Journal of Medical Imaging and Health Informatics
Vol. 4, 1–6, 2014

Optimized Graphical Processing Unit Processing Framework for Surface Rendering in 4D Ultrasound Imaging

Ahmed F. Elnokrashy[1,2], Marwan Hassan[1], Tamer Hosny[2], Ahmed Ali[2], Alaa Megawer[1], Amr M. Hendy[1,2], and Yasser M. Kadah[1,*]

[1]*Biomedical Engineering Department, Cairo University, Giza, Egypt*
[2]*IBE Tech, Giza, Egypt*

Four-dimensional (4D) ultrasound imaging extends the real-time capability of ultrasound to visualize a real-time volume that can be manipulated by the sonographer. Among the different visualization methods, surface rendering is a common mode for displaying volumetric datasets such as in obstetrical applications. A challenge in this mode is that surface shading is required to visualize the surface and enhances the surface contrast and this has very demanding computational requirements for 3D surfaces. Here, we present an optimized high-performance rendering pipeline based on four stages for preprocessing, volume rendering, surface shading, and postprocessing. The new approach is implemented to render volumes acquired on a 4D commercial ultrasound imaging system to illustrate its practicality. The results demonstrate diagnostic quality of rendered volumes at a computational time cost that is suitable for 4D real-time processing. Given its low cost of required hardware, the new pipeline has potential for making 4D imaging systems more affordable while maintaining diagnostic quality and performance.

Keywords: Surface Rendering, 4D Ultrasound Imaging, Ray-Casting, Visualization.

1. INTRODUCTION

Volume rendering in medical imaging offers a display of a volumetric dataset after it is projected onto a plane using ray-casting.[1] Many algorithms to extract surfaces of different structures inside the body have been developed for imaging modalities with sufficient resolution and signal-to-noise ratio (SNR) such as magnetic resonance imaging and computed tomography.[2] However, surface extraction in ultrasound imaging applications is a much tougher problem given the lower resolution and SNR in addition to the presence of speckle as a characteristic texture in ultrasound images. Moreover, for volume scanning applications, reflected waves from specular reflectors vary with the direction of scanning unlike scattering. If we add to all that the real-time nature of ultrasound imaging that require such rendering to be done on the fly rather than offline as with other modalities, we realize how challenging this problem is.

Four-dimensional (4D) ultrasound surface rendering is a challenge for many reasons including noisy dataset, nonuniform volume sampling and processing time constraints. Two-dimensional (2D) cross-sectional images of three-dimensional (3D) objects require realistic shading to create the illusion of depth.[3] Hence, the 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 utilized programming platforms for the implementation include OpenGL, which draws the volume cube and shading language that perform ray-casting of the volume and coordinate transformation in addition to 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.[4] Another method is based on gradient calculation where the gradient vector is calculated first then used to represent the normal vector to be used for surface shading calculations.[5]

*Author to whom correspondence should be addressed.

Surface shading is an excellent method for allowing the operator to visualize the 3D shape of a structure.[6] The method is intuitive since humans perceive objects as solid structures that can be viewed from different perspectives to form an impression of their shape. The surface shading method aims to simulate this process in the digital world[7] and serves as a common tool for visualization in many imaging modalities. In ultrasound, the low SNR and parallel tissue boundary discontinuities make defining smooth surfaces difficult. Smoothing of a surface can be performed at the rendering stage. An approach was presented to render the fetal surface from the ultrasound 3D dataset.[8] First, the 3D dataset is smoothed using median and Gaussian filters. Then, a surface primitive is extracted and used for surface shading. Another approach renders smooth surfaces from 3D ultrasound based on the oriented splatting.[9] In this case, the surface is extracted based on the variational classification principle. Fuzzy surface rendering is done by oriented splatting whereby it creates triangles aligned with the gradient of the surface function. Such triangles are then colored with a Gaussian function and rendered in a back-to-front order. Another group proposed an improved surface rendering technique for 3D ultrasound data of fetuses[10] where modified anisotropic diffusion filtering is first applied to the dataset. In spite of the success of such methods for their respective applications, such approaches were not intended for real-time processing in addition to not being optimized for implementation on the presently available GPU technology.[11] Several groups proposed GPU-based framework for the ultrasound data volume render[11–14] with main focus on processing time. An approach that would combine the optimization methods used in offline 3D visualization and real-time processing on GPUs would offer an excellent platform for 4D imaging applications.

In this work, we present an optimized framework for surface rendering of the 4D ultrasound volumetric data sets in real-time. This approach offers real-time performance while maintaining high quality of surface rendering that is close to what is offered by offline 3D rendering methods. Image space z-buffer shading is used,[3] which postpones the surface shading to the last stage of the surface rendering. An additional task is added to the ray-casting routine for edge detection, which can be implemented to get good results with modest computational effort. Shading based on a 2D texture offers optimized performance as compared to 3D texture and allows further enhancement of the surface and further post-processing while meeting real-time requirements. Moreover, such image space shading offers independent volume rendering and shading processing in the different stages. This allows pipelining for higher performance and makes it possible to process both the volume rendered and surface rendered images concurrently. The new approach is implemented and verified on a customized processing based on off-the-shelf components this ensuring low-cost of required hardware.

2. ACQUISITION AND RENDERING HARDWARE DESCRIPTION

The hardware setup used for the surface rendering is illustrated in Figure 1. Imaging was performed using the DIGISON-Q ultrasound scanner. This scanner has independent 32-channel digital transmitters and receivers. The processing in the receive system filters and detects the digital RF lines which are subsequently converted and sent via fast PCIe interface to the digital processing computer. The processing platform is custom-built from

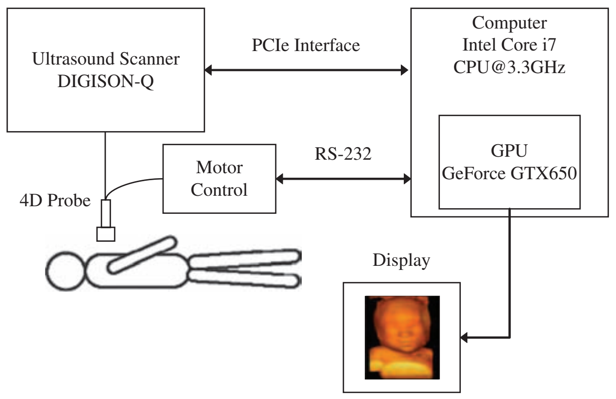

Fig. 1. Block diagram of the system data acquisition hardware.

off-the-shelf hardware components comprising Intel® Core™ i3 processor with 4 cores running at 3.3 GHz each with 4 GB of DDR3-1600 RAM, a GA-B75M-D3H motherboard (GIGABYTE Technology Co., LTD) with PCI-E Gen3 expansion slots, and a NVidia GeForce GTX650 video card (NVIDIA Corporation, Santa Clara, USA) with 1027 MB of video RAM and 384 graphical processing unit (GPU) cores. The computer components were chosen to have sufficiently fast data transfer rates for real-time data acquisition, processing, and graphical display. The data are acquired directly from the ultrasound scanner through a single lane PCIe Gen3 with a maximum bandwidth of 1 GB/s per lane. The total cost of the hardware of the processing platform was less than $1500, which is a very modest figure for such demanding application.

The rendering is performed using the multicore GPUs on the programmable video card in real-time. The data acquisition, transfer, reconstruction and rendering processes are implemented in different layers with software developed using Microsoft Visual Studio 2010, the Microsoft DDK, Intel Parallel Studio, The Intel Math Kernel Library, OpenGL[15] and NVidia CG language.[16] Each detected ultrasound line of the imaging system is low-pass filtered then undersampled to 512 samples per line with quantization of 4 Bytes/sample. The ultrasound data from the acquisition hardware are copied directly to the computer memory as raw scan line data (image sticks) and then arranged in an ordered polar (θ, r) grid prior to transfer to the GPUs for further rendering processing.

3. RENDERING METHODOLOGY

As the Frames are sent to the GPUs on the graphics card and before being integrated into the 3D texture memory, 2D image processing takes place in Pass (1). In Pass (1) after coordinate conversion, a homogenous filter is applied to smooth and slightly blur the scan-converted 2D image obtained from interpolating the sector radial scan lines into a uniform grid. The output from this stage is integrated directly into the 3D texture. In Pass (2), processing starts only after complete reception of all volume frames. Ray-casting and Z-buffer calculation are applied in this stage. The Z-buffer is filtered using homogenous filter in the subsequent Pass (3) then used to shade the rendered 2D image. Finally, Pass (4) further applies a homogenous filter on the final image before display. The block diagram of the proposed processing methodology is shown in Figure 2.

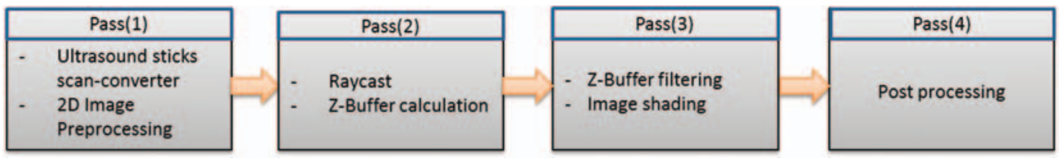

Fig. 2. Rendering pipeline.

3.1. Pass (1): Preprocessing

As the 4D ultrasound probe wobbles, ultrasound detected lines are sent to the GPUs synchronously. In this stage, we include a preprocessing step before integrating the ultrasound data to the 3D texture. The fragment shader uses a look-up table (LUT) to convert the image sticks from a polar (r, Θ) to a Cartesian (x, y) coordinate system for each image. Moreover, a homogenous filter is applied to the raw data. One of the major performance issues of this stage is skipping the empty spaces that result from the coordinate transformation process. In this work, we use a third component flag in the LUT to skip the empty spaces in the scan-converted image.

3.2. Pass (2): Volume Rendering

This processing stage takes care of transforming the volume data from polar to Cartesian coordinate in the probe motion direction also using a LUT-based method (since the in-plane direction transformation is done in Pass (1)). Then, volume ray-casting is performed to the transformed data where 3D Texture is used with a trilinear interpolation. The implementation here uses a parallel projection ray-casting model without loss of generality (current implementation is also applicable on the perspective projection with a slightly different point to plane distance calculation). Figure 3 illustrates ray-casting geometry. Alpha-blending comprising the combination of pixel values to allow for transparency is subsequently applied to the resultant volume[1] followed by a 3D smoothing Gaussian filter. This 3D filter kernel is calculated in the CPU and sent to the fragment program as a float array. By pipelining the samples, there is no need to read the entire 3D rendered volume samples but rather it is possible to read the front slice of samples and discards the oldest slice and so on to optimize the data transfer.[13]

As the last step, edge detection and z-buffer formation are performed. Such steps take advantage of the special form of the

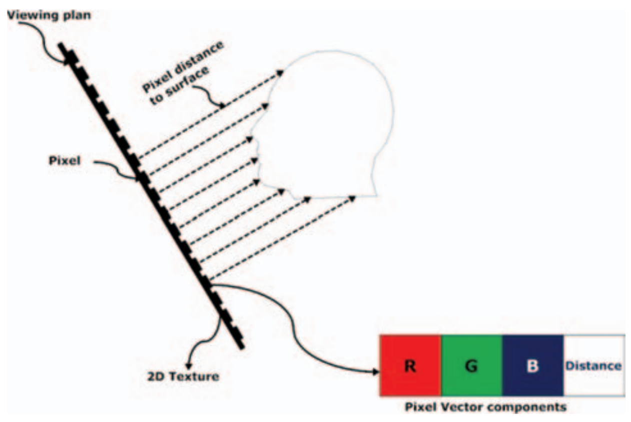

Fig. 3. Viewing plan with respect to the volume data set and distance map calculation. Lower right show the pixel vector components.

ultrasound imaging data especially in such cases when imaging the fetus face. Since the fetus face is preceded by amniotic fluid that produces very little reflected ultrasound signal compared to the baby face that acts close to a specular reflector, an excellent contrast between them is obtained. While ray-casting the volume, the data samples are saved in a circular buffer (a data structure that is connected from both ends with a position index to simplify the implementation of a first-in first out queue) with a reasonable length (A buffer length of as small as 8 was sufficient to maintain stable performance in the experimental verification of this work). We compute the following measure,

$$\text{Difference} = \sum_{i=0}^{i=(n/2)-1} \text{data}(i) - \sum_{i=n/2}^{i=n-1} \text{data}(i) \qquad (1)$$

Here, n is the buffer length. We compare the *Difference* value to a user-selectable 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 that he/she can change from the user-interface to enable distinguishing different surfaces of the volume based on this selection. Also taking the sign of the *Difference* into consideration can be used to detect different surfaces of the volume such as detecting the surface of fetal face versus detecting the surface of a fluid-filled structure such as the uterus.

3.3. Pass (3): Surface Shading

This stage represents the lighting stage of the rendered volume. This stage begins by filtering the rendered Z-buffer components using a homogenous filter. Then, the 2D rendered image is appropriately shaded using image space shading[3] and local illumination models[17] as follows. The ray-casting model 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. 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.[18]

A point light source is used, while a directional light source is still applicable. Point light source enhances the contrast of the fetus face. The Blinn–Phong model is used for the surface illumination where it computes the light reflected by an object as a combination of three different terms, ambient, diffuse, and specular intensity terms in the form:[17]

$$I_{\text{Phong}} = I_{\text{ambient}} + I_{\text{diffuse}} + I_{\text{specular}} \qquad (2)$$

The shadowing illumination components are first calculated by testing each surface pixel if it is shadowed or not using ray-tracing back from each pixel to the light source and if there is

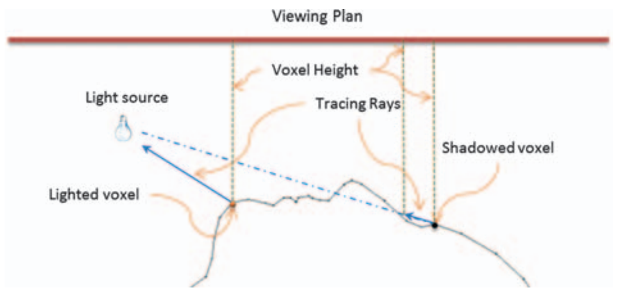

Fig. 4. Shadow ray tracing using height map.

no other pixel obstructs the ray this pixel considered as lighted pixel and if not it's a shadowed pixel. If a shadowed pixel is found, reduction of the pixel value by the shadowing coefficient constant is done. The geometry of the process using the map of voxel heights from the viewing plane is illustrated in Figure 4. It should be noted that shadowing noticeably affects the overall performance due to the ray-tracing needed. However, it is still achievable in our system using a modest graphics card because it is still done on 2D texture data rather than 3D texture data.

3.4. Pass (4): Postprocessing

To enhance the output image, an edge-preserving speckle reduction filter is applied on the final shaded image.[19–21] For simplicity, the homogenous filter was selected for use with this implementation of the new system.[19] Filtration is done on the Z-Buffer component and works as a sand paper smoothing of the rendered a surface. Surface rendering with different settings with low filtration using a 5×5 kernel or high filtration using a 9×9 kernel can be selected by the user. The main advantage of this approach is that the filtration is performed on a 2D texture dataset, which makes the memory fetching performance better than working on a 3D texture.

4. RESULTS AND DISCUSSION

The time to collect an ultrasound volume assuming 35 frames per volume and a depth of 16 cm can be calculated from the echo-ranging theory to be around 466 ms (since the time to receive one line from a depth d is equal to $2d/c$ where c is the speed of ultrasound and the total time is this time multiplied by number of lines per image and number of images in the scanned volume).[22] This time becomes shorter when the number of ultrasound frames per volume becomes less or the depth of scanning is decreased. Hence, this time maps directly to the quality of the rendered volume. Figure 5 shows sample 4D rendering results using data collected by scanning a 3D ultrasound training phantom (CIRS, Inc. Model 068) with different light positions showing light source from the front and front–left directions. Also, in Figure 6 presents the results of rendering with (a) low filtration, and (b) high filtration kernel sizes. The results show good diagnostic quality of rendering which was performed in real-time.

To obtain quantitative results, a number of experiments were performed to compute the rendering time variation with the different imaging parameters to assess the practicality of the developed system. The results of the variation of the rendering time with number of frames within the rendered volume are shown in Table I (for surface rendering mode, 3D Kernel size of $3\times3\times3$, 2D homogeneous filter kernel size of 9×9, image sector size of 77°, 3D scanning Fan angle of 61° and volume size of $512\times512\times512$). It is clear from the results that such variations do not have a high order of complexity as suggested by the small differences in time for the whole range of values considered. Table II presents the results of the variation of the rendering time with size of rendered volume (for surface rendering mode, 3D Kernel size of $3\times3\times3$, 2D homogeneous filter kernel size of 9×9, image sector size of 77°, 3D scanning Fan angle of 61° and 37 frames per volume). In this case, the variation show a linear O(n) variation with size of data in each dimension. Moreover, the study of the variation of the rendering time with size of 3D filter kernel is presented in Table III (for surface rendering mode, 2D homogeneous filter kernel size of 9×9, image sector size of 77°, 3D scanning Fan angle of 61°, 37 frames per volume and volume size of $512\times512\times512$) and shows a linear variation with number of points in the 3D filter kernel as expected. On the other hand,

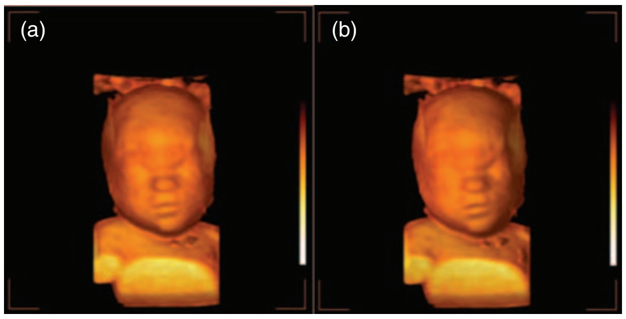

Fig. 5. Rendering with light position from the front (a) and front-left (b).

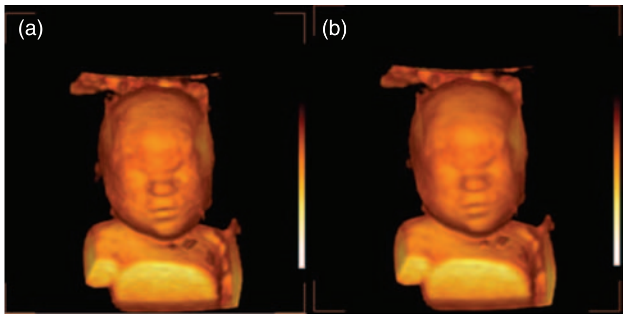

Fig. 6. Rendering with low filtration (a) and high filtration (b).

Table I. Rendering time variation with number of frames per volume.

Frames per volume	Rendering time (ms)
27	53.7
31	54.0
35	54.6
37	54.6

Table II. Rendering Time variation with 3D filter kernel size.

Volume data size	Rendering time (ms)
$512 \times 512 \times 512$	54.6
$384 \times 384 \times 384$	35.0
$256 \times 256 \times 256$	15.8
$128 \times 128 \times 128$	13.2

Table III. Rendering Time variation with 3D filter kernel size.

3D Filter kernel size	Rendering time (ms)
No filtering	21.5
$3\times3\times3$	54.6
$5\times5\times5$	187

Table IV. Rendering time variation with 2D homogeneous filter size.

Homogenous filter	Render time (ms)
3×3	54.3
5×5	54.9
7×7	55.5
11×11	56.8

the variation of the rendering time with size of 2D homogeneous filter kernel presented in Table IV suggest very small variation of rendering time with 2D homogeneous kernel size (for surface rendering mode, 3D filter kernel size of $3\times3\times3$, image sector size of 77°, 3D scanning Fan angle of 61°, 37 frames per volume and volume size of $512\times512\times512$). As a result, it is clear that the key issue in the rendering time is the choice of 3D filtration kernel size. Also, the results demonstrate that the performance offered by this midrange graphics card for different practical settings is acceptable and fulfill the timing constraints of real-time rendering. In other words, the frame rate in the developed system is limited only by the data collection speed not the reconstruction and visualization.

In our system, the 4D ultrasound probe assumes the geometry of a convex 2D probe making a convex motion trajectory in the cross-plane direction. In any given probe position along this trajectory, the acquired 2D ultrasound image is composed of ultrasound scan lines or sticks sampled on a polar grid. Hence, the system has to do two polar-to-Cartesian transformations for in-plane data to construct individual 2D images in the volume, then for such collection of images to form the 3D volume. The amount of required data acquisition and processing depend on both the fan angles of the 2D images and the fan angle of the probe motion and the selection of these parameters by the user affects the frame rate of the system that is mainly limited by the acquisition part physics. As shown by the results, the processing and display part is significantly faster in our implementation and therefore does not impose any limitations on this combined process.

Scan conversion is done during fetching the volume data using a simple LUT, a 2D texture memory is used for the LUT on the GPU side to do the job. The LUT data are calculated and/or updated in the CPU based on the current parameters of the data acquisition and then sent to the GPU side. Performance enhancement in the ray-casting step in Pass (2) is achieved in this work with early ray termination is done whenever the surface point is detected depending on the type of rendering required (rendering baby face is an example where this can be applied). In general, three criteria are used here for determining ray termination to optimize the performance. The first is based on leaving the bounding box designating the volume of interest selected by the user. The second is based on the high accumulated opacity value.[23] Finally, the third criterion is based on the scan-converter LUT detecting a ray going out from valid volume data cone to empty space with no data.

Further enhancement can be achieved in the edge detection step of 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 in Eq. (1) has the effect of a 3D moving average filtration of the volume dataset.

As higher performance GPU technology becomes available, 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 ray-casting 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 important feature. On the other hand, the performance of the memory part is determined by its bandwidth.

In Pass (2), 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 Pass (3), the input to this stage is a 2D texture, which has better fetching time performance than the 3D texture used in the second pass. So, without shadowing calculations, this stage does not significantly affect the rendering time performance.

5. CONCLUSIONS AND FUTURE WORK

In this work, we presented a high-performance rendering pipeline implemented on a customized platform made up of low-cost commercial components. This approach combines the optimization methods used in offline 3D visualization and real-time processing on GPUs to offer an optimized platform for 4D imaging applications. The processing system is verified by rendering volumes acquired on a 4D commercial ultrasound imaging system with a research interface with results showing diagnostic quality while maintaining real-time performance suggesting potential for practical utility.

Acknowledgments: The authors would like to acknowledge the partial financial support from a PDP grant funded by ITIDA/ITAC program.

References and Notes

1. M. Levoy, Display of surfaces from volume data. *IEEE Computer Graphics and Applications* 8, 29 **(1988)**.
2. R. A. Dreblin, L. Carpenter, and P. Hanrahan, *Volume Rendering, Computer Graphics* 22, 51 **(1988)**.
3. D. Gordon and R. A. Reynolds, Image space shading of 3-dimentional objects. *Computer Vision, Graphics, and Image Processing* 29, 361 **(1985)**.
4. W. Lorenson and H. Cline, Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics* 21, 163 **(1982)**.
5. K. H. Hoehne and R. Bernstein, Shading 3D-images from CT using gray-level gradients. *IEEE Trans. Medical Imaging* 5, 45 **(1986)**.
6. J. K. Udupa and G. T. Herman, 3D Imaging in Medicine, 2nd edn., CRC Press, Boca Raton **(2000)**.
7. K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, Real-Time Volume Graphics, edited by A. K. Peters, Ltd., Wellesley, MA **(2006)**.
8. T. R. Nelson and T. T. Elvins, Visualization of 3D ultrasound data. *IEEE Computer Graphics and Applications* 13, 50 **(1993)**.
9. R. Fattal and D. Lischinski, Variational classification for visualization of 3D ultrasound data, *Proc. IEEE Visualization 2001* (*Vis'01*), San Diego, U.S.A. **(2001)**, Vols. 403–410.

10. Y. Zhang, R. Rohling, and D. K. Pai, Direct surface extraction from 3D Freehand ultrasound images, *Proc. IEEE Visualization 2002*, Boston, U.S.A. **(2002)**, Vols. 45–52.

11. S. Lim, K. Kwon, and B. S. Shin, GPU-based interactive visualization framework for ultrasound datasets. *Computer Animation and Virtual Worlds* 20, 11 **(2009)**.

12. A. F. Elnokrashy, A. A. Elmalky, T. M. Hosny, M. A. Ellah, A. Megawer, A. Elsebai, A.-B. M. Youssef, and Y. M. Kadah, GPU-based reconstruction and display for 4D ultrasound data, *Proc. 2009 IEEE International Ultrasonics Symposium* (*IUS*), Rome, Italy **(2009)**, Vols. 189–192.

13. A. F. Elnokrashy, M. Hassan, T. Hosny, A. Ali, A. Megawer, and Y. M. Kadah, Multipass GPU surface rendering in 4D ultrasound, *Proc. Cairo International Biomedical Engineering Conference 2012* (*CIBEC'2012*), Cairo, Egypt **(2012)**, Vols. 39–43.

14. A. Birkeland, V. Solteszova, D. Hönigmann, O. H. Gilja, S. Brekke, T. Ropinski, and I. Viola, The ultrasound visualization pipeline-a survey, arXiv preprint arXiv:1206.3975 **(2012)**.

15. D. Shreiner, OpenGL Reference Manual: The Official Reference to OpenGL, Version 1.4, 4th edn., Addison-Wesley Professional, Reading, Massachusetts **(2004)**.

16. NVIDIA Corporation, Cg 3.1 Reference Manual Release 3.1, 2012. Available at: http://developer.download.nvidia.com/cg/Cg_3.1/Cg-3.1_April2012_ReferenceManual.pdf.

17. J. F. Blinn, Models of light reflection for computer synthesized pictures. *Computer Graphics* 11, 192 **(1977)**.

18. J. F. Blinn, Compositing, Part I: Theory. *IEEE Computer Graphics and Application* 14, 83 **(1994)**.

19. C. P. Loizou and C. S. Pattichis, Despeckle Filtering Algorithms and Software for Ultrasound Imaging, Morgan and Claypool Publishers, CA **(2008)**.

20. Z. A. Mustafa, B. A. Abrahim, I. A. Yassine, N. Zayed, and Y. M. Kadah, Wavelet domain bilateral filtering with subband mixing for magnetic resonance image enhancement. *J. Med. Imaging Health Inf.* 2, 230 **(2012)**.

21. B. A. Abrahim, Z. A. Mustafa, I. A. Yassine, N. Zayed, and Y. M. Kadah, Hybrid total variation and wavelet thresholding speckle reduction for medical ultrasound imaging. *J. Med. Imaging Health Inf.* 2, 114 **(2012)**.

22. P. R. Hoskins, K. Martin, and A. Thrush, Diagnostic ultrasound: Physics and Equipment, Cambridge University Press, Cambridge **(2010)**.

23. W. Li, K. Mueller, and A. Kaufman, Empty space skipping and occlusion clipping for texture-based volume rendering. *Proc. 14th IEEE Visualization 2003* (*VIS'03*), Seattle, U.S.A. **(2003)**, Vol. 42.

Received: 27 August 2013. Accepted: 8 September 2013.