Re-slicing Tomographic Volumes with Shell Rendering

CESAR AUGUSTO DE CARVALHO VANNINI[1]

[1]USC–Universidade do Sagrado Coração - R. Irmã Arminda, 10-50, 17011-060 Bauru, SP, Brasil
gvannini@usc.br

Abstract. In this work we show a technique based on Shell Rendering to process a volume composed by a set of tomographic images in such a way that it can be rotated and re-sliced in a new volume in an appropriate angle. We also propose two different techniques for dealing with quality of rendered images. In the last case, the quality of the new images is comparable to the original ones.

1 Introduction

In medicine, tomographic images are used for the purpose of better understanding the structures present in the human body. The image diagnostic, the surgery planning [9], the medical education and the study of internal body events are examples of applications that benefit from and require the visualization of these images. Tomographic images are acquired by special devices such as X-Ray Computer Tomography (CT) or Magnetic Resonance Imaging (MRI). Each tomographic image corresponds to a 2D image of a transversal cut of the patient. The sequence of the tomographic images make a volume or a scene.

The best tomographic devices can acquire tomographic images in three different planes: the longitudinal, sagittal or coronal planes of the patient. This can be a limiting feature to one's need of studying the growth of structures in a non covered plane or in a body incision in an non-straight angle. For this reason, this work proposes a technique for rotating the scene to an appropiate angle and re-slicing it such as the tomographic device. This technique is based upon the shell rendering technique for volume rendering. This work also proposes two different techniques for dealing with the rendering quality of images.

In the next sections, we describe the rendering technique used to render the final re-sliced images and its advantages compared to other techniques. We also describe the re-slicing process with the problems and their solutions using a slightly modified Shell Rendering algorithm. We demonstrate the application with the results and discuss some performance issues.

2 Rendering Techniques

Rendering techniques are used to transform a 3D model or object into a 2D image on the computer screen, providing this image with some 3D information such as linear perspective, shading or occlusion. Researchers have proposed several rendering techniques that can be classified into two major classes: surface rendering and volume rendering. In surface rendering, only the voxels in the vicinity of the object boundary are considered to contribute to the rendering, and an approximate representation of the surface of the object is computed using basic elements, such as voxels or polygons. In volume rendering, each voxel in the object has an opacity value assigned and is considered to contribute to the rendering, resulting in a semi-transparent volume [2, 5]. In this paper we are concerned only about volume rendering because the re-slicing uses its properties.

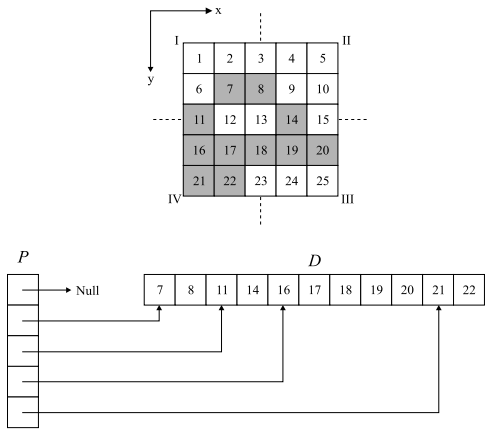


Figure 1: 2D shell structure example: (a) the original image with the pixels of interest in gray; (b) the correspondent array P of pointers and indexed list L for the image.

Shell Rendering is an approach suitable for both surface and volume rendering and is known as one of the fastest rendering algorithm in literature [1]. The shell structure is composed by a 2D array P of pointers and an indexed list L. By scanning the tomographic volume from the first to the last element (x, y, z-order), all the voxels with density value $f(v)$ greater than zero are stored in the list L, along with the coordinate x. Each element in $P(y, z)$ is a pointer to the first voxel at column x. A 2D shell structure example is depicted in Figure 1.

Shell rendering is an Object-order algorithm that op-

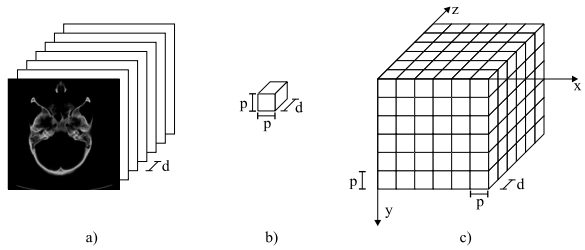


Figure 2: Forming a 3D scene: (a) original slices with d spacing; (b) a voxel; (c) the 3D scene.

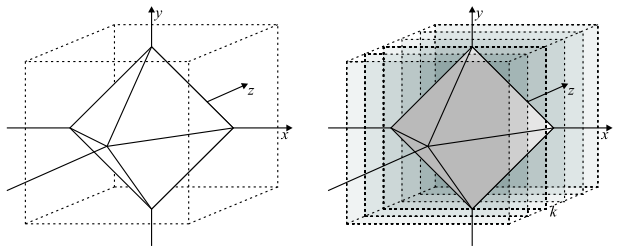


Figure 3: Re-slicing process: (a) the rotated volume and its domain; (b) the new re-sliced 3D scene.

erates by projecting voxels into the viewing plane while traversing the volume in a specified order [4]. In fact, it performs a direct voxel projection. One of the drawbacks with direct voxel projection, due to the nature of the projection transformation from the 3D volume domain to the 2D visualization plane domain, is the appearance of black holes in the rendered image. In order to overcome this problem, additional computation is required during or after the rendering.

3 Re-slicing Algorithm

The set of images generated by a tomographic scanner (i.e.: computer tomography (CT) or magnetic resonance imaging (MRI)) represents a 3D digital image data or a 3D scene. This data consists in a set of slices equally spaced parallel to one of the longitudinal, sagittal or coronal planes of the patient. Each voxel v in this volume is a cuboid with dimensions $p \times p \times d$ centered in a (x, y, z) coordinate, as the d dimension of the cuboid differs from the p dimension, depending on the spacing of the slices. The forming of a 3D scene is depicted in Figure 2. An interpolation operation is needed to convert this volume into an isotropic data set, where the voxel shape is a cube (i.e. $p = d$), to avoid distortions on the final rendered image. A simple linear interpolation algorithm can be used here to produce the isotropic data set [1, 7].

The next step consists of selecting the voxels of the object and inserting them on the shell structure. This particular shell is called body-shell, since it holds all the voxels of the object. The re-slicing is done by first rotating the data set with two angles of movement α and β (tilt and spin). In this operation, a new coordinate (u, v, w) is calculated for each voxel and stored in the indexed list L. For storing the new calculated coordinates, we used a slightly modified shell structure. The values of (u, v, w) were inserted in the L list, and the array P structure was kept the same. The new coordinates are given by the Formula 1 [8].

$$\begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = T_D \times R_\beta \times R_\alpha \times T_O \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (1)$$

Where:
(x, y, z) are the original coordinates for the voxel;
T_O is the origin translation matrix to center the object within center of the system;
R_α and R_β are the tilt (x) and spin (y) rotation matrixes;
$T_{D/2}$ is a translation matrix to restore the object to the initial position of the system;
(u, v, w) are the new calculated coordinates for the voxel.

The w value indicates the new slice that the voxel belongs to after the rotation. Figure 3a shows a cube in a flat line representing a tomographic volume rotated with $\alpha = 45^0$ and $\beta = 45^0$. The dashed cube represents the domain of the rotated cube, which is used to generate the new re-sliced 3D scene. Figure 3b shows the new re-sliced 3D scene, where the slice k is defined by w.

Generating the new set of re-sliced images is a straightforward process. By an exhaustive reading of the indexed list L, each pixel (u, v) can be rendered in an appropriate slice w using its density value $f(p)$.

4 Main Results

As described in Section 2, the final rendered image generated by direct voxel projection can present some black holes. There are many techniques for solving this problem, for example: considering the projection of the voxel to cover more than one pixel in the viewing plane or using a filter on the rendered image for removing black holes. The problems with these solutions are that the former generates undesired pixel shadow and the latter may eliminate fine details in the image.

We have tested two different approaches for dealing with this problem. The black holes are caused due to a bad

rounding of the projection coordinates to a discrete plane (x, y). This problem can be minimized by working with a projection plane with twice the resolution and a double rendering with two different roundings. The first rendering projects a normal sized pixel with the integer coordinate (u, v) multiplied by two. The second rendering projects a double sized pixel in the real coordinate (u, v) multiplied by two and then rounded to its integer value. This last rendering is resposible for projecting part of the poorly rounded non-projected pixels in the first rendering. At this point, the rendered image is a mess and has twice the resolution needed. A linear interpolation to achieve the final rendered image with normal resolution is not applicable. The final image can be done by considering only the pixels in even rows and columns of the double sized projection plane. The final result is an image with an average of 50% less black holes than the image rendered with the original algorithm.

The computational cost for this double rendered image and the quality of the results lead this work to another way of dealing with black holes. We noticed that the black holes are mainly one pixel wide, thus a simple linear filter is able to solve this problem better than the former presented solution. This filter searches for black pixels with 2-neighbour non-black in first a vertical and further in an horizontal fashion, and replaces its value with the average value of its neighbours. Since this kind of filter does not alter the values of gray pixels it does not blur the rendered image. We have found that using more than one iteration of this filter can completely solve the black holes problem, and four has been taken as the magic-number for this application. The performance of the linear filter and the visual quality of the images comparing to the double rendering algorithm have shown better results. It can be seen in Figure 4 the sucessive progression for eliminating black holes. Figure 4a depicts an image of a skull rotated with $\alpha = 58^0$ and $\beta = 90^0$ with no filter. Figure 4b is the same image generated with the double rendering and Figure 4c is a single rendered image with four iterations of the linear filter.

To generate the re-sliced images, a pre-calculated coordinate (u, v) of all voxels is stored along the L list. For each value of (α, β), the coordinates are calculated only once. Afterwards any slice can be retrieved without further calculation. Each slice is done by a 2D-render that searchs the list L for pixels $w = k$, where k is the number of the desired slice.

For testing the performance of the shell rendering re-slicing algorithm we used an AMD K6-II processor with 500Mhz of clock, 192Mb of memory and a SiS 305 video card with 16Mb of memory. The tests were done with three body-shells. The first is a CT of a dry skull, the second and third are MRI's from a knee and a throat. Table 1 compares the rendering times for the two methods of dealing with

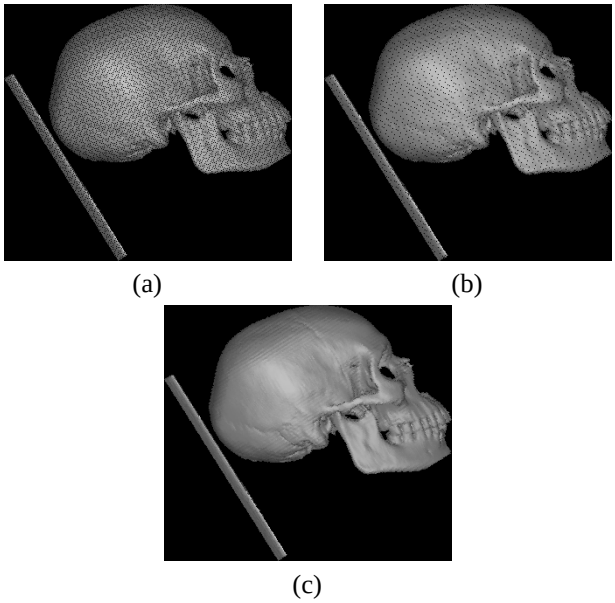


Figure 4: Black holes removal: (a) the original image with black holes; (b) double rendered image with better rounding; (c) a simple linear black holes removal filter.

	Voxels in Shell	no filter	double rendering	four iterations
skull	2.320.589	1.487s	2.295s	1.531s
knee	2.399.920	1.530s	2.339s	1.557s
throat	4.003.529	2.515s	3.891s	2.547s

Table 1: Shell size, rendering time for complete calcule of the rotation and generating a single slice.

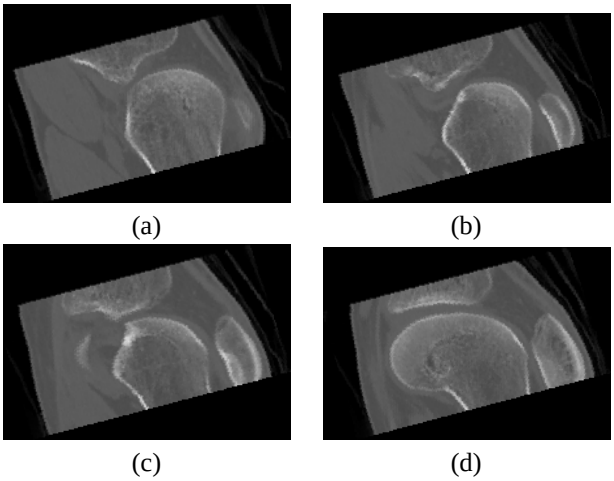
black holes with the time for a single re-slice rendering with no filter. It also shows the number of voxels pertaining to the shells.

Table 2 shows medium times for generating a single slice. These times do not include the time for rotating the shell and storing the rotated coordinates (u, v, w) in the L list. For this reason, the double rendering algorithm is not comparable with them, since it does not store the rotated coordinate but calculates it twice for each voxel projected every time. The number of the pixels in the projection plane were also included in this table, to clarify some time variations that happen when the four iteration linear filter is used.

The main results of the re-slicing can be seen in Figure 5 and Figure 6. The first sequence of images represents the re-sliced knee body-shell with $\alpha = 73^0$ and $\beta = 127^0$. Figure 5a through Figure 5d are slices with $w = [160, 170, 180, 190]$ respectively. The second sequence of images represents the re-sliced knee body-shell with $\alpha = 45^0$ and $\beta = 22^0$. Figure 6a through Figure 6d are slices with $w = [165, 175, 185, 195]$ respectively.

	rendered image size in pixels	no filter	four iterations
skull	433×433	0.091s	0.136s
knee	336×336	0.093s	0.120s
throat	370×370	0.154s	0.186s

Table 2: Medium times for generating a single slice after a pre-calculated rotation.



(a) (b) (c) (d)

Figure 5: Re-slicing results for knee rotated $\alpha = 73^0$ and $\beta = 127^0$: slice number (a) 160; (b) 170; (c) 180 and (d) 190.

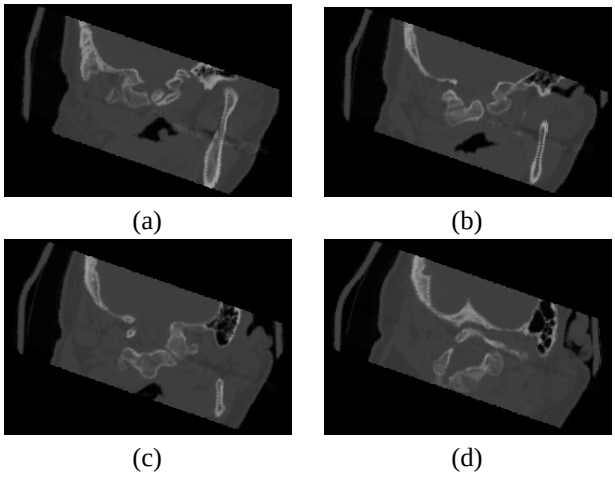
5 Conclusion

We have presented a technique to re-slice tomographic volumes based upon the shell rendering technique. It consists of using a slightly modified shell strucutre to store pre-calculated rotated values for each voxel in the shell and then uses these values to generate the re-sliced images. The visual quality of these images is comparable to the originals, and the speed of the algorithm enables a real-time visualization of the slices, that can be rendered in an approximate avegare of 0.15s each. We have also presented two different approaches in dealing with black holes left in the image by the voxel projection technique. The first consists of executing a better rounding on the projection coordinates, and the second, which is faster, consists of applying four iterations of a linear filter on the final rendered image.

References

[1] C. A. de C. Vannini, *Visualização Tridimensional em Medicina Usando Estereogramas Holográficos*, Master's thesis, Instituto de Computação, UNICAMP (Setembro 2000).

[2] J. K. Udupa and D. Odhner. *Shell Rendering*. IEEE Computer Graphics and Applications, 13(6):58-67, 1993.



(a) (b) (c) (d)

Figure 6: Re-slicing results for throat rotated $\alpha = 45^0$ and $\beta = 22^0$: slice number (a) 165; (b) 175; (c) 185; (d) 195.

[3] C. F. X. de Mendonça, A. X. Falcão, C. A. C. Vannini and J. J. Lunazzi. *Fast Holographic-Like Stereograms Display Using Shell Rendering and a Holographic Screen*. In proc. of SPIE on Medical Imaging San Diego, CA. (1999).

[4] J. K. Udupa and D. Odhner. *Fast Visualization, Manipulation and Analysis of Binary Volumetric Objects*. IEEE Computer Graphics and Applications, 11(6):53-62.

[5] P. Lacroute and M. Levoy. *Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation*. In proceedings of SIGGRAPH'94, pp 451-458, ACM, Orlando, Florida. "Computer Graphics, 28(4):451-458 (1994)"

[6] A. X. Falcão. *Visualização de Volumes Aplicada à Área Médica*. Masters Thesis, FEEC, UNICAMP (Fevereiro 1993).

[7] R. K. Ahuja, T. L. Magnanti and J. B. Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, Englewood Cliffs (NJ, 1993).

[8] R. Gonzalez and R. R. Woods. *Digital Imaging Processing*. Addison-Wesley, New York, (1993).

[9] M. W. Vannier, J. L. Marsj and J. O. Warren. *Three-dimensional computer graphics for craniofacial planning and evaluation*. Computer Graphics, 17:263-274 (1983).