Efficient CT Image Reconstruction in a GPU Parallel Environment

Tomás A. Valencia Pérez1, Javier M. Hernández López1, Eduardo Moreno-Barbosa1, Benito de Celis Alonso1, Martín R. Palomino Merino1, and Victor M. Castaño Meneses2

Corresponding Author:
Victor M. Castaño Meneses, PhD
Molecular and Materials, Engineering Department, Universidad Nacional Autónoma de México, Queretaro, Mexico
E-mail: vmcastano@unam.mx

Key Words: Computed tomography, iterative algorithms, GPU, parallelization, reconstruction, image quality

ABSTRACT
Computed tomography is nowadays an indispensable tool in medicine used to diagnose multiple diseases. In clinical and emergency room environments, the speed of acquisition and information processing are crucial. CUDA is a software architecture used to work with NVIDIA graphics processing units. In this paper a methodology to accelerate tomographic image reconstruction based on maximum likelihood expectation maximization iterative algorithm and combined with the use of graphics processing units programmed in CUDA framework is presented. Implementations developed here are used to reconstruct images with clinical use. Timewise, parallel versions showed improvement with respect to serial implementations. These differences reached, in some cases, 2 orders of magnitude in time while preserving image quality. The image quality and reconstruction times were not affected significantly by the addition of Poisson noise to projections. Furthermore, our implementations showed good performance when compared with reconstruction methods provided by commercial software. One of the goals of this work was to provide a fast, portable, simple, and cheap image reconstruction system, and our results support the statement that the goal was achieved.

INTRODUCTION
X-ray computed tomography (CT) is a nondestructive technique in which a source of X-rays (ionizing radiation) revolves around an object of interest, generating axial images of its structure. CT is today an indispensable tool in medicine for the diagnosis of multiple diseases (1). Since its introduction in 1970, there are about 30,000 CT scanners in the world and its number continues to increase exponentially (1). Even if CT techniques represent one of the greatest developments in the field of X-rays in the past 50 years (1), there is still room for improvement. Some of the proposed lines of work to achieve this objective include reduction of patient exposure time, reduction of acquisition and processing times, development of new techniques with new functionalities, and cost reduction (2). Some of the solutions to these points have so far focused on the development of higher performance hardware that will lower costs for and dose delivery to patients without compromising the effectiveness of the diagnosis. Work has also been done on the use of iterative methods for image reconstruction, reducing processing times and reducing the dose necessary for the acquisition of images with diagnostic value.

Current CT scanners use filtered back projection (FBP) for regular image reconstruction (3, 4). This technique is mathematically based on the radon transform (RT) (5, 6). In this technique, the image of an object is reconstructed from a set of X-ray projections of the aforementioned object (7). There are many computational algorithms that can be used to solve the RT. The theoretical framework of reconstruction methods can be found in the literature (2, 8, 9). Nevertheless, these are usually classified into 2 methodologies: analytical reconstruction methods and/or iterative reconstruction methods. In the first method, the use of FBP algorithms has been the standard to date (10), because it is a simple and fast reconstruction algorithm with low computational cost. However, it is prone to the appearance of artifacts and sometimes has low spatial resolution. In addition, it requires a large number of projections without noise for the reconstruction solution to be of quality. This is far from real-life situations, in which only a finite set of projections will be available. Moreover, “approximate” reconstruction methods are also known as iterative algorithms. These are less sensitive to incomplete sets of data, noise, and artifacts (10, 11). This translates into better quality of the reconstructed images. In addition, by requiring less information for reconstruction, they can use fewer projections/radiation by reducing the doses delivered in a study to a patient. For a review of the mathematics behind these methods, see the
The article is structured as follows: the next section presents the hardware used. This is followed by the results section in which different image quality parameters were calculated for a series of clinical and phantom images based on the reconstruction time and the number of iterations. In a couple of cases, Poisson noise was added to blur the original images and perform an evaluation of the effect of noise on the different reconstruction parameters. Finally, results of the reconstructed images were compared with those of commercial software.

**MLEM Algorithm**

Lange et al. (26) presented the MLEM reconstruction algorithm for emission and transmission tomography. The MLEM algorithm was derived from the following mathematical expression:

\[
\mathbf{x}_{i+1} = \frac{\mathbf{x}_i}{\sum_{i=1}^{N} \frac{a_{ij}y_i}{\sum_{l=1}^{M} a_{il}x_l^i}}
\]  

Here \(x_j^i\) is a pixel, \(j\) is the image in the \(r\)-th iteration, \(y_i\) is the interval \(i\) of the projection measured given by the CT scanner, \(a_{ij}\) is the system matrix \(M \times N\) whose coefficients connect the \(x_j\) values of the image with projections \(y_i\) and describe the probability of detecting a photon in a pixel \(j\) for the \(i\) bin.

All algorithm implementations developed in this study required of an \(a_{ij}\) system matrix that was used in the resolution of the MLEM algorithm given in the literature (1). Creating a system matrix \(a_{ij}\) depended on the number of projection lines \(y_i\), the number of projection angles and the size of the image to be reconstructed \(n \times n\). This matrix could be calculated in different ways, and in this project, the methodology given in the literature (27-29) was used to generate and model the matrix in MatLab.

**Implementation of the Algorithm for CT Image Reconstruction**

As already mentioned, to compare processing times and performance, the MLEM algorithm was implemented, first serially, in MatLab and C languages, and then in parallel using CUDA on the graphics card.

*Serial Implementations.* Two versions of the MLEM algorithm were developed, one in MatLab and one in C (Serial-MatLab and Serial-C, respectively). The different processes that are part of the programmed algorithms are described as follows:

1. **Input Data:** In a first step, the files provided by the CT scanner were loaded. This included the system matrix \(a_{ij}\) and the measured projections \(y_i\). These structures were called *matrix* and *vector*, respectively. The \(a_{ij}\) matrix and the vector \(y_i\) provided information such as the number of rows and columns of the system matrix or the number of elements of the projection vector \(y_i\).

2. **Initial Image Estimation:** To reduce the number of iterations, an initial estimate of the image was used, \(-x_0^i\). It was created either with all its values equal to the average of the projection data or by setting its values at an intensity between 1 and 255 (grayscale).

3. **Forward-Projection:** The first iteration was performed on the image with an initial estimate that simulated the projections of a given \(x_0^i\). This was required for the calculation.

**METHODOLOGY**

First, the MLEM algorithm was implemented in series, both in MatLab* (MathWorks, Natick, MA) and in C. Second, the same algorithm was implemented in parallel using CUDA. Third, different image quality parameters were calculated for a series of clinical and phantom images based on the reconstruction time and the number of iterations. In a couple of cases, Poisson noise was added to blur the original images and perform an evaluation of the effect of noise on the different reconstruction parameters. Finally, results of the reconstructed images were compared with those of commercial software.
of \( y_i^{\text{simulated}} = \sum_j a_{ij} x_j^0 \) to determine the corresponding projection values.

4. Comparison with the measured data: The comparison step was performed by dividing the original projection data \( y_i \) with the data from simulated projections \( y_i^{\text{simulated}} \), obtaining a relation \( Y_i = \frac{y_i}{y_i^{\text{simulated}}} \). Here \( Y_i \) formed a multiplicative correction factor for each projection.

5. Back-Projection: The calculated \( Y_i \) was back-projected to obtain a correction factor \( X_j \) for the initial image. This was done with the following system calculation \( X_j = \sum_i a_{ij} Y_i \).

6. Normalization: The correction factor \( X_j \) was normalized before being multiplied by \( x_j^0 \). It was divided by a weighting term based on the system model to apply the desired intensity of each image correction factor. This produced \( e_j = X_j / \sum_i a_{ij} \), where \( e_j \) represented the normalized correction factor.

7. Image Update: The initial estimation of image \( x_j^0 \) was multiplied by the normalized correction factor \( e_j \), obtaining the new image estimation \( x_j^1 \), which was then fed into the algorithm in the initial image estimation step.

These processes were executed iteratively until the relative error between the estimated and the actual image reached a value <5%. To verify that this behavior was maintained, up to 1000 iterations were performed.

From the serial implementation of the algorithm, the performance was measured for each of the processes. It was found that the processes that consumed the most resources were the steps of Forward-Projection and Back-Projection. Between them, they used ~75% of computational time.

Parallel Implementations. Two versions were developed: parallel using GPU (Parallel-GPU) and parallel using GPU but optimizing data transfer (Parallel-Opt). Both implementations resolved equation (1) using both C and CUDA. CUDA was used to exclusively develop the Forward-Projection and Back-Projection processes, known for being the ones that consume the most computing resources.

The calculations performed on the data structures were handled in C by four kernels (Back-Projection, Forward-Projection, Normalization, Image Update), which distributed the data to the blocks of threads. This was done in similarity to the serial case. The different processes of the parallel versions are described as follows:

1. Forward-Projection module: To this end, a Forward-Projection module, using resources from the cuBLAS library from CUDA, was programmed (30, 31). No Forward-Projection or Back-Projection modules exist per se in cuBLAS, so they were developed specifically by authors for this project. Matrix \( a_{ij} \) was loaded to the Host and transferred to the Device; vector \( x_j^0 \) was generated in the Host and also transferred to the Device. Matrix \( a_{ij} \) was partitioned in grids of threads blocks. Each thread read a row of the \( a_{ij} \) matrix and multiplied it by vector \( x_j^0 \) to obtain a vector in the Device. The resulting \( y_i^{\text{simulated}} \) vector was then transferred to the Host.

2. Back-Projection module: As in the previous process, the Back-Projection module was launched on the Device. This module required most of the computational resources for the entire reconstruction process, as it consisted of 2 operations: calculation of the transposition of \( a_{ij} \) and that of its product with vector \( Y_i \).

3. Optimization of GPU computing time. The main data transfers between Host and Device were carried out when the Forward-Projection and Back-Projection processes were executed. To improve the performance of the programs, the transfer of data between the RAM of the CPU and the global memory of the GPU was handled by joining the Forward-Projection, comparing the measured data and Back-Projection modules in one function, and maintaining data calculations in the GPU. This created the optimized parallel version (Parallel-Opt).

As in the serial case, these processes were executed iteratively until the relative error between the estimated and the actual image reached a value <5%.

Image Quality Parameters

In this study, 2 clinical CT images were initially reconstructed along with a 512 × 512 image of the Shepp–Logan phantom (SLP) (32). The first clinical image was an axial cut of the brain that covered the cerebellum, as well as the nasal and auditory cavities (33). The second clinical image represented an abdominal axial cut through the lungs, heart, pancreas, and liver (33). The matrices of both CT images were 512 × 512 with a dynamic range of 8 bits per pixel, with a system matrix of 23944 × 261244 and a vector projection of 23944 × 1 for a 5° sweeping angle.

To evaluate the quality of the reconstructed images, different image quality parameters were used. The main parameters studied were: contrast-to-noise ratio and signal-to-noise ratio (34). The first indicated how well 2 neighboring tissues could be distinguished. The latter how strong a signal was with respect to background noise. Other image quality parameters evaluated were: mean squared error (MSE) (35), peak signal-to-noise ratio (PSNR) (36) or structural similarity index (SSIM) (37). MSE provided a measurement of the differences between a reconstructed image and a reference image. The lower the MSE value, the better the result. The PSNR described the relationship of the maximum possible power of a signal with the power of noise corruption. It was represented on a decibel scale, and a high PSNR value meant that the reconstruction was of high quality. Finally, the SSIM evaluated the following 3 image parameters: luminance, contrast, and structural correlation. The SSIM was used to measure the similarity between 2 images, namely, original CT image and reconstructed images. SSIM values were normalized between 0 and 1; being 1 the situation in which both images were equal.

All image quality parameters were measured for all the CT images and for the 4 implementations of the algorithm developed and the two versions of TIGRE. In addition, 2 scenarios were considered, in which 50% Poisson noise was added to the images under study to assess the impact of noise on the reconstruction time and quality.

Comparison with Commercial Software

To compare the implementations developed in this work for the MLEM algorithm with those of an external (commercial) code, the TIGRE MatLab library was used (38). From it, 2 iterative
Tomographic reconstruction toolboxes based on GPUs were used, namely, the OS-SART algorithm that is used to solve reconstructions of the conical beam CT images using simultaneous algebraic reconstruction techniques with ordered subsets, and the ASD-POCS algorithm that is based on the most pronounced adaptive descent projection in convex subsets (39).

**Hardware and Software**

Serial and parallel implementations were developed and executed on a laptop with an Intel Core i7-5500U processor running at 2.40 GHz and equipped with an NVIDIA GeForce 840 M graphics card, with a global memory of 2048 MB and 384 CUDA cores on an Ubuntu operating system 15.04 (40). MATLAB R2017a was used (27) as well as GCC 4.9.2 (41) for the serial part. CUDA C 7.5 (31) was used for the parallel calculations. The Open Source Computer Vision library (OpenCV 3.2.0) was used to calculate image quality measurements (42, 43).

**RESULTS**

Computer algorithms and code developed here will be available after petition to authors.

---

**Figures 1 to 3** present the results obtained for the 4 implementations of the algorithm developed here (Serial-MatLab, Serial-C, and the 2 parallel versions) and the external test software (2 versions).

Images in Figure 1 correspond to a reconstruction with Parallel-OpT software with a maximum of 1000 iterations. In the first row, SLP and two 512 \times 512 CT images are presented. In the following row 4 128 \times 128 regions of interest of the previous images are presented. These regions were selected randomly by authors. The original image of the SLP (first column) is shown in the first row of Figure 1. In the second column, the reconstructed image “iteration 64” is shown, which will be referred from now on as “the best” and called iteration64. Number 64 just quantifies the number of iterations necessary to reach it. In the third column, the reconstructed image called “functional” is shown. In this case it is called “iteration 35,” and it is characterized by the fact that the differences in its quality parameters with those of the “the best” are <50%. The last column shows the “functional” image with the Poisson noise. The CT images of the central row correspond to the axial section of the brain, while the abdominal axial section is presented in the last row of Figure 1. The data of the last 2 rows are presented in the same scheme as in the SLP image.
The values of the different quality parameters of each image are presented in Table 1. The minimum number of necessary iterations to obtain the given image quality value and the time taken to achieve it are also presented.

Figure 2 shows the mean values for the different quality parameters of the reconstructed images of Figure 1: MSE, PSNR, and SSIM. Table 2 vs. number of iterations presents the quantification of these values. Images obtained from reconstruction had 512 × 512 matrices formed from CT projections that varied between 0° and 180° in 5° steps (views). Initially, the quality values of the reconstructed images were calculated for all serial and parallel implementations; however, the differences in the absolute values of these quality parameters were <0.001%.

In addition, the dependence between the optimal number of views/projections for image reconstruction in the implementation was studied. These results are presented as complementary material (see online supplemental Figure 1). In this analysis, the image quality parameters, as well as the time calculations, were compared based on the number of iterations and the number of views. It was found that the use of jumps of 5° per view to cover the entire field of vision, as well as 50 to 100 iterations, produced reliable results without extending the calculation time while obtaining high-quality image parameter values.

It is worth mentioning that when the field of view was covered either with projections that varied between 0°–180° and 0°–360°, the reconstructions of the different programs produced very similar results (regarding image quality parameters). Because of this and from this point, only reconstructions with a system matrix that varied from 0° to 180° were taken into account.

Figure 3 presents the average computation time for the 4 implementations/programs developed by the authors, as well as the 2 versions based on TIGRE. It also shows the results with added Poisson noise. The data are presented versus the number of iterations.

It can be observed that the differences in computation time between the Parallel-OpT and Serial-C implementations were of the order of 12 times faster for the parallel solution. The Parallel-OpT was 170 times faster than Serial-MatLab. Parallel-OpT and Parallel-GPU implementations differed by a factor of 2 when computer processing time was considered.

Finally, to test the stability and effectiveness of these programs, data from a clinical study were reconstructed using the Parallel-OpT implementation. Further 101 CT images, corresponding to a single axial cut of the pelvis, were obtained by each volunteer. Data were downloaded from The Cancer Genome Atlas Sarcoma database (TCGA-SARC) (33); therefore, no ethical permission was needed to perform this study on our side. In each image the SSIM was calculated as in previous sections. The SSIM

| Table 1. Quality Parameters from Figure 1 |
|-----------------------------------------|
| CNR | SNR | SSIM | Shepp-Logan Phantom |
| ---- | ---- | ---- | --------------------|
|     | PSNR | MSE | Iterations | Time (s) |
| Original | 0.98 | 5.17 | 31.49 | 46.1 | 64 | 408.31 |
| The best | 5.10 | 0.99 | 31.59 | 46.2 | 64 | 408.31 |
| Functional | 4.90 | 0.99 | 28.85 | 84.8 | 35 | 55.99 |
| Functional + Noise | 0.98 | 28.60 | 88.0 | 35 | 54.96 |

| Head |
|------|
| Original | 0.35 | 3.54 | 27.84 | 107.0 | 46 | 295.01 |
| The best | 3.51 | 0.98 | 25.43 | 186.2 | 28 | 44.93 |
| Functional | 3.60 | 0.97 | 24.52 | 229.6 | 28 | 47.24 |
| Functional + Noise | 0.96 | 27.84 | 107.0 | 46 | 295.01 |

| Abdomen |
|--------|
| Original | 0.98 | 4.63 | 41.65 | 4.45 | 24 | 155.62 |
| The best | 4.79 | 0.98 | 41.50 | 4.60 | 24 | 39.04 |
| Functional | 4.71 | 0.98 | 39.36 | 7.53 | 24 | 39.18 |
| Functional + Noise | 0.96 | 39.36 | 7.53 | 24 | 39.18 |
Figure 2. Image quality parameters: MSE, PSNR, and SSIM versus number of iterations. Image parameters were obtained after reconstruction with Parallel-OpT. (A) mean squared error (MSE), (B) peak signal-to-noise ratio (PSNR), and (C) structural similarity index (SSIM).
data are presented in Figure 5 (using MatLab Distribution Fitter app software). SSIM values reached a maximum of 0.9. Data presented a normal distribution form of $0.902 \pm 0.019$ (mean ± SD). This was confirmed by conducting a Shapiro–Wilk test calculated with SPSS software ($P < .05$). All data, except one, exceeded the threshold value of 0.85 SSIM, a standard limit for good image quality.

DISCUSSION AND CONCLUSIONS
This work shows the parallel implementation of the MLEM reconstruction algorithm using graphic card capabilities. Software solutions developed here were compared with reconstructions of clinical images using several image quality parameters (CNR, SSIM, MSE, etc.). Computer time used for image reconstruction in CUDA-based programs was shorter than that associated with serial solutions. Acceleration factors that varied between 2 and 170 times were obtained. In comparison with the TIGRE software, images were obtained of similar quality, but in some cases and for some specific image quality factors, images were 7 times better for our CUDA programs. The addition of 5% noise to signals did not increase the reconstruction time proportionally; in fact, it left it almost unaltered. Moreover, image quality decreased in general by a 5% factor (except for the PSNR parameter that lost 30% of its maximum value in some scenarios). The clinical study results showed that calculations were consistent and reproducible. Furthermore, the quality of image reconstruction was conserved even at short reconstruction times.

Table 2. Quantification of Data Presented in Figure 2

|                          | MSE       | PSNR      | SSIM     |
|--------------------------|-----------|-----------|----------|
|                          | Min. Iteration | Time (s) | Min. Iteration | Time (s) | Max. Iteration | Iteration | Time (s) |
| Shepp–Logan              | 32.2      | 1000      | 1548     | 33        | 1000        | 1548      | 0.992   | 77      | 119.7   |
| Shepp–Logan + Noise      | 58.95     | 1000      | 1595     | 30.42     | 1000        | 1595      | 0.980   | 35      | 56.55   |
| Abdominal CT             | 0.842     | 146       | 227.4    | 48.8      | 146         | 227.4     | 0.986   | 146     | 227.4   |
| Abdominal CT + Noise     | 7.46      | 22        | 35.98    | 39.4      | 22          | 35.98     | 0.967   | 16      | 26.39   |
| Head CT                  | 100       | 1000      | 1586     | 28        | 1000        | 1586      | 0.987   | 268     | 426.5   |
| Head CT + Noise          | 139.1     | 1000      | 1612     | 26.69     | 1000        | 1612      | 0.959   | 43      | 43.94   |
| Shepp–Logan with OS_SART| 262.2     | 30        | 26.8     | 23.94     | 30          | 26.8      | 0.900   | 30      | 26.8    |
| Shepp–Logan with ASD_POCS| 272.6    | 40        | 795.8    | 23.77     | 40          | 795.8     | 0.925   | 40      | 795.8   |

Figure 3. Calculation of computing times for different implementations developed in this study as well as TIGRE algorithms. An enlarged image of the first 200 iterations is presented in the lower right corner.
All these results imply that the MLEM implementations in CUDA using GPUs’ capabilities presented here were reliable and fast.

With respect to image acquisition, it was shown that to generate the system matrix, only projections in the range of 0° to 180° were needed. This contrasted with the acquisition of full-field projections (from 0° to 360°). This is a well-established fact in the field, so no data or images were presented to support this fact in this work. In general, acquisitions over small angles (1°, 2°, or 3°) at 1000 iterations took up to 5000 seconds. This amount of time was obviously greater than reconstructions in steps of 5° or larger. Measurements with larger angles were described in the online supplemental Figure 1. There it could be appreciated that the 5° measurements presented a perfect compromise between reconstruction quality and reconstruction time. Therefore, this value was the one selected for reconstruction in this study.

Figure 2 showed how the different image quality metrics (MSE, PSNR, SSIM) decreased first and then increased rapidly as the number of iterations increased. This allowed to deduce an “ideal” range of iterations for image reconstruction for this software implementation that was set between 50 and 100. Therefore, as it can be seen in Figure 2B, the PSNR parameter for the SLP and the abdominal image reached a local maximum value within the range of 50 to 100 iterations. For the head image, this effect was similar but less prominent. Obviously when larger number of iterations were performed, values of this last parameter were larger and therefore better. Nevertheless, the time used to improve 5% of this parameter could be easily 5–10 times larger. That made this approach of larger iterations less attractive for practical purposes. The maximum values of the SSIM metric for the SLP, the head, and the abdominal images were 0.992, 0.987, and 0.986, respectively. These maximum values were reached within the proposed range of iterations (see Figure 2C). After the maximum, the values remained stable and were not affected by a greater number of iterations. As indicated before in the text, values of SSIM over 0.85 represented images of good clinical quality.

Once the iteration interval and the degrees between projections used to acquire the images were established, the processing times of the different implementations developed were compared (Figure 3). It was found that the difference in time between *Serial-C* and *Parallel-OpT* to achieve the same result was ~1000 seconds at 50 iterations (12 times more) and 9000 seconds in 1000 iterations (170 times more). These results showed that implementations of the MLEM algorithm based on CUDA could perform good image reconstructions in shorter times than traditional CPU-based methods. Surprisingly, the addition, the Poisson noise did not affect the reconstruction times of the Parallel-OpT implementation. When compared with the results from TIGRE implementations, equal or better time performances were achieved, always with better image quality reconstruction.

If the reader considers that one of the basic ideas behind this work was to implement the MLEM algorithm in a novel way by using the GPUs’ capabilities in commercial/low capability computing systems, he/she can observe that this was accomplished. Therefore, the solution for image analysis presented here could be used in institutions or health systems with low financial
Figure 5. Histogram of SSIM values of reconstructed images from a clinical study. The data present an equally normalized distribution of values. (A) Parallel-Opt, (B) TIGRE 1, and (C) TIGRE 2.
resources in a reliable way, conserving image quality and maintaining reconstruction times as low as possible.

ACKNOWLEDGMENTS
We would like to thank CONACyT México for the support of the first author with a doctoral grant.

REFERENCES
1. Bushong SC. Manual de radiología para técnicos: Fisica, biología y protección radiológica. Elsevier, España, 2010.
2. Buzug TM. Computed Tomography: From Photon Statistics to Modern Cone-Beam CT. Springer-Verlag, Berlin, Heidelberg, 2008.
3. Feldkamp LA, Davis LC, Kress JW. Practical cone-beam algorithm. J Opt Soc Am A. 1984;1:612–619.
4. Kak AC, Slaney M, Wang G. Principles of Computerized Tomography Imaging. Med Phys. 2002;29.
5. Deans SR. The Radon Transform and Some of Its Applications. JohnWiley & Sons, Inc., New York, 1983.
6. Radon J. Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten. Akad Wiss. 1917;69:262–277.
7. Natterer F. The Mathematics of Computerized Tomography. Society for Industrial and Applied Mathematics, USA, 2001.
8. Hullon BF, Nyuys J, Zaidi H. Iterative reconstruction methods. In: Quantitative Analysis in Nuclear Medicine Imaging. Springer US, Boston, MA, 2006:107–140.
9. Zeng G. Medical Image Reconstruction: A conceptual tutorial. Springer, Higher Education Press, Beijing, 2010.
10. Hsieh J, Nett B, Yu Z, Sauer K, Thibault J-B, Bouman. C A. Recent advances in CT image reconstruction. Curr Radiol Rep. 2013;1:39–51.
11. Vandenbergher S, Asseler YD, Van De Walle R, Kauppinen T, Koole M, Bouvens L. Iterative reconstruction algorithms in nuclear medicine. 2001;25:105–111.
12. Geyer LL, Schoepf UJ, Meinel FG, Nance JW, Bastarrrika G, Leipsic JA, Paul NS, Rengo M, Laghi A, De Cecco CN. State of the art: iterative CT reconstruction techniques. Radiology. 2015;276:339–357.
13. Shepp LA, Vardi Y. Maximum likelihood reconstruction for emission tomography. IEEE Trans Med Imaging. 1982;1:113–122.
14. Cook S. CUDA Programming: A Developer’s Guide to Parallel Computing with GPUs. 2013.
15. Sanders y J, Kandrot E. CUDA by Example: An Introduction to General-Purpose GPU Programming, 1st ed. Addison-Wesley Professional, USA, 2010.
16. Schellmann M, Gorlatch S, Meiländer D, Kösters T, Schäfers K, Wübbeling F, Burger M. Parallel medical image reconstruction: from graphics processing units (GPU) to CPU. J Supercomput. 2011;57:151–160.
17. NVIDIA. GPU-Accelerated applications. 2019.
18. Eklund A, Dufort P, Forsberg D, LaConte SM. Medical image processing on the GPU – past, present and future. Med Image Anal. 17(8):1073–1094, 2013. https://doi.org/10.1016/j.media.2013.05.008 [Internet]. Available online 5 June 2013.
19. Kalaiselvi S, SriRamakrishnan P, Somasundaram K. Survey of using GPU CUDA programming model in medical image analysis. Inform Med Unlocked. 2017;9:133–144.
20. Tatarchuk N, Shoff J, DeCoro C. Advanced interactive medical visualization on the GPU. J Parallel Distrib Comput. 2008;68:1319–1328.
21. Flores LA, Vidal V, Mayo P, Rodenas F, Verdu G. CT image reconstruction based on GPUs. Procedia Comput Sci. 2013;18:1412–1420.
22. Flores LA, Vidal V, Verdu G. GPU based algorithms in CT imaging. Annals of Multicare and GPU Programming, Vol 2, España, 2015.
23. Belzunce M, Verrazza C, Cohen I, Venäläpiä E. CUDA parallel implementation of image reconstruction algorithm for positron emission tomography. Open Medical Imaging Journal. 6. 10.2174/18743471012060601018, 2012.
24. Xie L, Hu Y, Yan B, Wang L, Yang B, Liu W, Zhang L, Luo L, Shu H, Chen Y. An effective CUDA parallelization of projection in iterative tomography reconstruction. PLoS One. 2015;10:1–17.
25. Prate G, Xing L. GPUs computing in medical physics: a review. Med Phys. 2011;38:2685–2697.
26. Lange K, Bahn M, Little R. A theoretical study of some maximum likelihood algorithms for emission and transmission tomography. IEEE Trans Med Imaging. 1987;6:106–114.
27. Gopi ES. Digital Signal Processing for Medical Imaging Using Matlab. New York, NY, USA: Springer, 2013.
28. Van Helsetryck T, Wuyts S, Goossens M, Batenburg Kees J, Sijbers J. The implementation of iterative reconstruction algorithms in MATLAB, Masters Thesis, Department of Industrial Sciences and Technology, University College of Antwerp, Belgium, July 2007.
29. Han L. Tools for 2D Tomographic Reconstruction, (2017), GitHub repository, https://github.com/phynhan/phynhan/tree/tomo2d [Internet]. Available from: https://github.com/phynhan/phynhan/tree/tomo2d [Internet]; Accessed: April 20, 2019.
30. NVIDIA. CUDA CUBLAS Library, June 2017. https://docs.nvidia.com/cuda/ archive/8.0/pdf/CUBLAS_Library.pdf [Internet]; Accessed: May 2, 2019.
31. NVIDIA. CUDA CUBLAS-NVIDIA Developer. 2018. https://developer.nvidia.com/cublas [Internet]; Accessed: May 2, 2019.
32. Shepp LA, Logan B F. The Fourier reconstruction of a head section. IEEE Trans Nucl Sci. 1974;21:21–43.
33. Clark K, Vardi B, Smith K, Freyman J, Kirby J, Koppel P, Moore S, Phillips S, Maffitt D, Fringle M, Tarbox L. Prior F. The Cancer Imaging Archive (TCAI): maintaining and operating a public information repository. J Digit Imaging. 2013;26:1045–1057.
34. Dougherty G. Digital Image Processing for Medical Applications. Cambridge University Press, United Kingdom, 2009.
35. Wang Z, Bovik A C. Mean squared error: lot it or leave it? A new look at signal fidelity measures. IEEE Signal Process Mag. 2009;26:98–117.
36. Salomon D. Data Compression: The Complete Reference. Berlin, Heidelberg: Springer-Verlag, New York, 2006.
37. Wang Z, Bovik AC. Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process. 2004;13:600–612.
38. Bigni A, Dosanjh M, Hancock S, Soleimani M. TIGRE: a MATLAB-GPU toolbox for CBCT image reconstruction. Biomed Phys Eng Express. 2016;2.
39. Sidky EY, Pan X. Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization. Phys Med Biol. 2008;53:4777–4807.
40. Helmke M. The Official Ubuntu book. Prentice Hall, Upper Saddle River, NJ, USA, 2014.
41. GCC, the GNU Compiler Collection - GNU Project - Free Software Foundation (FSF) version 4.9.2. Released 2014. https://gcc.gnu.org/ [Internet]; Accessed: May 4, 2019.
42. OpenCV-OpenCV modules version 3.2 Released 2016. https://docs.opencv.org/3.2.0/ [Internet]; Accessed: June 5, 2020.
43. Kaehler A, Bradski G. Learning OpenCV 3: computer vision in C++ with the OpenCV library. O’Reilly Media, Inc.; USA, 2016.

Supplemental Materials
Supplemental Figure 1: https://doi.org/10.18383/j.tom.2020.00011.sup.01

Disclosures: The authors have nothing to disclose.

Conflict of Interest: None reported.