Medicine Meets Virtual Reality 11
J.D. Westwood et al. (Eds.)
IOS Press, 2003

Real-Time PC based X-ray Simulation for Interventional Radiology Training

Manivannan Muniyandi[1], Stephane Cotin[2], Mandayam Srinivasan[1], Steven Dawson[2]

[1]*Laboratory for Human and Machine Haptics, Massachusetts Institute of Technology, Cambridge, MA 02139*

[2]*The Simulation Group, Massachusetts General Hospital-CIMIT, Harvard Medical School, Boston, MA 02114*

Abstract. The ability to simulate realistic fluoroscopic images in real-time is a key aspect of any interventional radiology training system. In this paper, we propose a method for rendering X-ray images in real-time on a PC with consumer level graphics hardware, while improving the quality of the images. Although volume rendering techniques form the basis of our algorithm, we studied the characteristics of actual X-ray images to develop a method that can provide a new level of realism. In addition, the integration of the various levels of information contained in a CT scan in the rendering pipeline can be exploited to produce even more realistic, patient-specific X-ray or fluoroscopic images. Although the results presented here are preliminary, the performance of multi-texturing and multi-stage rasterization features available on recent low-cost graphics hardware already allows us to render X-ray images at about 30 frames per second.

1. Introduction

Interventional radiology[1] represents a revolutionary advance in the field of vascular disease treatment but requires advanced skills in mentally reconstructing three-dimensional anatomy from two-dimensional fluoroscopic images and demands excellent hand-eye coordination. To address the need for training, the traditional master-apprentice model using animals or human patients appears as an inherently inefficient method, involving ethical problems of animal-based training, radiation exposure for the trainees, and the use of expensive devices and resources. On the other hand, the recent developments of computer-based training systems could offer a significant practical value, especially for training technical procedures or tasks [1, 2, 3].

Since interventional radiology is a medical discipline that replies extensively on imaging techniques – intra-operative X-ray images mostly – it seems that the visual feedback provided by a training system should offer a level of quality similar to what is available in the real world. There are three different classes of algorithms that have been developed in the context of interventional radiology training: actual fluoroscopic images, polygon-based techniques, and volume rendering techniques. While the use of actual fluoroscopic images limits the interaction to a fixed viewpoint, polygon models require that every anatomical structure to be rendered is previously segmented, thus limiting the level of detail that can be rendered. Therefore it seems that using a method relying on volume rendering,

[1] The branch of radiology concerned with providing diagnosis and treatment of disease by a variety of percutaneous procedures performed under the guidance of x-ray imaging.

combined with a high-resolution CT scan volume of the anatomical structure of interest, would be an ideal approach.

There are many possible ways to implement 3D volume rendering depending on the goal of the application, but very few actually address the problem of rendering X-ray images from CT scan volumes at real-time frame rates. Moreover, the extensive use of textures, and large amount of tri-linear interpolation necessary to generate high-quality images, has restricted the implementation to high-end graphics workstations with special-purpose hardware [4] and/or parallel implementation [5]. In low-end graphics workstations the data is preprocessed at the expense of image quality and realism to achieve real-time.

In this paper we present a novel technique for X-ray or fluoroscopic image simulation exploiting the recent developments in modern consumer graphics boards. Our approach exploits multi-texturing and multi-stage rasterization features of these low-cost graphics processors without compromising image quality or real-time interactivity.

2. X-ray and fluoroscopy principles

The basic principle upon which X-ray imaging methods depend is the attenuation of the X-ray beam as it passes through tissues of different densities. The X-ray beam is produced in an X-ray tube and exits from the tube where it is collimated, then enters the patient in the area of interest, and finally is partially absorbed by the patient's tissues. The portion of the beam that penetrates the patient reaches a cassette that contains screens. The X-rays energize crystals in the screens, which produce light proportional to the amount of X-ray energy that energized them. The light generated by the screens exposes radiographic film, which is processed to produce an image.

Following the same principles, fluoroscopy is an X-ray test that uses a continuous beam of X-rays to follow movement in the body. During fluoroscopy, X-rays are directed continuously at an area of the body, and the resulting pictures are displayed on a monitor similar to a TV screen. Fluoroscopy can be used to evaluate the movement of the lungs or heart but is mostly used to guide the placement of instruments or devices in the body, such as during angiography. A contrast agent that shows up on X-rays can be injected into a blood vessel during fluoroscopy to make the outline of blood vessels visible. It is in part because of the continuous exposure of the patient to X-ray radiation that a high level of training could help minimize exposure time.

In order to simulate accurately the fluoroscopic process, we must first generate high quality X-ray images and second generate these images at a frame rate of about 20 images per second. Since photon energy, atomic number, density, electron density, and thickness of the material affect absorption and scattering of the photon [8], our approach and main contribution will be to attempt at simulating as many characteristics as possible, in real-time, to produce highly realistic images.

2.1 X-ray beam intensity attenuation

The attenuation of an X-ray beam traversing a thin slice of homogeneous material is given as:

$$I = I_0 e^{-\mu d} \quad (1)$$

with I_0 the input intensity, μ the coefficient of linear attenuation of the material, which, and d the traversed length. The attenuation coefficient varies with the cube of the atomic number of the material being traversed. Therefore, mass attenuation coefficient of bone (atomic number about 11.6) is significantly higher than the fat (atomic number 7.4) explaining why bone and fat appear highly contrasted in an X-ray image.

When the beam traverses several structures of various thickness and attenuation coefficients, the previous continuous equation can be rewritten as:

$$I = I_0 e^{-\sum_i \mu_i d_i} \quad (2)$$

with μ_i the linear attenuation coefficient of structure i, and d_i the thickness of the structure. One can also notice that this equation can be used to describe the change in intensity of a beam that would traverse a discretized representation of the anatomy – like a CT scan for instance – where d_i would correspond to the slice thickness along the ray and μ_i be the attenuation coefficient of the anatomic structure sampled in slice i. Finally, it is important to remember that attenuation is usually combined with scattering effects that play a role in the final appearance of the X-ray image. Assuming a narrow well-collimated mono-chromatic x-ray beam, the attenuation coefficient absorption coefficient is the sum of contributions of absorption and scattering effects. However, it remains a function of the beam intensity I, $\mu = \mu(I)$ as illustrated by Figure (1).

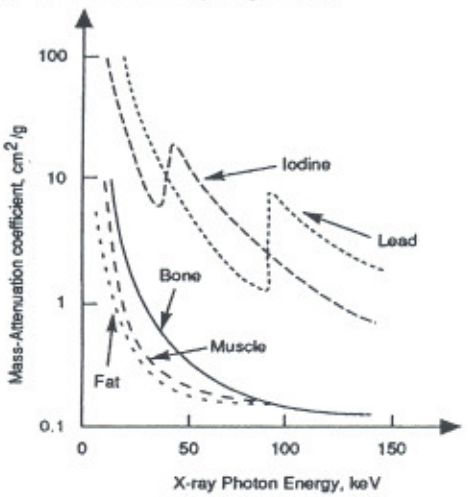

Figure 1 - Non-linear relationship between X-ray photon energy and attenuation coefficient of various tissue, adapted from [9].

2.2. Relationship between bean attenuation and CT scan characteristics

Equation (2) is a key to the simulation of X-ray images using volume rendering techniques, as it will be illustrated in section (3). By computing the contribution of each slice of the CT scan image to the decrease of energy of the X-ray beam, and then by summing all the contributions, we can compute a realistic X-ray image. The first step requires knowing what will be the attenuation of a ray passing through a voxel of a given intensity. This can be computed since the intensity of a voxel in a CT data volume is directly related to Hounsfield Units[2] and the attenuation coefficient μ can be determined from Hounsfield Units [7]. Similarly, slice thickness and inter-slice distance parameters of the CT scan allows us to measure d. What remains to be computed are $e^{-\mu_i d_i}$ and the sum of the contribution of each slice, as we will discuss in section (3).

[2] Named after the inventor of the CT scanner, it is the basic unit of measurement used in computed tomography, providing a measure of radiodensity.

3. X-ray and fluoroscopy simulation

A common approach to the problem of X-ray rendering is to process CT volumetric images [1, 6, 7] since CT images are actually created by combining a radial sequence of X-ray images. One approach consists in applying almost the inverse process of what is used in Computed Tomography by computing Fast Fourier Transforms to extract the frequency spectrum of the projection of the data set in the Fourier domain and then computing the projection by applying an inverse 2D Fourier transformation to that spectrum [8]. By reducing the computational complexity from 3D to 2D this allows for faster rendering although not real-time. Another approach is to use a volume rendering technique. Volume rendering is based on light interaction with the particles – voxels – in a volume of data. In the Ray Casting approach [6], for each ray cast through the data set, the stepping and interpolation stages generate and accumulate a number of sample points along that ray. By specifying how the intensities of voxels traversed by a ray are combined with each other, it is possible to generate images similar to X-ray images. A third way of dealing with the problem consists in applying similar principles, but instead of dealing with very large volumes of voxels, the X-ray image is generated from polygonal models [2]. Although it allows the creation of rather realistic images, in real-time, this technique requires segmenting the anatomy, defining an attenuation coefficient for each structure, and evaluating the length of each polygonal volume that is penetrated by the beam, thus making it less flexible and accurate than volume rendering techniques. Our approach uses hardware implementation of display traversal in graphics boards to accelerate volume rendering. In this paper, we use rendering techniques that exploit multi-texturing capabilities of modern consumer graphics boards to improve both performance and the image quality.

3.1. X-ray beam attenuation as a transfer function

From our discussion in section (2.2), we know that the attenuation coefficient μ can be described as a function of the intensity of a voxel; therefore it can be described as a transfer function in the volume rendering process. Although the transfer function could be computed analytically, we followed a heuristic approach – in collaboration with an interventional radiologist – that allowed us to design a transfer function linking intensity in a CT scan with attenuation coefficient.

This was then implemented using *OpenGL* [10] and a paletted texture – a table of color indices – to store the transfer function. *OpenGL* provides an extension `glColorTableEXT` to map a 2D texture coordinate to a paletted texture. With another extension `GL_SHARED_PALETTE_EXT` we can maintain a single palette for all textures thus reducing the memory complexity. Although the transfer function describes the attenuation coefficient at each traversed voxel, an additional step is required to compute the actual contribution of the voxel in the final image, while ensuring the validity of Equation (1).

3.2 Combining data from CT scan slices

A good approximation of Equation (1) can be implemented in *OpenGL* by blending a texel element with the corresponding frame buffer element [9], taking advantage of recent graphics boards that provide programmable rasterization hardware. It allows to explicitly control how color, opacity, and texture components are combined to form the resulting fragment. With multi-stage rasterization rather complex calculations can be performed in a single rendering pass [11, 12]. To gain explicit control over this special hardware, *NVIDIA*™ [13] provided the *OpenGL* extension `NV_register_combiner`. With this

extension enabled, the standard *OpenGL* texturing units are bypassed and substituted by a register-based rasterization unit. This unit consists of eight extremely flexible general rasterization stages and one final combiner stage. One combiner stage is divided into an RGB-portion and a separate Alpha-portion. The Alpha-portion is designed similarly to the RGB-portion, but could be programmed independently. The output registers of the general combiner at any stage could be combined by a final combiner stage that is slightly different from general combiners. The core idea of our method is to use *NVIDIA*'s register combiner to blend several slices simultaneously but also compute the change in intensity due to the attenuation originating from each slice. Since the idea of using the combiner hardware for blending multiple slices has been described in [12], we only discuss it for completeness and instead will focus on the computation of the attenuation.

Equation (1) describes the decrease in intensity when passing through an object. If we consider a slice of the CT scan as a thin cross-section of the same object, then equation (1) becomes: $\Delta I = -\mu I \Delta d$ which can be used instead of Equation (2). Again assuming that Δd is known and constant, we can write $\alpha = \mu \Delta d$, which implies $\Delta I = I_d - I_s = -\alpha I_s$, i.e. $I_d = I_s(1-\alpha)$ which can be implemented easily using *OpenGL*.

In the following discussion the character I indicates the intensity of a pixel and A refers to an alpha value – i.e. a rescaling of μ from 0 to 255 to follow *OpenGL* requirements. Subscripts t_0 and t_1 indicate the texture values of the first and second texture map, while d and s indicate a destination and source value of an incoming fragment respectively. Using `glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA)` for back-to-front X-ray rendering and the *GL_REPLACE* texture environment, the resulting color value in the frame buffer after blending the first textured polygon amounts to:

$$I_d' = I_{t0} A_{t0} + I_d (1 - A_{t0})$$

$$I_d'' = I_{t1} A_{t1} + I_d' (1 - A_{t0}) = I_{t1} A_{t1} + I_{t0} A_{t0} (1 - A_{t1}) + I_d (1 - A_{t0})(1 - A_{t1})$$

This could be achieved using a register combiner if the output of the RGB portion of the combiner is:

$$I_s = I_{t0} A_{t0} (1 - A_{t1}) + I_{t1} A_{t1}$$

and the result of the alpha portion is directly used as alpha value of the output register:

$$A_s = (1 - A_{t0})(1 - A_{t1})$$

The resulting fragment is then blended into the frame buffer using the blending function `glBlendFunc(GL_ONE, GL_SRC_ALPHA)`.

3.3 Taking slice thickness into account

We made earlier the assumption that the slice thickness or inter-slice distance was constant in a CT scan. In reality, slice thickness for instance is not the same in all principle directions – usually a voxel is not a cube but a parallelepiped elongated in the direction normal to the slice plane. To address this problem the CT data could be interpolated in a preprocessing step; this is however not a viable solution especially when the volume data is of very large size – texture memory in consumer graphics hardware is usually limited to 64 MB or 128 MB. Another way to address this issue is to exploit one more time the register combiner in the graphics hardware. Register combiners are used to interpolate slices on the fly before rendering them. With *NVIDIA*'s hardware this takes a little overhead. The hardware set up for this approach is shown in Figure (2).

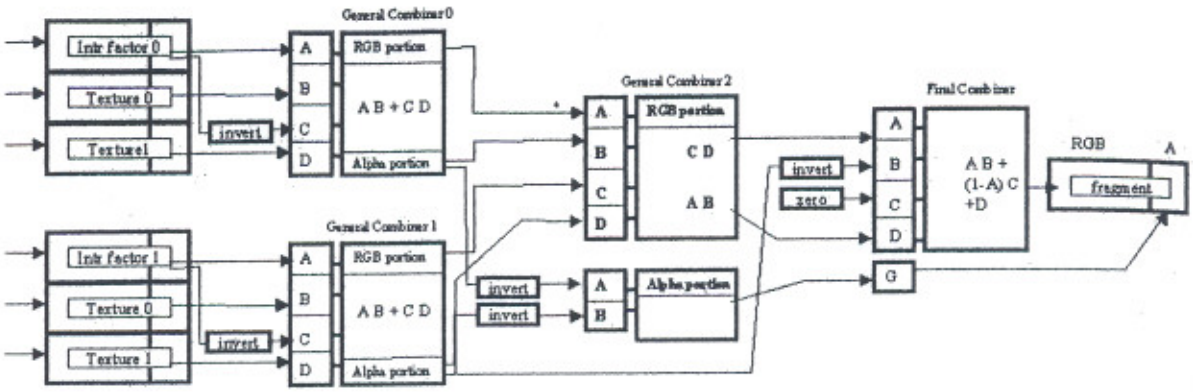

Figure 2 – Register combiner setup for on-the-fly slice interpolation and blending.

4. Results and conclusion

To validate the quality of the simulated X-ray images and the speed of the algorithm we used a set of head CT scans of different resolutions. The quality of the images was evaluated by a radiologist while the speed was measured in frames per second at a screen resolution of 1280x960. The results are illustrated in Figure (3) and Table (1).

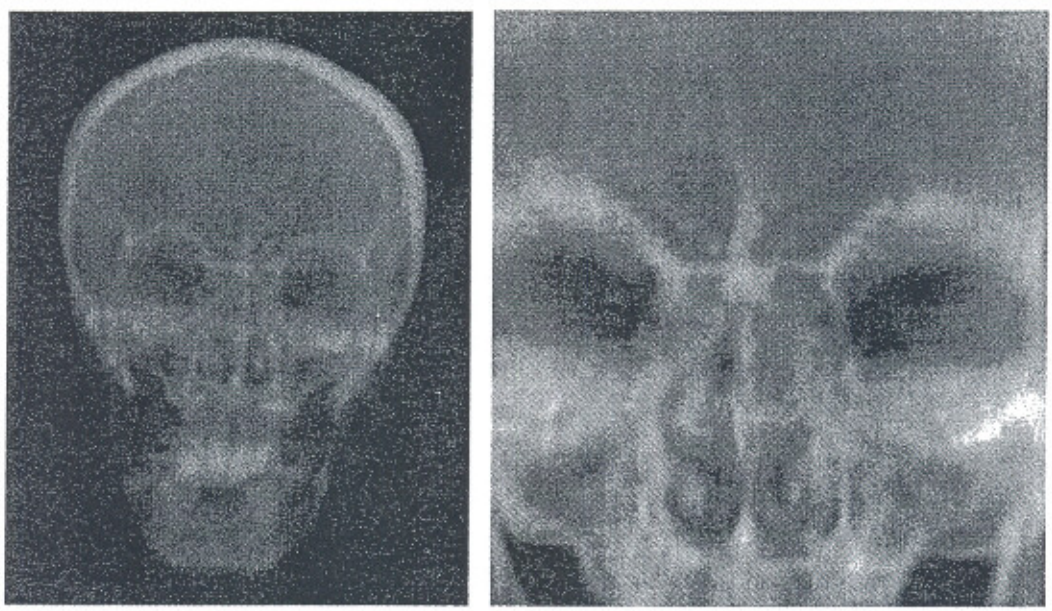

Figure 3 – Simulated X-ray image of the head (left) and close-up (right).

Volume Size	256x256x106	256x256x256	512x512x512
Frame Rate	30.2	20.2	4.5

Table 1 - X-ray simulation fram rates of different sizes of volume data.

In this paper we demonstrated that advanced features of recent consumer graphics hardware such as multi-texturing and multi-stage rasterization could be used for realistic

and real-time X-ray simulation. In addition, we illustrated that it is possible to reproduce accurately the actual interaction of X-ray photons with matter by using *OpenGL* and register combiners. Although this technique has been implemented on *NVIDIA* hardware, it can be generalized to other hardware.

While the results presented in this paper describe a work in progress, the quality of the simulated X-ray images has been evaluated by radiologists as very high. Moreover, since our method directly relies on CT data, it is anticipated that the quality will improve with CT scan technology. Also, this work is part of an effort to develop a complete endovascular training system including realistic haptic feedback, physical modeling and fluid flow computation. However, these results will enable other X-ray based diagnostic and procedures – such as angioplasty, mammography, or xeroradiography – to be simulated as well.

Acknowledgements

The authors would like to acknowledge Paul Neumann for his helpful comments.
This work was supported by the U.S. Army Medical Research Acquisition Activity under contract DAMD 17-02-2-0006. The ideas and opinions presented in this paper represent the views of the authors and do not, necessarily, represent the views of the Department of Defense.

References

[1] R. Mullick, H. T. Nguyen, Y. P. Wang, J. K. Raphel, and R. Raghavan. Overview of Visible Human® based applications at CieMed, The Visible Human Project Conference, 1996.

[2] S. Cotin, S. Dawson, D. Meglan, D. Shaffer, M.Ferrell, R. Bardsley, F.Morgan, T. Nagano, J. Nikom, P. Sherman, M.Walterman, J. Wendlandt. ICTS: An Interventional Cardiology Training System, Cathet Cardiovasc Intervent 2000; 51: 522–527.

[3] SimSuite, Interventional Radiology Simulation, http://www.medSimulations.com

[4] T. Cullip and U. Newman. Accelerating volume reconstruction with 3d texture hardware. Technical Report TR93-027, University of North Carolina, Department of Computer Science, Chapel Hill, NC., 1993.

[5] J. P. Singh, A. Gupta, and M. Levoy. Parallel visualization algorithns: Performance and architectural implications. *IEEE Computer*, 7(27):45–55, July 1994.

[6] Y. Park, R. Lindeman, J. Hahn. X-ray Casting: Fast Volume Visualization Using 2D Texture Mapping Techniques, Seventh IEEE Visualization conference, San Francisco,CA. October, 1996

[7] T. van Walsum, K. Zuiderveld, J. Chin-A-Woeng, B. Eikelboom, M. Viergever (1997). CT-based simulation of fluoroscopy and DSA for endovascular surgery training. In Proceedings CVRMed-MRCAS '97, Lecture Notes in Computer Science, Vol. 1205, Springer-Verlag, Berlin, pp. 273--282.

[8] S. Napel, S. dunne, and B. Rutt. Fast Fourier Projection for MR angiography, Magnetic Resonance in Medicine, 19, 393-405 (1991)

[9] K. Shung, M. Smith, B. Tsui, Principles of Medical Imaging, Academic Press, 1992.

[10] The OpenGL Programming Guide 3rd Edition. The Official Guide to Learning OpenGL 1.2.

[11] R. Westermann, T. Ertl: Efficiently using Graphics Hardware in Volume Rendering Applications. In Proc. of SIGGRAPH 1998

[12] C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, T. Ertl. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization. In Proc. Eurographics/SIGGRAPH Workshop on Graphics Hardware, 2000.

[13] NVidia corporation, http://www.nvidia.com