J. Nucl. Phys. Mat. Sci. Rad. A. Vol. 7, No. 2, Feb. 2020, pp.165–171

CHITKARA

Journal of Nuclear Physics, Material Sciences, Radiation and Applications
Journal homepage: https://jnp.chitkara.edu.in/

Study of CT Images Processing with the Implementation of MLEM Algorithm using CUDA on NVIDIA'S GPU Framework

T. A. Valencia-Pérez*, J. M. Hernández-López, E. Moreno-Barbosa and B. de Celis-Alonso

Faculty of Physical Sciences Mathematics Benemérita Universidad Autónoma de Puebla, Avenida San Claudio y 18 Sur, Colonia San Manuel, Building FM2-203, Ciudad Universitaria, C.P. 72570, Puebla, Mexico

*Email: antonio_2827@hotmail.com

ARTICLE INFORMATION

Received: October 10, 2019
Accepted : February 3, 2020
Published online: February 28, 2020

Keywords:
Computed tomography, Algorithms, GPU, Reconstruction, Image quality

DOI: 10.15415/jnp.2020.72021

ABSTRACT

In medicine, the acquisition process in Computed Tomography Images (CT) is obtained by a reconstruction algorithm. The classical method for image reconstruction is the Filtered Back Projection (FBP). This method is fast and simple but does not use any statistical information about the measurements. The appearance of artifacts and its low spatial resolution in reconstructed images must be considered. Furthermore, the FBP requires of optimal conditions of the projections and complete sets of data. In this paper a methodology to accelerate acquisition process for CT based on the Maximum Likelihood Estimation Method (MLEM) algorithm is presented. This statistical iterative reconstruction algorithm uses a GPU Programming Paradigms and was compared with sequential algorithms in which the reconstruction time was reduced by up to 3 orders of magnitude while preserving image quality. Furthermore, they showed a good performance when compared with reconstruction methods provided by commercial software. The system, which would consist exclusively of a commercial laptop and GPU could be used as a fast, portable, simple and cheap image reconstruction platform in the future.

1 Introduction

X-ray (XR) Computed Tomography (CT) is a nondestructive technique in which an XR source rotates around an object of interest generating axial slices of its internal structure. CT is nowadays an indispensable tool in medicine for the diagnosis of multiple diseases [1]. Since its introduction in 1970 there are around 30,000 CT-scanners in the world and the number continues to rise exponentially. Currently, the country with the highest number of CT-scanners is Japan with 107,2 per million inhabitants [2] while in the United States of America 253.80 scans per 1000 inhabitants are completed per year [2]. Even if CT techniques represent the major development to the field of X rays in the last 50 years, there is still room for improvement. The major lines of work to achieve such objectives would be: 1. Reduction of patient exposure, 2. Reduction of acquisition and processing times 3. Development of new techniques with new functionality and 4. Cost reduction [3]. Some of the solutions to these points have been: higher performance hardware which will allow for lower costs as well as lower dose delivery to patients all without compromising diagnostic effectiveness. The use of statistical iterative methods for image reconstruction reducing processing times. The development of Portable CT-scanners, with capacities lower than standard CT-scanners [4] we examined the use of a portable head CT scanner (CereTom but cheaper and useful if the patient cannot be diagnosed in a radiology department. In clinical and emergency room environments, the speed of acquisition and information processing are crucial and trump in relevance the other developments presented here.

The method used by actual CT-scanners for image reconstruction is the Filtered Back Projection (FBP) one [5], [6], which is mathematically based on the Radon Transform (RT) [7], [8]. In it, the image of a given object is reconstructed from a set of XR back-projections of the aforementioned object [9]. There are many computational algorithms that can be used to solve the RT. Traditionally they are classified into either: Analytical Reconstruction methods or Iterative Reconstruction methods. For different reviews on reconstruction methods readers can consult [3], [10], [11]. For the first group, the FBP algorithm has been the golden standard to date [12]. Nevertheless, recent works on statistical based methods for tomographic image reconstruction, specifically Maximum Likelihood Expectation Maximization (MLEM) [13], have seen a large increase in use lately. In the FBP approach, the reconstruction of an image is achieved using the RT in conjunction with an algorithm based on the central slice theorem [14]. In general lines, it is a simple and fast

ISSN No.: 2321-8649(Print) ISSN No.: 2321-9289(Online); Registration No.: CHAENG/2013/51628

reconstruction algorithm with low computational cost. In contrast, the appearance of artifacts and its low spatial resolution in reconstructed images must be considered. Furthermore, the FBP requires of optimal conditions of the projections and complete sets of data. That is, it requires an (almost) infinite number of projections without noise so that the solution of the reconstruction is perfect. This is far from real life situations in which only a finite set of projections will be available. Therefore, "approximate" reconstruction methods must be developed, and this is the main advantage of iterative algorithms. In general, they are less sensitive to incomplete sets of data and the artifacts that arise in the reconstruction process can be reduced or even eliminated [12], [15]. This translates into a better quality of reconstructed images. For a review on reconstruction methods see [16]. The main limitation that these techniques present with respect to analytical reconstruction methods, is their high computational costs. One way to deal with this limitation is parallelizing (dividing in parts) image reconstruction. To this end and taking advantage of the fact that image reconstruction problems have a high degree of data parallelism and a large number of independent arithmetic calculations; Graphic Processing Unit (GPU) with the parallel programming model of Compute Unified Device Architecture (CUDA) [17]–[19] can be used to speed up image computational times.

In 2007, the NVIDIA corporation launched CUDA, which is hardware architecture, used to work with its own GPU´s. This architecture was formed by execution units, called Streaming Multiprocessors (SM), which at the same time were formed by computing cores called Streaming Processors (SP), or CUDA cores. It is them which executed instructions (i.e. mathematical operations or data addressing and transfer in memory). The hardware structure of these graphical cards used at the same time the central processing unit (CPU) and the GPU. The sequential part of applications was executed in the CPU (Host) and the computationally intensive part was accelerated by use of the GPU (Device) in parallel. This technology allowed researchers working in highly specific programming languages for graphics such as OpenGL [20] to work in more standard high-level programming environment such as C/C++ [21]. It also facilitated the writing of applications to the GPU and incorporated specialized libraries among which are signal management, image and sound, linear algebra (cuBLAS), etc.

Since its introduction, CUDA has been widely implemented and used in research in fields as different as: Bioinformatics, Computational Chemistry, Imaging and Computer Vision, Weather and Climate, Numerical Analytics, etc. [22]. Che et al. [23] presented a review on the general-purpose applications of graphics processors using CUDA. As it could be expected, medical image processing was one of the first fields in which NVIDIA GPU´s and CUDA were used [24], [25] mainly because they can dramatically accelerate parallel computing, are affordable and energy efficient. In the field of medical imaging, GPUs are in some cases crucial for enabling practical use of computationally demanding algorithms. This review presents the past and present work on GPU accelerated medical image processing, and is meant to serve as an overview and introduction to existing GPU implementations. The review covers GPU acceleration of basic image processing operations (filtering, interpolation, histogram estimation and distance transforms. Tatarchuk et al. [26] pre-operative planning, and surgical training. The task of visualization is no longer limited to producing images at interactive rates, but also includes the guided extraction of significant features to assist the user in the data exploration process. An effective visualization module has to perform a problem-specific abstraction of the dataset, leading to a more compact and hence more efficient visual representation. Moreover, many medical applications, such as surgical training simulators and pre-operative planning for plastic and reconstructive surgery, require the visualization of datasets that are dynamically modified or even generated by a physics-based simulation engine. In this paper we present a set of approaches that allow interactive exploration of medical datasets in real time. Our method combines direct volume rendering via ray-casting with a novel approach for isosurface extraction and re-use directly on graphics processing units (GPU presented a set of approaches that allowed interactive exploration of medical datasets in real time. The development of a fast GPU-based algorithm to reconstruct high quality images from under sampled and noisy projection data was also presented by Flores et al. [27]. Belzunce and colleagues presented a parallel GPU implementation of the iterative reconstruction algorithm MLEM 3D using CUDA. This was achieved for nuclear medicine data (SPECT and PET) and an acceleration factor of up to 85 times was achieved with respect to a single thread CPU implementation [28]. More applications on the use of GPU's for analysis of medical images reconstruction can be found in a review from Xing et al. [29].

As mentioned before, the main disadvantage of MLEM algorithms is their high computational and time costs, a reason why FBP algorithms are still the golden standard in the field. Nevertheless, these algorithms and their applications to image reconstruction have the advantage that they can be parallelized. This makes GPU characteristics and technologies an interesting tool to develop new image analysis solutions. In this paper a

methodology to accelerate tomographic image analysis based on iterative algorithms (MLEM), combined with the use of NVIDIA GPU programmed in the CUDA architecture is presented.

2 Methods

The line of work presented in this study consisted of: First, the MLEM algorithm implementation was solved in parallel using GPU with CUDA. Second, different image quality parameters as a function of computational time and number of iterations were calculated for a clinical study and aphantom collection using our MLEM algorithm implementation and open software solutions of the MLEM algorithm.

2.1 The MLEM Algorithm

In a paper by Lange et al. [30], the expectation maximization reconstruction algorithm for emission and transmission tomography was derived. In contrast, here the MLEM algorithm was used for the development of an CT equivalent image reconstruction system. The different analytical steps followed in this algorithm are presented in Fig. 1.

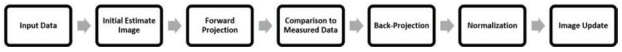

Figure 1: The MLEM Algorithm. The different processes which are completed to solve the MLEM algorithm are presented in this image.

Table 1: Quantification of the data presented in Fig. 4.

	MSE			PSNR			SSIM		
	Min	**Max**	**Mean**	**Min**	**Max**	**Mean**	**Min**	**Max**	**Mean**
Clinical study – Parallel-OpT	0	105.4	60.13	0	32.3	29.63	0.93	1	0.95
Phantom collection – Parallel-OpT	35.79	92.32	60.52	28.48	32.59	30.38	0.89	0.94	0.91
Phantom collection – TIGRE	272.66	360.92	348.88	22.56	23.77	22.72	0.87	0.93	0.92

The different processes of the MLEM algorithm could be expressed mathematically by the following formula:

$$x_j^{r+1} = \frac{x_j^r}{\sum_{i=1}^{M} a_{ij}} \sum_{i=1}^{M} \frac{a_{ij} y_i}{\sum_{l=1}^{N} a_{il} x_l^r} \qquad (1)$$

Here was the pixel j of the x image at the iteration, was the bin i of the measured projection given by the CT scanner, was the system matrix whose coefficients connected the image values with the projections and described the probability of detecting a photon in pixel j in projection bin i.

The creation of system matrix depended on the number of projection lines, the number of projection angles and the size of the image which could be calculated in different ways. In this paper we have used [31]–[33] to simulate and generate the system matrix computationally with MatLab.

Data parallelization of the MLEM algorithm was the most important property that facilitated this process. Operations performed on data structures were handled by four kernels (Back-Projection, Forward-Projection, Normalization, Image Update), which distributed the data to threads blocks. The algorithm was developed here: Parallel-OpT presents the two main CUDA routines employed in the parallel solution of the MLEM algorithm in pseudocode (see Fig. 2).

The open source codes used to solve the MLEM algorithm were the reconstruction methods from the open

Pseudocódigo: Rutina Forward-projection.

Require: Vector *Image* and Matrix *SystemMatrix*
Step 1. Memory allocate for the device.
Step 2. Create the CUBLAS context.
Step 3. Transfer data from the host to the device.
Step 4. CUBLAS Level-2. Matrix-vector operations $y = alpha \cdot A \cdot x + beta \cdot y$
status=cublasSgemv(handle,CUBLAS_OP_N,
SystemMatrix.rows,SystemMatrix.cols,
&alpha,d_SystemMatrix.data,SystemMatrix.rows,
d_Image,1,&beta,d_ForwardProjection,1);
Step 5. Transfer data from the device to the host.
Step 6. Return ForwardProjection.
Step 7. Free device memory
Step 8. Destroy CUBLAS context
return Vector *ForwardProjection*

Pseudocódigo: Rutina Back-projection.

Require: Vector *ForwardProjection*, Vector *Projections* and Matrix *SystemMatrix*
Step 1. Memory allocate for the device.
Step 2. Transfer data from the host to the device.
Step 3. Launch the Cuda Kernel of the Comparative function.
Number of threadsPerBlock.
Number of blocksPerGrid.
ComparativeKernel<<<blocksPerGrid,
threadsPerBlock>>>(d_Projections, d_ForwardProjection,
d_Comparative)
Step 4. Create the CUBLAS context.
Step 5. CUBLAS Level-2. Matrix-vector operations $y = alpha \cdot A \cdot x + beta \cdot y$
status=cublasSgemv(handle,CUBLAS_OP_T,
SystemMatrix.rows,SystemMatrix.cols,
&alpha,d_SystemMatrix.data,SystemMatrix.rows,
d_Comparative,1,&beta,d_BackProjection,1);
Launch the Cuda Kernel of the Factor-Correction function.
Number of threadsPerBlock.
Number of blocksPerGrid.
Factor-CorrectionKernel<<<blocksPerGrid,
threadsPerBlock>>>(d_ColSum, d_BackProjection,
d_FactorCorrection)
Step 6. Transfer data from the device to the host.
Step 7. Free device memory
Step 8. Destroy CUBLAS context
return Vector *FactorCorrection*

Figure 2: Routines Forward- and Back- projection.

source Matlab library TIGRE [34]. The tomographic iterative GPU-based reconstruction toolbox was used here for comparative purposes: The ASD-POCS library which is based on the adaptive steepest descent projection onto convex subsets algorithm (see [35] for a detailed explanation on both algorithms).

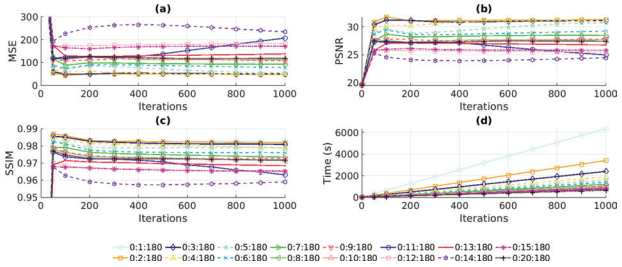

Figure 3: MSE, PSNR and SSIM and time as a function of the number of iterations and angular degree in each view, used to cover the 180 degrees of an image. Parameters obtained when reconstructing the Shepp-Logan phantom with the Parallel-OpT algorithm. In each graph different angles were used to acquire the different views that covered the whole 180 degrees used to image an object.

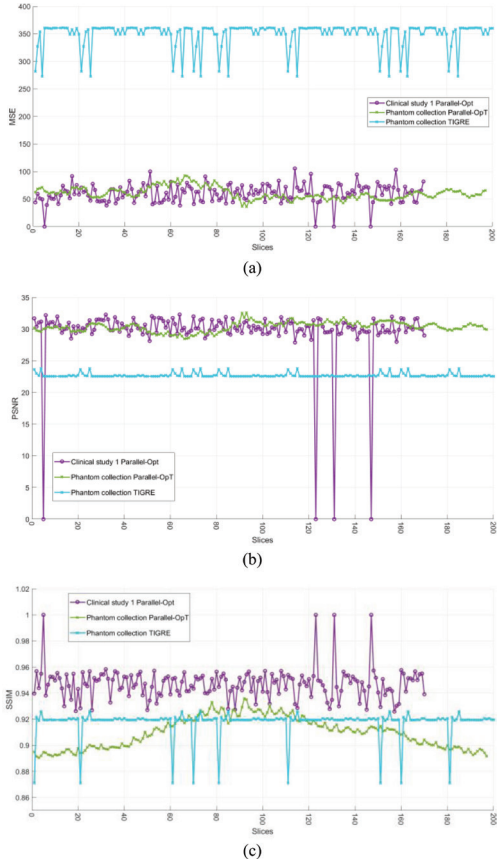

Figure 4: MSE, PSNR and SSIM image quality parameters vs. Slices. (a) MSE (b) PSNR (c) SSIM image parameters obtained after reconstruction with the Parallel-OpT and TIGRE method: ASD_POCS.

2.2 Hardware and Software

The parallel algorithm was developed and executed on a laptop with an Intel Core i7-5500U processor running at 2.40 GHz and equipped with an NVIDIA GeForce 840M GPU graphics card with a global memory of 2048 MB and 384 CUDA cores on an Ubuntu operating system. MatLab R2017a and C 4.9.2 were employed for the serial part. CUDA C 7.5 was employed for the parallel part and Open Source Computer Vision library (OpenCV 3.2.0) [36] was used to calculate the quality measurements.

3. Results and Conclusions

A clinical study with 171 slices and aphantom collection with 199 slices were used in this paper [37] the National Cancer Institute (NCI). The CT images were pixels with a dynamic range of 8 bits per pixel.

An additional study was performed to verify the optimal number of views for image reconstruction with the algorithms calculated in this paper. In this analysis, image quality parameters as well as computational time were calculated as a function of number of iterations and as a function of number of views. As a compromise conclusion, it was found that using 5-degree jumps per view to cover whole field of view as well as 50 to 100 iterations, produced reliable results without extending computing time and obtaining high quality image parameter values. It is also worth mentioning that reconstructions calculated with the different algorithms produced very similar results, when considering image quality parameters, when covering the field of study with projections varying between 0-180 degrees and 0-360 degrees (data not shown here). Because of this, the reconstructions in this work, only used a system matrix varying from 0 to 180 degrees.

The Fig. 4 was produced to assess that algorithms implement in this work, produced images similar in quality to those obtained from commercial systems from which they were derived and shows reconstructions using Parallel-GPU and TIGRE algorithm for a maximum of 1000 iterations. The results were summarized in the following table:

We observe that we obtained a similar mean SSIM value with our implementation of the MLEM algorithm and the TIGRE algorithm, and with very similar maximum and minimum values. We also found that we obtained better mean of the metrics with Parallel-Opt. As we can see from Figure 4 y Table 1, our implementation of the MLEM algorithm in the reconstruction of each slice had a good behavior, having obtained close values in its minimum and

maximum of the metrics evaluated, this shows the reliability of Parallel-OpT.

The Fig. 5 presents computing time for Parallel-OpT implementation and an algorithm developed by TIGRE.

It can be observed that differences in time between the Paralell-OpT and TIGRE algorithm were of the order of 3 times faster for the parallel solution.

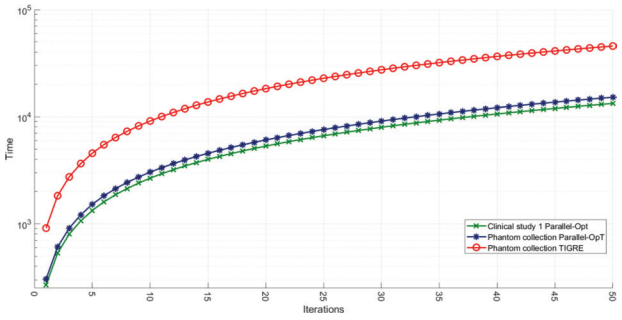

Figure 5: Computing times for Parallel-OpT and TIGRE algorithm.

One limitation for the implementation developed here was the size of the system matrix of images used. Their size (512 × 512) made memory of the GPU insufficient to process images directly. Therefore, during analysis, images had to be divided before reconstruction. Blocks were built with a ratio image size/block of 16. This way there was no computing bottleneck when sending serially each block from the CPU to the GPU. This data transfer was limited by GPU´s bus bandwidth and its latency.

The work environment of our CUDA C system provides developers with a full range of tools and solutions pertaining to the CUDA ecosystem with a relatively small learning curve as it can be seen as the extension of well-known programming tools belonging to the C language. The advantages of using this approach for the reconstruction of tomographic images is that its implementation is very economical and easy to access for many people as independent researchers, small universities or research institutes that do not have sufficient funding for this purpose.

References

[1] S. C. Bushong, Manual de radiología para técnicos: Física, biología y protección radiológica. Elsevier, 2010.

[2] Health at a Glance 2017. OECD Publishing, 2017. https://doi.org/10.1787/19991312

[3] T. M. Buzug, Computed Tomography: From Photon Statistics to Modern Cone-Beam CT. Springer, Berlin, Heidelberg, 2008.

[4] K. Peace et al., J. Neurosci. Nurs. **42**, 109, 2010. https://doi.org/10.1097/JNN.0b013e3181ce5c5b

[5] L. A. Feldkamp, L. C. Davis and J. W. Kress, J. Opt. Soc. Am. **1**, 612 (1984). https://doi.org/10.1364/JOSAA.1.000612

[6] A. C. Kak, M. Slaney, G. Wang, Med. Phys. **29,** 107 (2002). https://doi.org/10.1118/1.1455742

[7] S. R. Deans, The Radon Transform and Some of Its Applications. John Wiley & Sons, Inc., 1983.

[8] J. Radon, Akad. Wiss. **69**, 262 (1917).

[9] F. Natterer, The Mathematics of Computerized Tomography. Society for Industrial and Applied Mathematics, 2001. https://doi.org/10.1137/1.9780898719284

[10] B. F. Hutton, J. Nuyts, Y. H. Zaidi, "Iterative Reconstruction Methods", en Quantitative Analysis in Nuclear Medicine Imaging, H. Zaidi, Ed. Boston, MA: Springer US, 2006, pp. 107–14\0. https://doi.org/10.1007/0-387-25444-7_4

[11] G. L. Zeng, Medical Image Reconstruction. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[12] J. Hsieh, B. Nett, Z. Yu, K. Sauer, J.-B. Thibault, C. A. Bouman, Curr. Radiol. Rep. **1**, 39 (2013). https://doi.org/10.1007/s40134-012-0003-7

[13] L.A. Shepp Y. Vardi, IEEE Trans. Med. Imaging **1**, 113 (1982). https://doi.org/10.1109/TMI.1982.4307558

[14] H. Shi, S. Luo, Z. Yang, G. Wu, PLoS One **10**, 1 (2015). https://doi.org/10.1371/journal.pone.0138498

[15] S. Vandenberghe et al., Computerized Medical Imaging and Graphics **25**, 105 (2001). https://doi.org/10.1016/S0895-6111(00)00060-4

[16] L. L. Geyer et al., Radiology **276**, 339 (2015). https://doi.org/10.1148/radiol.2015132766

[17] S. Cook, CUDA Programming: A Developer's Guide to Parallel Computing with GPUs, núm. 1. 2013.

[18] J. Sanders, E. Kandrot, CUDA by Example: An Introduction to General-Purpose GPU Programming, 1st ed. Addison-Wesley Professional, 2010.

[19] M. Schellmann et al., J. Supercomput. **57**, 151 (2011). https://doi.org/10.1007/s11227-010-0397-z

[20] R. Whitrow, OpenGL Graphics Through Applications, 1a ed. Springer Publishing Company, Incorporated, 2008.

[21] GNU Project, "GCC, the GNU Compiler Collection". 1987.

[22] NVIDIA, "GPU-Accelerated applications". 2019.

[23] S. Che et al., J. Parallel Distrib. Comput. **68**, 1370 (2008). https://doi.org/10.1016/j.jpdc.2008.05.014

[24] A. Eklund, P. Dufort, D. Forsberg and S. M. LaConte, Med. Image Anal. **17**, 1073 (2013). https://doi.org/10.1016/j.media.2013.05.008

[25] T. Kalaiselvi, P. Sriramakrishnan and K. Somasundaram, Informatics Med. Unlocked **9**, 133 (2017). https://doi.org/10.1016/j.imu.2017.08.001

[26] N. Tatarchuk, J. Shopf and C. DeCoro, J. Parallel Distrib. Comput. **68**, 1319 (2008). https://doi.org/10.1016/j.jpdc.2008.06.011

[27] L. A. Flores, V. Vidal, P. Mayo, F. Rodenas and G. Verdú, Procedia Comput. Sci. **18**, 1412 (2013). https://doi.org/10.1016/j.procs.2013.05.308

[28] M.A. Belzunce, C.A. Verrastro, E. Venialgo and I. M. Cohen, Open Med. Imaging J. **108** (2012). https://doi.org/10.2174/1874347101206010108

[29] G. Pratx and L. Xing, Med. Phys. **38**, 2685 (2011). https://doi.org/10.1118/1.3578605

[30] K. Lange, M. Bahn and R. Little, IEEE Trans. Med. Imaging. **6**, 106 (1987). https://doi.org/10.1109/TMI.1987.4307810

[31] E. S. Gopi, Digital Signal Processing for Medical Imaging Using Matlab. New York, NY, USA: Springer, 2013. https://doi.org/10.1007/978-1-4614-3140-4

[32] V. Hemelryck Tessa, W. Sarah, G. Maggie, B. Kees Joost, y J. Sijbers, "ITERATIVE RECONSTRUCTION ALGORITHMS The implementation of iterative reconstruction algorithms in MATLAB", 2007.

[33] L. Han, "Tools for 2-D Tomographic Reconstruction", GitHub repository. GitHub, 2017.

[34] A. Biguri, M. Dosanjh, S. Hancock and M. Soleimani, Biomed. Phys. Eng. Express **2**, 55010 (2016). https://doi.org/10.1088/2057-1976/2/5/055010

[35] E. Y. Sidky and X. Pan, Phys. Med. Biol. **53**, 4777 (2008). https://doi.org/10.1088/0031-9155/53/17/021

[36] I. Intel Corporation Willow Garage, "Open Source Computer Vision". 2000.

[37] K. Clark et al., J. Digit. Imaging. **26**, 1045 (2013). https://doi.org/10.1007/s10278-013-9622-7

CHITKARA

Journal of Nuclear Physics, Material Sciences, Radiation and Applications

Chitkara University, Saraswati Kendra, SCO 160-161, Sector 9-C, Chandigarh, 160009, India

Volume 7, Issue 2 **February 2020** **ISSN 2321-8649**