2012 Cairo International Biomedical Engineering Conference (CIBEC)
Cairo, Egypt, December 20-21, 2012

Multipass GPU Surface Rendering in 4D Ultrasound

Ahmed F. Elnokrashy[1,2], Marwan Hassan[1], Tamer Hosny[2], Ahmed Ali[2], Alaa Megawer[1] and Yasser M. Kadah[1]
[1]Biomedical Engineering Department, Cairo University, Egypt
[2]IBE Tech, Giza, Egypt
E-mail: nokrashy@ibetech.com

Abstract—**4D ultrasound imaging extends the real-time capability of ultrasound to display a real-time volume, which can then be explored by the sonographer. Surface Rendering is one of the common modes used to display the 3D datasets. Surface shading is required to visualize the surface and enhances the surface contrast. Here, we show a surface rendering implementation based on three passes. In the first pass, ray casting is used with edge detection algorithm and then the edges are stored to viewing plan distance in the z-buffer of the 2D rendered image. In the second pass, filtration of the z-buffer is performed. This stage is so critical because the fuzzy nature of the ultrasound dataset then shade the 2D rendered image using image space shading. Finally, the third pass applies additional image processing mainly image smoothing on the rendered 2D image. The new method is applied to render volumes acquired on a research system and quality and computation time results show potential for clinical utility.**

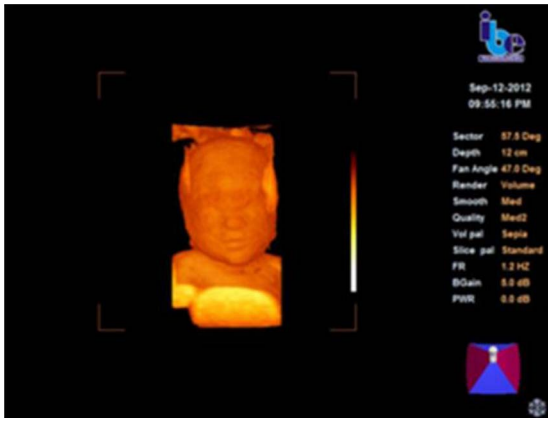
Keywords— **Surface Render, 4D ultrasound, Raycasting, Visualization, GPU.**

I. INTRODUCTION

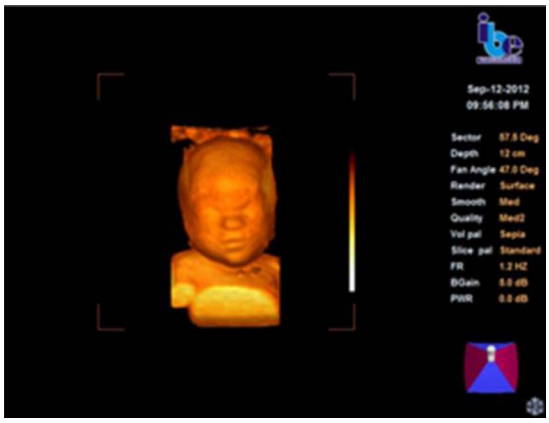
4D ultrasound surface rendering is a challenge for many reasons, noisy dataset, polar coordinate sampling and processing time constrains. Two-dimensional images of 3D objects require realistic shading to create the illusion of depth [5]. Fig. 1 compares rendering with surface shading and without surface shading. Implementation of a complete system requires several steps. The first step is rendering the volume dataset considering the problem of polar sampling of ultrasound dataset, also surface detection must be take place while raycasting the volume to generate the z-components of the rendered image, and due to the fuzzy nature of the ultrasound the volume rendering must include filtering of the data sets. The second step is Z-component filtration due to the noisy nature and surface discontinuities then uses the z-components to shade the surface. Finally, the third step includes additional smoothing that enhances the rendered 2D image.

The used platforms for the implementation include OpenGL, which draws the volume cube and Shading Language that raycasts the volume and coordinate transformation and filtration). Alternatively, surface shading can be utilized where traditional surface rendering algorithms convert the volume data into geometric primitives in a process known as iso-surface extraction. The geometric primitives (polygon mesh) are then rendered to the screen by conventional computer graphics algorithms [13]. Another method is based on gradient calculation where the gradient vector is calculated first then used to represent the normal vector which will be used for surface shading calculations [14].

In this work, image space z-buffer shading is used [5]. This method postpones the surface shading to the last stage of the surface rendering after the raycasting rendering. The only task added to the raycasting routine is the edge detection which does not need much additional computational effort. Working on a 2D texture to shade the surface showed a much better computation time performance than working with 3D texture, which allows more enhancement of surface smoothing and further additional postprocessing stages. This method shows good quality particularly with the noisy nature of the ultrasound datasets. Also, it is fast enough to work well with the real-time processing constraints of 4D ultrasound. Filtration of the surface is straightforward and does not need significant computation time. So, the new method can be implemented on datasets with speckle dominated texture like ultrasound imaging and still work well with higher quality datasets from CT and MRI. One of the advantages of the image space shading is the independent processing in the different stages, whereas the volume render is completely done on a stage and shading is done on a separate stage. This allows pipelining of these processing stages for higher performance. Moreover, this makes it possible to process both the volume rendered and surface rendered images concurrently.



(a) Rendering without surface shading



(b) Rendering with surface shading

Fig. 1. Comparison between rendering with surface shading and without surface shading.

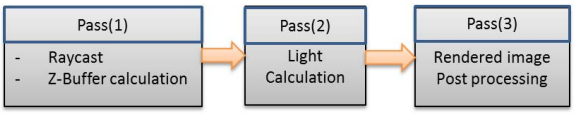


Fig. 2. Rendering Pipeline

II. RENDERING PIPELINE

A. *First Pass*

Two passes are used for volume rendering. Fig. 2 shows the rendering pipeline stages. The first Pass renders the volume using the raycasting technique. Each ray that travels through the volume presents a single pixel of the rendered 2D image, as shown in Fig. 3, while the ray is passing through the volume a surface detector function is applied, when the surface detector finds the surface point the distance between the surface point to the viewing plan is saved in the 4th components of each pixel vector as in Fig. 3. This stage is ended by 2D rendered image, each pixel presented by 4 components vector; the first 3 components presents the rendered pixels color and the 4th component presents the Z-Component.

B. *Second Pass*

This pass presents the lighting stage of the rendered image. This pass begins by filtering the rendered Z-Buffer components. Filtration here is intended only with the distance map component of each pixel vector. Then calculates the unit normal vector per pixel, Using three neighbors pixels for normal vector calculation, the (X, Y) components of each pixel are known by their location in the 2D image, the (Z) components is presented by the height map value.

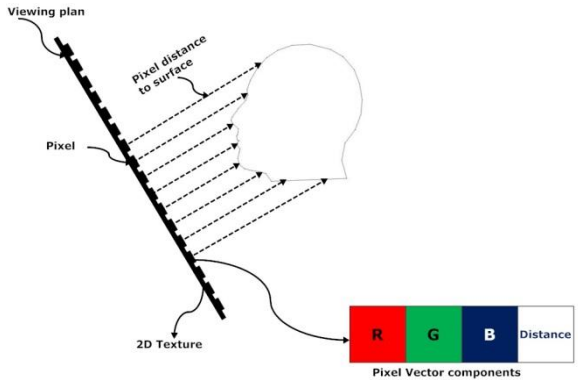


Fig. 3. the viewing plan with respect to the volume data set and distance map calculation. Lower right show the pixel vector components

For now each pixel has primary color (Pass 1 output), local Position(X, Y, Z) and the local normal vector. Use the local position and the normal vector to calculate the three light components based on Local Illumination Models By Bill-Phong [3] (diffusion, specular and ambient) (See Section IV for more details for more details). Then blend the light components with the primary color to illuminate the rendered image. It is clear that changing the position of the light source does not need to re-render pass one stage again, it's one of the main advantages of this method that the user can move the light source without the need to Raycast the volume again , also recalculate the light components does not need stress calculation as pass one stage. In Fig. 5, surface rendering is presented with light position at the front whereas Fig. 6 presents surface rendering with light position at the front-left.

Shadowing is another optional component, Ray tracing here is used to calculate the shadow component. For each pixel a ray travels from the current pixel to the light source. Fig. 4 shows shadow Ray tracing using height map. If the ray is obstructed with the height of another pixel, shadowed pixel is found.

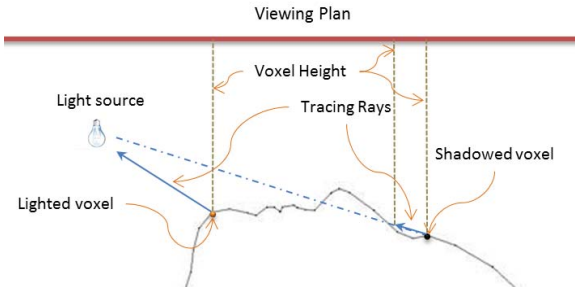


Fig. 4. Shadow Ray tracing using height map

C. Pass three

In this stage, a simple 2D filter is applied on the final image, we here apply 2×2 moving average filter. This filter is optional in the current implementation and the future plan for tis part is to use a more sophisticated filter for this stage such as nonlinear anisotropic diffusion.

D. Surface Detector

The Ultrasound dataset has a specific nature, especially in the fetus face display, the fetus face is preceded by amniotic fluid which is looks dark relative to the fetus face. While raycasting the volume, the samples are saved in a circular buffer. The length of this buffer is problem dependent. A buffer length of 8 was suitable for the current application. We apply the following formula,

$$Difference = \sum_{i=0}^{i=\frac{n}{2}-1} data(i) - \sum_{i=\frac{n}{2}}^{i=n-1} data(i). \quad (1)$$

Here, n is the buffer length. We compare the *Difference* value to a user-determined threshold to do the segmentation. In case of a surface point, we store the length of the ray at this point in the 4th element of the rendered pixel vector, this value will present the distance to the viewing plane (or the Z-Component).

It should be emphasized that the threshold value is a user controlled parameter, which enables distinguishing between different surfaces of the volume. Also taking the sign of the *Difference* into consideration can be used to detect different surfaces of the volume, here we intend only with the fetal surface that has *Difference* positive sign, as we go from black area to white area.

More enhancements can be done for filtering the sampled data. Instead of using the current sample to be saved in the circular buffer, we can use a 2D window perpendicular to the ray, taking the window average and using this averaged value as the current sample in the circular buffer. Using the 2D window average along with the filter that we already use in the *Difference* calculation has the effect of a 3D moving average filtration of the volume dataset.

III. RAYCASTING MODEL

3D Texture is used with a trilinear interpolation. The implementation and testing here is done on parallel projection raycasting, while the current implementation is also applicable on the perspective projection, but in this case when the surface point is found the distance to the viewing plan is calculated as a point to plane distance calculation. Performance Enhancement can be done based on the required rendering type, as if just need to display the surface points early ray termination is applicable.

IV. ILLUMINATION MODEL

The raycasting idea is to directly evaluate the volume-rendering integral along rays that are traversed from the viewing plan. For each pixel in the image, a single ray is cast into the volume. Then the volume data is resampled at discrete positions along the ray. Fig. 3 illustrates raycasting principle. By means of the transfer function, the scalar data values are mapped to optical properties that are the basis for accumulating light information along the ray. Typically, compositing can be performed in the same order as the ray traversal. Therefore, front to-Back compositing is applied as [2]:

$$C_{dst} = C_{dst} + (1 - \propto_{dst}) C_{src} . \quad (2)$$

$$\propto_{dst} = \propto_{dst} + (1 - \propto_{dst}) \propto_{src}. \quad (3)$$

A point light source is used, while a directional light source is applicable. Point light source enhances the contrast of the fetus face. Blinn-Phong model is used for the surface illumination. This model computes the light reflected by an object as a combination of three different terms, an ambient, a diffuse, and a specular term as: [3]

$$I_{Phong} = I_{ambient} + I_{diffuse} + I_{speculer}. \quad (4)$$

The shadowing illumination components are first calculated by testing each surface pixel if it is shadowed or not. If a shadowed pixel is found, reduction of the pixel value by the shadowing coefficient constant is done.

Fig. 5. Surface rendering with light position at front

V. SURFACE FILTRATION

Surface filtration takes place in pass two as in "Section II.B"; we have used a simple linear filter kernel based

on moving average filter, it's well known that the linear filter blur the image but in our case the fetus face does not has high frequency components and so the image quality is still reasonable after the maximum kernel size is used.

Filtration is done on the Z-Buffer component. It works as a sand paper smoothing a surface. Fig. 7 presents surface rendering with low filtration with surface filtration using a 7×7 kernel. On the other hand, Fig. 8 shows surface rendering with medium filtration using a 9×9 kernel. Even with the noisy nature of ultrasound data, the light quality was reasonable. Fig. 8 shows surface rendering with medium filtration. The main advantage of this method is that the filtration is done on a 2D texture dataset, which makes the memory fetching performance better than working on a 3D texture.

Fig. 6. Surface rendering with light position at front-left

Fig. 7. Surface rendering with low filtration

Fig. 8. Surface rendering with medium filtration

VI. COORDINATE SYSTEM

2D Ultrasound images are composed of about 128 ultrasound scan line which form the ultrasound image, the lines are sampled on a polar grid which is the case of convex probe, most 4D ultrasound probes are formed from 2D convex probe, in our case the ultrasound scan lines transformation from polar coordinate to Cartesian coordinate takes place in the ultrasound hardware, so the input for our system is a 2D images aligned on a polar grid. The polar grid presents the fan angle of the 4D probe.

Scan conversion is done during fetching the volume data using a simple look-up table; a 2D texture memory is used for the look-up table in the GPU to transform from polar coordinate to Cartesian coordinate. Because of the 2D image comes from the ultrasound system is in the Cartesian coordinate format, the transformation is done on the fanning direction only. The look-up table is calculated in the CPU and sent to the GPU just after any user parameters change.

VII. PERFORMANCE ISSUES

As higher GPU technology is reached, advanced memory textures and throughput rate are expected to grow significantly. Here, we used a mid-range graphics card to confirm that the current implementation does not significantly affect the raycasting performance. The two major specifications of the graphics card that affect performance are the core and memory specifications. For the core, the number of cores used and the texture filling rate are the most dominant feature. On the other hand, the performance of the memory part is determined by its bandwidth.

In the first pass, all the overheads added to the regular raycasting are just storing the most recent sample in a circular buffer, one addition, one subtraction, and one comparison. Because the new method does not need more fetching the 3D dataset memory, there is no significant decrease of the timing performance. In the second pass, the input to this stage is a 2D texture, which has better fetching time performance than the 3D texture used in the first pass. So, without shadowing calculations, this stage

does not significantly affect the rendering time performance.

Shadowing noticeably affects the performance due to the ray-tracing needed. However, it is still achievable on the used graphics card because it is still done on just a 2D texture dataset.

The time to form ultrasound data sets assume 35 frames per volume and a depth of 16 cm can be computed from the theory to be around 466 ms. This time is reduced according to the number of ultrasound frames per volume, which is directly mapped to the quality of the rendered volume. The rendering time for 2D ultrasound frame resolution of 512×512 with maximum filter kernel size, the rendering time was measured to be about 375 ms. For the same volume size with minimum filter kernel size, the rendering time was about 38 ms. Hence, the performance offered by this midrange graphics card for high smoothing requirements was barely acceptable while for lower smoothing it offered a very high speed of rendering that is much less than the timing constrains of the ultrasound.

VIII. Conclusions and Future Work

Surface rendering of ultrasound dataset using the image space z-buffer shading was developed based a commercial GPU medium-range graphics card. The current implementation met the time constraints of the 4D ultrasound and overcomes the speckled nature of the ultrasound dataset. The timing performance encourages further the enhancement of this method and in particular the surface filtration stage given the available time and processing power. Further work is needed to better address the issue of surface detection. A simple edge detector was used in this work just to demonstrate the idea but more elaborate filters can be explored with potential to enhance surface detection.

REFERENCES

[1] Marc Levoy. “Display of surfaces from volume data,” IEEE Computer Graphics and Applications.” vol. 8, no. 3, pp. 29-37, 1988.

[2] Klaus Engel, “Real-Time Volume Graphics.” ,2006, ISBN 1-56881-266-3

[3] J. F. Blinn. “Models of Light Reflection for Computer Synthesized Pictures.” Computer Graphics, vol. 11, no. 2, pp. 192-198, 1977.

[4] J. F. Blinn. “Compositing, Part I: Theory” IEEE Computer Graphics and Application, vol. 14, no. 5, 1994.

[5] R. A. Reynolds. “Image space shading of 3-dimentional objects.” Computer Vision, Graphics, And Image Processing, 29, pp. 361-376, 1985.

[6] Dreblin, R.A. CARPENTER and Hanrahan. “Volume rendering, Computer Graphics.” vol. 22. no. 4, pp. 51-58, August, 1988

[7] Y.Zhang, R.Rohling, D.K.Pai. “Direct Surface Extraction from 3D Freehand Ulrasound Images.”, IEEE Visualization 2002, Oct. 2002.

[8] J.K.Udupa, G.T.Herman, 3D Imaging in Medicine second edition , CRC Press, 2000.

[9] Sakas G, Walter S. “Extracting surfaces from fuzzy 3dultrasound data.” In: Cook RL, editors. ACM SIGGRAPH: Siggraph ’95, Conference Proceedings. New York, Reading MA: ACM Press, Addison-Wesley, pp. 465,474, 1995.

[10] R.Fattal,_ D. Lischinski, Variational Classification for Visualization of 3D Ultrasound Data, In: IEEE Visualization Computer Society 2001.

[11] T. R. Nelson and T. T. Elvins. Visualization of 3D ultrasound data. IEEE Computer Graphics and Applications, vol. 13 no. 6,pp 50–57, Nov. 1993.

[12] J.Kniss, S.P.ze, C.Hansen, P.Shirley, A.McPherson, A Model for Volume Lighting and Modeling, IEEE transactions on visualization and computer graphics, vol. 9, no. 2,pp. 192-162, April-June 2003.

[13] W. Lorenson and H. Cline. Marching cubes: a high resolution 3D surface construction algorithm. Computer Graphics, vol. 21 no. 4, pp. 163–169, 1982.

[14] Hoehne, K.H. and Bernstein, R., ‘‘Shading 3D-Images from CT Using Gray-Level Gradients,’’IEEE Transactions on Medical Imaging, vol. 5, no. 1, pp. 45-47, March, 1986.