Real-time Simulation of Radiological Images Using CUDA Technology

Elena Gianaria
Dipartimento di Informatica
Università degli Studi di Torino
Corso Svizzera 185, 10149 Torino, ITALY
Email: elena.gianaria@unito.it

Elena Gallio
S.C. Fisica Sanitaria
A.O.U. Città della Salute e della Scienza di Torino
Corso Bramante 88/90, 10126 Torino, ITALY
Email: egallio@cittadellasalute.to.it

***Abstract*—Radiography is nowadays a common medical exam, used for diagnosing several diseases, but has the disadvantage of exposing people to a dose of radiation. For this reason, it is important to study methods for reducing such dose. In this paper we present a digital X-ray simulation tool that simulates a radiological exam on a virtual patient. The software builds a physically-realistic radiography in real-time thanks to GPGPU programming and CUDA technology. It aims to be used in radiological departments, for testing new dose reduction procedures and training health operators. We validated the software comparing the results with real radiographic images, and we tested it on different graphic cards obtaining running times that are 35 to 250 times faster than the corresponding CPU implementation.**

Keywords*—*Simulating X-ray images; Medical Imaging; Graphics Processor Unit; CUDA;

I. Introduction

Originally designed for accelerating computer graphics applications, the graphics processing unit (GPU) has emerged as versatile platform for massively parallel computing, without loss of reliability and accuracy. The use of the GPU has clear advantages for complex computational problems in different scientific disciplines such as linear algebra, differential equation, ray tracing, data-mining, biophysics, fluid dynamics as well as medical physics. A key topic in this last field is the simulation of radiography images. A software application, for simulating accurate and realistic images, should implement some fundamental components: a primary beam and its attenuation through human body and detector materials; the scatter radiation; the image noise; a virtual patient and a digital detector. The first three components are usually simulated through Monte Carlo method that provides good results, but it requires long computation times, making it unusable for a real-time software. Since the generation of realistic X-ray images requires complex operations independently performed on each pixel, the GPU is suitable for this kind of massively parallel problems. Furthermore the use of GPU permits a real-time interaction, so the results of simulation are available in few seconds. Such tool is useful in a radiological department for training health operators: the operator can analyze in real-time the influence of exposure parameters (photon beam energy, intensity of radiation beam, detectors) on a virtual patient and dose optimization processes can be performed, reducing the radiation dose to patients. Moreover, such tool allows to make the best use of available resources, saving time, human workload and occupancy time of the radiological room.

In this paper, we present a real-time tool for simulating digital X-ray images, based on GPU computing and CUDA architecture. We reproduce a radiological room including a source of primary beam, a virtual phantom and a detector. We also take into account the attenuation of primary beam and the typical noise of the radiological image. The software is validated by the comparison between real and simulated radiography. We test our tool on GPUs having different level of performance for evaluating its response in real-time. Finally we make a comparison of the computational times for generating an image on the GPUs and on CPU.

Other applications that exploit GPU computing for the simulation of medical images have been proposed. Shi *et al.* [1] have provided a comprehensive reference in the GPU-based medical image processing. Pratx and Xing [2] besides report the basic principles of GPU computing and the main performance optimization techniques, have summarized existing applications in different areas of medical physics, such as image reconstruction, dose calculation and treatment plan optimization, image processing. Ino *et al.* [3], Ruijters *et al.* [4] and Dorgham *et al.* [5] have considerably improved computation time of 2D to 3D registration algorithms, parallelizing on GPU the generation of digitally reconstructed radiographs (DRRs). Vidal *et al.* [6] have implemented a tool for the simulation of X-ray attenuation through complex objects. Chou *et al.* [7] have accelerated and optimized the forward-projection algorithm for medical image reconstruction. Viswanathan *et al.* [8] have sped up the simulation of X-ray radiography and computed tomography images, based on graphics rasterization and image reconstruction techniques.

Each one of the previous tools performs only a part of the process of radiography simulation: on the other side, our application aim to implement them all in one framework. In the following we present our approach. In Section II the theoretical model is presented in details, while in Section III the simulation tool is explained. Our results and conclusion are outlined respectively in Section IV and Section V.

II. Theoretical model

The simulation tool displays a virtual radiological room, in which there are an X-ray source, a virtual patient and a

This work was partly funded by the project "Sviluppo di un modello virtuale polifunzionale basato su database radiologici", Ricerca Scientifica Applicata.

detector. The virtual body is traversed by several X-rays beams, one for each detector pixel, produced by the source. A certain amount of X-ray is absorbed by the virtual body, depending on the tissues density. The remaining radiation beam hits the detector behind the virtual body, then is converted in the dose and, by adding the noise components, in a digital value (gray level) for each pixel using the detector response function.

Primary beam and its attenuation. The interaction of the primary beam with the virtual phantom is modeled using ray-tracing techniques and X-ray attenuation laws, also called *Beer–Lambert* law:

$$I = I_0 e^{-\mu_x}$$

The source, punctual and polychromatic, is an ordinary X-ray tube with energy and intensity user-adjustable. The different beams are obtained from IPEM [9] by setting traditional clinical X-ray tube features and dividing the spectrum in $5keV$ (Kiloelettronvolt) energy bins. The user can choose among a set of different input files, one for every beam, that indicate the fluence of photon normalized at a distance of $1m$ and the energy for each energy bin.

Virtual phantom. The virtual body consists in a 3D voxel matrix generated from CT images. Let μ_{voxel} be the linear attenuation coefficient of each voxel. It is estimated from the HU (Hounsfield scale) value:

$$HU = 1000 \cdot ((\mu_{voxel} - \mu_{water})/\mu_{water})$$

where for μ_{water} we consider the effective energy of the TC radiation beam of anthropomorphic phantom, corresponding to $60keV$. For the other energies μ_{voxel} is multiplied by a factor f equal to the linear attenuation coefficient of the crossed material divided by the one of water, at the same energy. The f values are obtained from ICRU 44 [10] taking into account the tissues density: the trend of the f coefficients is quite similar in soft tissues such as air, water and muscle, but is different for high density tissues like bones. For this reason we set up a discriminant value μ_{voxel} equal to $0.25cm^{-1}$, that is the linear attenuation coefficient of compact bone: if μ_{voxel} is lesser than $0.25cm^{-1}$ the anatomical section crossed is a soft tissue and/or water; otherwise it is a high density region.

Detector. The detector is designed like a set of coplanar points belonging to a planar surface having the same dimensions of the image to generate. Each point simulates the center of the corresponding pixel of the real detector. A radiation beam is simulated from the source to each pixel. The distance between the source and the coplanar point (beam path) is divided into n equal step increments. The number of steps can be set by the user and for each of them the attenuated photon fluence (Φ_{att_i}) is computed for each energy bin as:

$$\Phi_{att_i} = \Phi_i \prod_{j=1}^{n} e^{-\mu_j \Delta x}$$

where μ_j is the linear attenuation coefficients and Δx is the step length (distance between two step increments). The dose contribution of each energy bin is calculated through the formula:

$$D_i = \left(\frac{\mu_{en}}{\rho}\right)_a \cdot \psi_i \cdot \left(\frac{100}{d_{ij}}\right)^2 \cdot f_c$$

where $(\mu_{en}/\rho)_a$ is the mass absorption coefficient in air, derived from ICRU 44 [10], ψ_i is the attenuated energy fluence of the single energy bin $(\Phi_{att_i} \cdot E_i)$, $(100/d_{ij})^2$ is the distance correction factor and f_c is the conversion factor from eV to J. The total dose D incident in the pixel center is equal to:

$$D = \sum_{i=1}^{N} D_i$$

where N is the number of energy bin of the selected beam. The total dose carried from the source to each pixel of the digital detector is converted into pixel value PV through the response function of the detector:

$$PV = b + a \cdot \ln(K_{air})$$

where K_{air} is the incident air kerma (kinetic energy released per unit mass) equal to total dose D, a and b are the fit parameters. Our tool offers three different detectors: two indirect digital radiography (D.R.), the former Philips DigitalDiagnost and the latter Kodak DR 7500, and a computed radiography (C.R.) system, the Kodak DirectView C.R. 900. It also offers two types of Look-Up Table (LUT) for the visualization of radiological image: the first one linear for the raw images and the second one sigmoidal, i.e. the standard LUT for these detectors.

Noise and random numbers. In addition to the response function, for each detector is carried out an analysis of the noise, as follows. The variance σ^2 is decomposed into its basic components [11]:

$$\sigma^2 = \alpha \cdot D + \beta \cdot D^2 + \gamma$$

where α, β and γ are the weight coefficients of Poisson, multiplicative and additive noise respectively. Taking into account the trends of variance, for each pixel of the simulated image is calculated a dose depending from noise. The necessary Gaussian variables are generated through the *Box-Muller* method [12]: let U_1 and U_2 be random numbers having the same probability density function, then:

$$X_1 = \sqrt{-2\ln U_1} \cdot \cos 2\pi U_2;$$
$$X_2 = \sqrt{-2\ln U_1} \cdot \sin 2\pi U_2;$$

are independent and identically distributed random variables having normal distribution with zero mean and unit variance. The random numbers (U_1 and U_2) for this method are obtained by a combination of three Tausworthe generators [13] (*TausStep*) with a linear congruential generator (LCG) [14] without mod operation (*LCGStep*), as suggested by *S. Mohanty et al.* [15], with different starting seeds:

$$\begin{aligned} U_1 = & (TausStep(seed_1, 13, 19, 12, 4294967294UL) \quad \oplus \\ & TausStep(seed_1, 2, 25, 4, 4294967288UL) \quad \oplus \\ & TausStep(seed_1, 3, 11, 17, 429496280UL) \quad \oplus \\ & LCGStep(seed_1, 1664525, 1013904223UL)) \quad \cdot \\ & 2.3283064365387 \cdot 10^{-10}; \end{aligned}$$

where U_2 is calculated as U_1 using $seed_2$ instead of $seed_1$. The starting seeds are the result of another linear congruent generator:

$$seed_i = id_i \cdot 1099087573UL;$$

where id_i is a combination of the row and column indexes of the image with GPU thread and block numbers to avoid correlated noise. Four combination id_i implement three noise contributions:

$$id_1 = \sqrt{i \cdot j} \cdot (thread_x + block_y);$$
$$id_2 = \sqrt{j \cdot i} \cdot (thread_y + block_x);$$
$$id_3 = id_4 = \sqrt{i \cdot j} \cdot (i + j);$$

where i and j are the row and column indexes of the image respectively; $thread_x$, $thread_y$ and $block_y$, $block_x$ are the thread index within the block and the block index within the grid respectively. id_1 and id_2 are used for the calculation of X_1 and X_2, the former for the Poisson noise and the latter for the multiplicative one; id_3 and id_4 are used for calculating the X_1 variable for the additive noise. This combination is adapted from the image generation field applying the *Park-Miller* algorithm proposed by *W.B. Langdon* [16] to $seed_i$ before inserting it in the *TausStep* and *LCGStep* combination. It is very important that each pixel has its own starting seed, different and independent from those of the other pixels, because "miscalculated" random numbers produce correlated noise.

We decided to implement this complex approach for the simulation of noise due to the limitation of CUDA about the generation of random number. It is a widely discussed argument in GPU computing, and from CUDA 2.0 it is available a library called CURAND, that allows to generate pseudorandom and quasirandom numbers. Despite the sequence of numbers satisfies most of the statistical properties of a truly random sequence, it is generated by a deterministic algorithm. It means that if each thread always starts with the same initial seed, it is always generated the same sequence of numbers, losing the randomness. As a consequence such method is not suitable for imaging, in particular for the simulation of the statistical fluctuation of noise.

III. Graphical Tool

The main window of our program, shown in Figure 1, is composed by three panels: the first one represents the virtual radiological room; the second one displays the preview of the radiographic image; the third one shows the histogram of the acquired image. The input of the application is a set of tomographic images, in which each pixel provides information about the density of the voxel, or rather of the represented

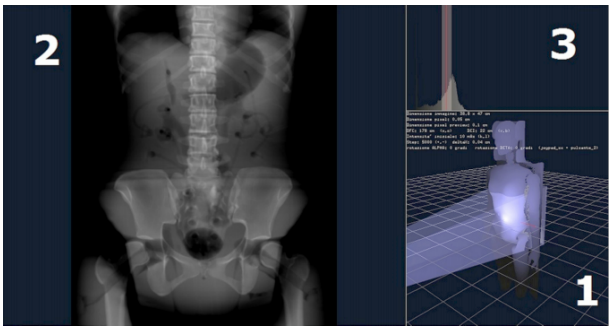


Fig. 1: Display window of the simulation tool.

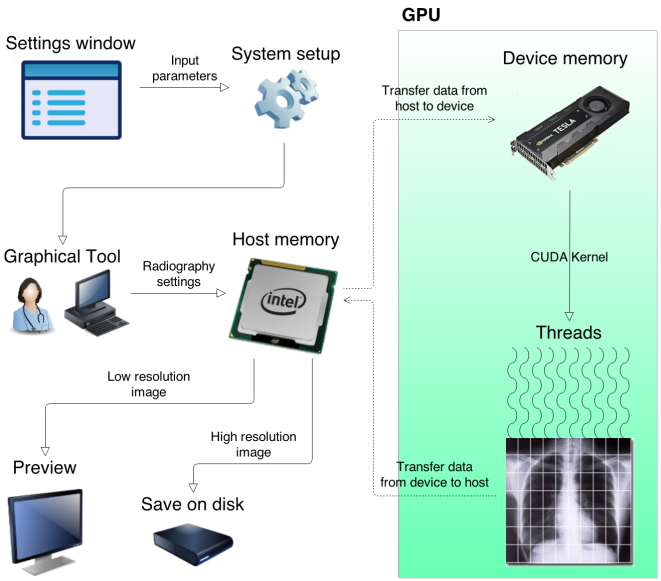


Fig. 2: Data flow diagram of the simulation tool.

tissue. A 3D matrix of the phantom is built from the set of voxels, and then loaded into the GPU memory. The user can set, through a simple settings window, some parameters involved in the X-ray exam, such as energy (photon spectrum) and intensity (mAs) of the radiation beam, radiation field size (i.e. the image sizes), two rotation angles of the source-detector system (alpha and beta), the distance between source and phantom (DFC) and between phantom and detector (DCI), the number of steps, phantom, detector, LUT type, noise component and also the GPU model. The simulation starts with a standard configuration of the position of the source-detector system, and provides a radiography preview that has lower resolution with respect to the final image, in order to speed up the application. Some parameters can be modified during the virtual exam, through keyboard or gamepad controls, such as the beam intensity, the beam collimation, the number of steps, the value of alpha, beta, DFC and DCI. The image preview is updated after every change, in order to permit a real-time interaction. When all these parameters satisfy the user, it is possible to generate the radiography as an high resolution version of the image in preview, that is saved on disk in *tiff* format. The entire process is schematized in the diagram shown in Figure 2. The X-ray image is generated on the GPU using the algorithms presented in Section II, summarized in Algorithm 1, executed in parallel: each thread computes an image pixel.

IV. Results

Our tool produces physically-realistic X-ray images in real-time. Below, we provide a brief description of the software validation (as this goes beyond the scope of the paper), and the comparison of computing times. For testing we used three different NVIDIA graphics cards: Tesla C1060 (240 cores, 4GB of memory, released in 2008), GeForce GTX 680 Kepler (1536 cores, 2GB of memory, released in 2013) and Tesla K40c (2880 cores, 12GB of memory, released in 2014).

The validation of the software was based on the comparison

Algorithm 1 CUDA kernel function

```
for each pixel of the image do
    initialize variables;
    for each steps do
        compute the linear attenuation coefficient μ_voxel;
    end for
    for each energy bin do
        compute the attenuated photon fluence Φ_att_i;
        compute the dose contribution D_i;
    end for
    compute the total dose incident in the pixel center D;
    compute the noise;
        # here the generation of random numbers
    convert the total dose in gray level value;
end for
```

between acquired and simulated images, using different set of input parameters and two different phantoms. For each image, both the simulated and the actual one, we selected some regions of interest (ROIs) for making a comparison between the respective gray levels.

In the first validation phase, we used a cubical phantom of plexiglass. Since it has a uniform density, it is ideal for testing the correctness of our computation algorithm of gray levels. Figure 3 shows the Gaussian noise distribution of a central ROI of the image ($81kVp$, $12.8mAs$, $DCI = 7.6cm$, $DFC = 172.4cm$, $\alpha = \beta = 0$, $step = 2500$) for the DirectView D.R. 7500 detector. Mean pixel value and standard deviation are reported for the three graphics cards: the distributions obtained are similar.

In the second validation phase, we used an anthropomorphic phantom (3 Dimensional Torso, model 602, CIRS, Tissue Simulation and Phantom Technology, Norfolk, Virginia, USA). We have acquired and simulated images of five different clinical exams: abdomen, anterior-posterior (AP) vertebral column, lateral vertebral column, posterior-anterior (PA) chest and lateral chest. The validation was repeated on all the available GPU, with similar results, but for sake of brevity we report only the results obtained with the GeForce GTX 680. Figure 4 shows a qualitative comparison between the real and the simulated radiography of lateral chest (exposure parameters $117kVp$ and $12.5mAs$), without the contribution of scatter

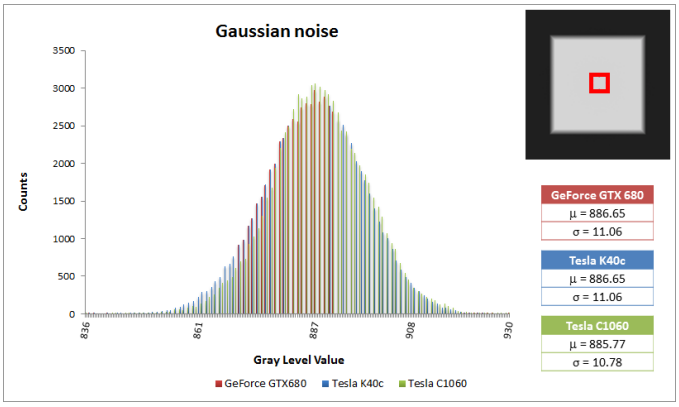


Fig. 3: Gaussian distribution of noise for the radiography of cubic phantom using the detector Kodak DirectView D.R. 7500.

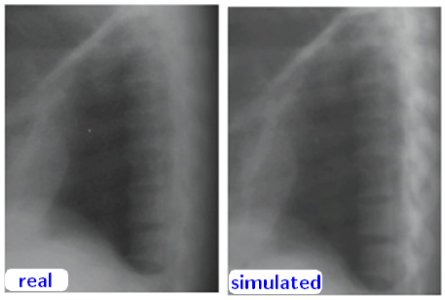
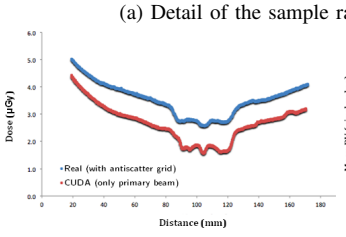


Fig. 4: Sample radiography of lateral chest.

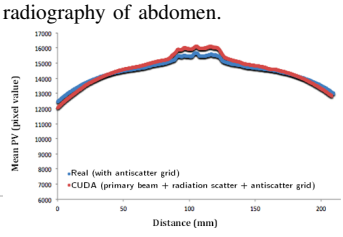
radiation. Figure 5 shows some analysis on the simulated radiography of abdomen (a). Real images were acquired using an anti-scatter grid, i.e. a device, placed between the patient and the detector, for limiting the amount of radiation scatter created during the exposure. The first graph (Figure 5b) is related to the trend of dose: the profiles are quite similar and the shift is due to the absence of the contribution of scatter radiation and anti-scatter grid in the simulated image. The second graph (Figure 5c) shows the trend of gray levels, the most significant analysis for radiologists. In this case we added the contribution of scatter radiation and anti-scatter grid to the simulated image, in a post-processing phase, by applying a Gaussian filter. It can be noted that the two graphs are very similar.

The application was tested on the three GPUs listed above and on a CPU (PC Intel Xeon X5560 @ 2.80 GHz 2.78 GHz, 4 cores (2 logical cores per physical), 48 GB RAM), in order to make a comparison of the performance on different devices. The CPU version of our algorithm runs on single core. For the test we used the anthropomorphic phantom CIRS and two detectors: the Kodak DirectView C.R. 900 and the Philips DigitalDiagnost. The pixel size of the Kodak detector is $0.17 \times 0.17mm^2$, whereas the one of the Philips detector is $0.14 \times 0.14mm^2$. We repeated the test on each of the five X-ray exams listed above. The results are reported in Figure 6a and depicted in Figure 6b. The computation time on GPU includes also the data transfer time between host and device. The Figure 6b shows a scatter plot in which we assign the computational time for simulating the Kodak detector to the horizontal axis,

(a) Detail of the sample radiography of abdomen.



(b) Trend of dose.

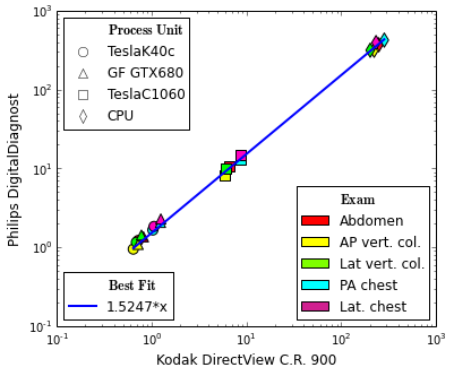


(c) Trend of gray levels.

Fig. 5: Statistical analysis on real and simulated radiography of abdomen.

Exam	Kodak DirectView C.R. 900			
	TeslaK40c	*GF GTX680*	*TeslaC1060*	*CPU*
Abdomen	0.72	0.82	6.63	249.31
AP vertebral column	0.64	0.72	5.88	222.84
Lateral vertebral column	0.69	0.78	6.08	202.45
PA chest	1.03	1.24	8.69	285.23
Lateral chest	1.06	1.25	8.64	233.26
	Philips DigitalDiagnost			
Abdomen	1.22	1.36	10.63	370.15
AP vertebral column	0.95	1.09	8.24	316.15
Lateral vertebral column	1.16	1.43	10.07	316.25
PA chest	1.66	2.07	13.11	424.65
Lateral chest	1.84	2.29	15.06	394.68

(a)



(b)

Fig. 6: Computational times (in seconds), on different devices, for simulated images of anthropomorphic phantom CIRS, using two detectors.

and the computational time for simulating the Philips detector to the vertical axis; the shape states the process unit (GPUs and CPU) while the color states the medical exam. As expected, a lower pixel size of the detector requires more computational time. In particular by applying simple linear regression through the origin on the data (the blue line in the graphic) we can notice that the correlation between the two times is on average equal to 1.52. Looking at Figure 6a we can make a comparison of computing times with respect to the fastest device, the Tesla K40c, noting that on average it is:
- 1.2 times faster than the GeForce GTX 680, because they are both very recent GPUs, designed for high performances;
- 8.6 times faster than the Tesla C1060: this value gives an idea of the improvements done in GPU architecture in the last five years;
- 245 times faster than the CPU version: it is a very significant difference that highlights the advantages of the parallel computing on GPU.

V. Conclusion

In this paper we presented a simulation tool for digital X-ray imaging that takes advantage of the computational power of the graphics processing unit by using the CUDA parallel computing platform. Our tool reproduces a radiological room and simulates the main aspects of a real radiological exam. We implemented an efficient generation of random numbers, entirely computed on GPU, in order to obtain a real noise distribution and to simulate physically-realistic medical images. The software allows a real-time interaction: for this reason it can be used in a radiological department for training health operators and for dose optimization processes, saving the time of the radiologist and reducing the occupancy time of the radiological room, and for reducing the radiation dose to patients. Since our tool is designed to be used in a hospital, in our opinion the GPU-based method is a more attractive solution with respect to the CPU cluster-based method, because it is cheaper, needs less maintenance and power consumption, and has higher fault tolerance. Since different radiological departments may have at their disposal different hardware, we tested the program on different GPUs to ensure its correct operation in real-time. Moreover, our tests show increasing performance on more recent GPUs, and they confirmed that the Tesla K40c is the fastest graphics card available on the market. Furthermore, in contrast with the CPU version, the Tesla C1060, our oldest card, is around 35 times faster, and the Tesla K40c, our newest card, is around 250 times faster.

References

[1] 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.

[2] G. Pratx and L. Xing, "GPU computing in medical physics: A review," *Medical Physics*, 2011.

[3] F. Ino, J. Gomita, Y. Kawasaki, and K. Hagihara, "A gpgpu approach for accelerating 2-d/3-d rigid registration of medical images." in *ISPA*, ser. Lecture Notes in Computer Science. Springer, 2006.

[4] D. Ruijters, B. M. ter Haar-Romeny, and P. Suetens, "Gpu-accelerated digitally reconstructed radiographs," in *Proceedings of the Sixth IASTED International Conference on Biomedical Engineering*, ser. BioMED '08. ACTA Press, 2008.

[5] O. Dorgham, S. Laycock, and M. Fisher, "Gpu accelerated generation of digitally reconstructed radiographs for 2-d/3-d image registration," *Biomedical Engineering, IEEE Transactions on*, 2012.

[6] F. P. Vidal, M. Garnier, N. Freud, J. Létang, and N. W. John, "Simulation of x-ray attenuation on the gpu," in *TPCG*, W. Tang and J. P. Collomosse, Eds. Eurographics Association, 2009.

[7] C. Chou, Y. Chuo, Y. Hung, and W. Wang, "A fast forward projection using multithreads for multirays on gpus in medical image reconstruction." *Med Phys*, 2011.

[8] K. Viswanathan and K. Balasubramaniam, "Modeling and simulation schemes in x-ray radiography and computed tomography," 2009.

[9] A. J. Reilly, "Report 78-spectrum processor." Institute of Physics and Engineering in Medicine, 1997.

[10] *Tissue Substitutes in Radiation Dosimetry and Measurement*. International Commission on Radiation Units and Measurements, 1989.

[11] A. I. di Fisica Medica, "Apparecchi di radiografia digitale diretta amfpi, linee guida per i controlli di qualità," 2009.

[12] G. E. P. Box and M. E. Muller, "A note on the generation of random normal deviates," *The Annals of Mathematical Statistics*, 1958.

[13] R. Tausworthe, *Random Numbers Generated by Linear Recurrence Modulo Two*. Jet Propulsion Laboratory, National Aeronautics and Space Administration, (U.S.), 1965.

[14] C. Fontaine, "Linear congruential generator," in *Encyclopedia of Cryptography and Security*, H. C. A. van Tilborg, Ed. Springer, 2005.

[15] S. Mohanty, A. K. Mohanty, and F. Carminati, "Efficient pseudo-random number generation for monte-carlo simulations using graphic processors," *Journal of Physics: Conference Series*, 2012.

[16] W. B. Langdon, "A fast high quality pseudo random number generator for nvidia cuda," in *GECCO (Companion)*, F. Rothlauf, Ed. ACM, 2009.