satoh, S. Shimizu, T. Sanuki, K. Kamijoh, *et al.*An implementation of the feldkamp algorithm for medical imaging on cell.*IBM White*

Paper 2005;

[17] S. Rit, M. V. Oliva, S. Brousmiche, R. Labarbe, D. Sarrut, and G. C. Sharp, "The Reconstruction Toolkit (RTK), an open-source cone-beam CT reconstruction toolkit based on the Insight Toolkit (ITK)," in *J Phys Conf Ser*, 2014, p. 012079.

[18] M. Leeser, S. Mukherjee, and J. Brock.Fast reconstruction of 3D volumes from 2D CT projection data with GPUs.*BMC research notes* 2014;vol. 7.

[19] T. Valencia-Pérez, J. Hernández-López, E. Moreno-Barbosa, and B. de Celis-Alonso.Study of CT Images Processing with the Implementation of MLEM Algorithm using CUDA on NVIDIA'S GPU Framework.*Journal of Nuclear Physics, Material Sciences, Radiation and Applications* 2020;vol. 7.

[20] Y. Zhu, Y. Liu, Q. Zhang, C. Zhang, and X. Gao.A fast iteration approach to undersampled cone-beam CT reconstruction.*Journal of X-ray science and technology* 2019;vol. 27.

[21] G. Di Domenico.Fast Cone-beam CT reconstruction using GPU.2015;

[22] L. Shi, B. Liu, H. Yu, C. Wei, L. Wei, L. Zeng, *et al.*Review of CT image reconstruction open source toolkits.*Journal of X-Ray Science and Technology* 2020;

[23] Y. Okitsu, F. Ino, and K. Hagihara, "Accelerating cone beam reconstruction using the CUDA-enabled GPU," in *International Conference on High-Performance Computing*, 2008, pp. 108-119.

[24] Y. Okitsu, F. Ino, and K. Hagihara.High-performance cone beam reconstruction using CUDA compatible GPUs.*Parallel Computing* 2010;vol. 36.

[25] L. A. Shepp and B. F. Logan.The Fourier reconstruction of a head section.*IEEE Transactions on nuclear science* 1974;vol. 21.

[26] M. D. Lam, E. E. Rothberg, and M. E. Wolf.The cache performance and optimizations of blocked algorithms.*ACM SIGOPS Operating Systems Review* 1991;vol. 25.

[27] E. Athanasaki and N. Koziris.Fast indexing for blocked array layouts to reduce cache misses.*International Journal of High Performance Computing and Networking* 2005;vol. 3.

[28] MathWorks, "MATLAB R 2019 a," ed, 2019.

[29] M. Frigo and S. G. Johnson.The design and implementation of FFTW3.*Proceedings of the IEEE* 2005;vol. 93.

[30] S. Zhang, G. Geng, and J. Zhao.Fast parallel image reconstruction for cone-beam FDK algorithm.*Concurrency and Computation: Practice and Experience* 2019;vol. 31.

[31] M. Kachelriess, M. Knaup, and O. Bockenbach.Hyperfast parallel-beam and cone-beam backprojection using the cell general purpose hardware.*Medical Physics* 2007;vol. 34.

[32] D. Soimu, I. Buliev, and N. Pallikarakis.Studies on circular isocentric cone-beam trajectories for 3D image reconstructions using FDK algorithm.*Computerized Medical Imaging and Graphics* 2008;vol. 32.

استراتيجية تسريع لتوليد صور الأشعة المقطعية المخروطية باستخدام نظام متعدد النواه

ملخص البحث:

الهدف من البحث هو تسريع الصورة المتكونة من الحزمة المخروطية للأشعة المقطعية والتي يتم استخدام خوارزمية فيلد كامب في تكوين الصورة المقطعية باستخام عدة مساقط . وقد تم استخدام تقسيم الدورات والتي تعتبر واحدة من اهم التقنيات المستخدمة في محسنات المترجم وبالتالي إلى تحسين استخدام الذاكرة المؤقتة والتقليل من معدل فقد المعلومات في ذاكرة التخزين المؤقت. وبذلك تم اعادة بناء البرنامج المسئول عن توليد الصورة المقطعية من عدة مساقط للاشعة المخروطية على وحدة المعالجة المركزية. والنتائج العملية أوضحت ان الطريقة المستخدمة في تحسين أداء الذاكرة المؤقتة حققت ٨.٨٩ صورة في الثانية كمعدل معالجة الصور باستخدام وحدة المعالجة المركزية لتكوين صورة بابعاد 512^3 من ٣٦٠ مسقط بابعاد 512^2 .وتم تحسين الوقت الكلي للتنفيذ البرنامج بمعدل ٣.٣٧ مقارنة بطريقة تكوين الصور التقليدية على وحدة المعالجة المركزية. وأيضا تم استخدام مقاييس الخطأ لاختبار كفاءة الصورة المتكونة من تحسين كفاءة الذاكرة المؤقتة فإنها لم تؤثر على كفاءة الصورة الأصلية.