International Journals of Advanced Research in Computer Science and Software Engineering
ISSN: 2277-128X (Volume-8, Issue-10)

Research Article | **October 2018**

4D Dynamic Volume Rendering using Raycasting

A V Krishna Rao Padyala, Srinivasa Rao Namburi
Assistant Professor, Department of IT, Bapatla Engineering College, Andra Pradesh, India
guntur.krishna@gmail.com, srihimasri@gmail.com

Abstract- Medical imaging has become one of the most used diagnostic tools in the medical profession in the last three decades. Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) technologies have become widely adopted because of their ability to capture the human body in a non-invasive manner. A volumetric dataset is a series of orthogonal 2D slices captured at a regular interval, typically along the axis of the body from the head to the feet. Volume rendering is a computer graphics technique that allows volumetric data to be visualized and manipulated as a single 3D object. Some of the volume rendering methods are Isosurface rendering, image splatting , shear warp, texture slicing, and raycasting.

CT and MRI hardware was limited to providing a single 3D scan of the human body. Functional imaging let capture of anatomical data over time.One of them is Functional MRI (fMRI), is used to capture changes in the human body over time.This paper presents creation of generic software capable of performing real-time 4D volume rendering via raycasting, on desktop.

Keywords- ray casting, body visualization, mobile ray casting

I. INTRODUCTION

The idea of seeing inside of the human body without surgery was a paradigm shift that has since spawned the creation of similar imaging techniques such as Magnetic Resonance Imaging (MRI), Computed Tomography (CT), Positron Emission Tomography (PET), and Ultrasound to name a few of the more commonly used methods. Today's CT and MRI scanners are capable of generating 2D slices of the order of 512 x 512, 1024 x 1024 pixels, and sometimes higher. The higher the number of pixels, the more detailed information can be obtained. While 1024 x 1024 resolution is not considered high relative to the current TVs, computer monitors, and cell phones, it is high for medical imaging because CT and MRI scanners are limited by physics.

Computer Tomography is a medical imaging technology that uses ionizing radiation from an x-ray tube to generate cross-sectional slices of the patient's body [1].An x-ray generating tube is placed to one side of the patient and x-rays are emitted in the direction of the patient with an x-ray sensor on the opposite side of the tube. This process generates a single planer image. The tube and sensor are then rotated slightly around an axis relative to the patient and take another planer image. This process is continued until a complete scan of the patient is performed.

Magnetic Resonance Imaging (MRI) is a technology built upon the use of magnetic fields and radio frequency pulses. MRIs work by placing the scanned object, often a person, into a powerful unidirectional magnetic field. The purpose of the magnetic field is to align nuclei in the object, typically hydrogen. Nuclei with an odd number of protons or neutrons exhibit the quantum-mechanical phenomenon known as Nuclear Magnetic Resonance (NMR), or a spin causing a weak magnetic field. A specific radio frequency (RF), known as the Larmor frequency, is determined by the magnetic field strength and is able to resonate the atoms and knock them out of alignment [3]. When this RF signal is removed, the atoms oscillate back into alignment. The time taken for the atoms to oscillate back into alignment is measured to determine properties like tissue density. This process would provide one sample for the entire object. To obtain samples at 3D points in the space of the object, gradient magnetic fields are used to change the magnetic field strength along the object and thus change the Larmor frequency required to resonate the atoms along the object [6,7].

II. VOLUMETRIC DATA AND VOLUME RENDERING

While CT and MRI scanners can obtain slices from any angle or direction, they are typically obtained by scanning along the axis of the body, from head to feet or vice versa, as seen in Figure 1. For CT scanning, the field is an x-ray and for MRI the field is a magnetic field. These 2D slices can then be combined to create a single 3D block of data representing the entire scan of the patient, as shown in Figure 1, known as volumetric data. This difference between static and functional imaging created a need for new ways to store the data. The Digital Imaging and Communications in

Medicine(DICOM) format was designed for storing static data like X-rays and CT scans, but it is not easily extended to 4D time-varying data. Instead, new data formats, such as Neuroimaging Informatics Technology Initiative(NIfTI).

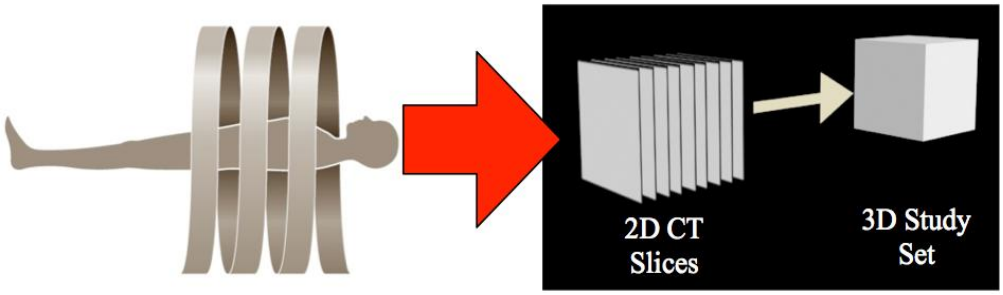

Figure 1: CT scanner showing the scanned body path to a 3D volume.

Volume rendering is the process of taking volumetric data and visualizing it. surface rendering that represents an object as a series of vertices making up an outer shell to show the object's shape, volume rendering sees an object as a three dimensional lattice of vertices. Surface rendering does not contain values inside the object shell, whereas volume rendering contains values at vertices throughout the mass of data being represented. The main classes of volume rendering techniques are Iso-surface Surface Rendering, Image Splatting, Shear Warp, Texture Slicing, and Raycasting.

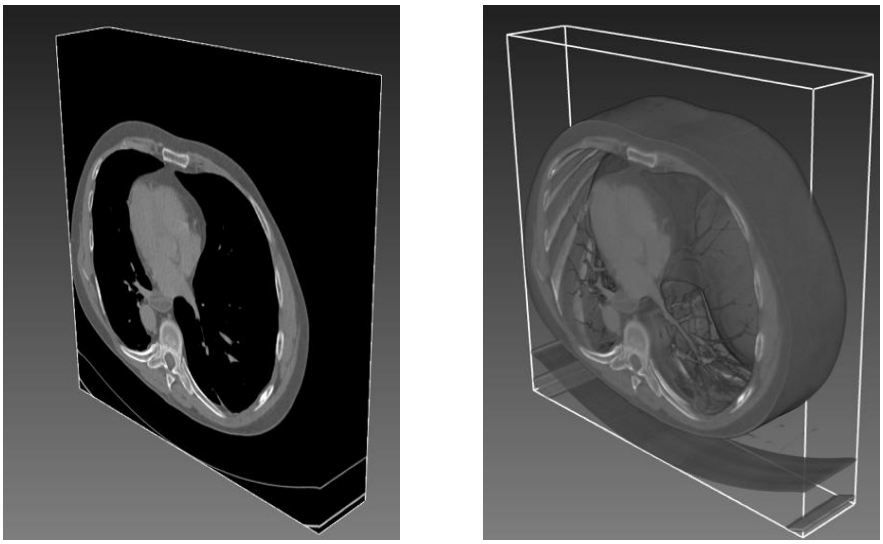

Figure 2: Visualizing the surface data (left) versus volume rendering to see the internal data (right).

III. RAYCASTING AND VOLUME PIPELINE

Raycasting is referred to as a direct volume rendering technique [4]. It involves casting rays from each pixel in the frame buffer through the volume in the view direction. All points along the rays' path that intersect with the volume are sampled and composited to generate the final pixel color. When volume rendering on the central processing unit (CPU), it is common to divide the screen into sections to try and parallelize the process. Each section is given to one of the CPU cores to render. The sections rendered by each core are then combined together to get the final image to show [5][6][7]. This provides a faster render than a single threaded application. Raycasting is a parallel method by nature with all rays executing independently of each other. This makes raycasting an ideal method for use on massive parallel architectures like graphics processing units (GPUs). There has been a significant movement in parallel computing to use commodity GPUs containing upwards of 3,000 computational cores.

Volume rendering begins with acquiring a volumetric dataset. volumetric data is all comprised of a set of samples, known as voxels, in the three dimensions (i.e. x, y, and z). Each point contains a measured value, v. The measured value of the voxel can vary widely depending on the application. In fMRI brain scans, the voxel value is a one-dimensional value representing the blood oxygenation level-dependent (BOLD) signal change [8]. This is used because neural activity has been linked with local changes in brain oxygen content[9]. The change of oxygen levels over time also requires the addition of a time component, t, to the volume data sample, resulting in a (x,y,z,t,v) for each sample.

The volumetric data is then run through the volume-rendering pipeline. The actual pipeline itself might differ depending on the application or the type of data being used, but the basic pipeline is the same for all volume renderers. Figure 3 shows the basic volume-rendering pipeline. Not all steps of this pipeline are used in all cases, nor are they always performed in this specific order.

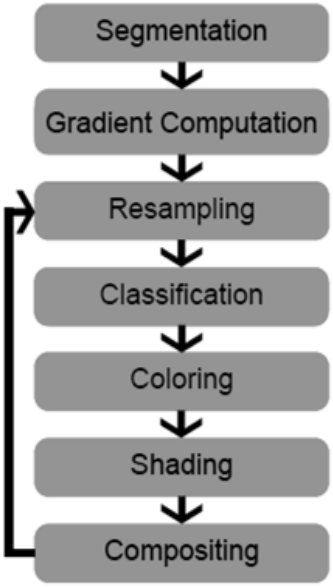

Figure 3: Generic volume-rendering pipeline

Segmentation

Segmentation is an optional step in the volume-rendering pipeline that partitions a single volume into multiple sub-volumes. The segmentation step is almost always performed just once before rendering. This is used extensively in the medical field when attempting to identify tumors or other masses within a patient.

Visualizing segmented data is typically done in one of three ways, visualizing the segmented data separate from the original volume, render the segmented volume as a surface within the original volume, or tag and store each voxel contained within the original volume. These three methods can then be used in the rendering process to change how the segmented volume is visualized to give it greater contrast to the rest of the volume. This is typically done by rendering the segmented volume in a different color or as a higher opacity than the rest of the volume.

Gradient computation

Gradient computation is another optional step that is typically only calculated once before rendering the volume. Gradients improve visual quality of the render by finding all the edges or boundaries between different materials in the volume. Three of the more commonly used gradient methods are the Central Different Gradient Estimator, the Intermediate Different Operator, and the Sobel Operator.

Resampling

Resampling is typically the first step in a continuous render loop. The goal of resampling is to be able to sample the volume data at different positions in three dimensional space to be used later in the render. The sampling techniques used are one of the primary differences between volume rendering techniques. Texture slicing samples the 3D space along planes that slice through the volume. Raycasting, on the other hand, samples at evenly space intervals along a ray.

Classification

Classification is the process of determining the subset of interpolated points to make up the final image. This is one of the most powerful tools in volume rendering from a visualization standpoint as it determines what parts of the volume are shown. This is done by mapping the voxel intensity values to opacity values between zero and one.

Coloring

Coloring is another step in the rendering process that provides immense power in understanding data. The purpose of coloring is not always to provide photo realism, but to provide sufficient contrast to identify desired features.

Shading

Shading is the simulation of reflections of light and shadows on a surface. The importance of shading cannot be understated as it provides a better understanding of surface contours and depth in a computer graphics scene. In volume rendering, the shading model is applied per voxel after coloring. Figure 18 shows the importance of shading on a CT of a human skull.

Figure 3: Shading on a human skull CT scan. Left shows no shading. Right image shows the same skull with shading

Compositing

The last stage of the volume-rendering pipeline is to take all the voxel data sampled by an individual ray and composite the information into a single color for that pixel. This compositing process is typically done either in a back-to-front or a front-to-back order for each ray. Front-to-back compositing is more commonly used because of the performance benefits and is the method used in this research.

IV. 4D VOLUME RENDERING & TOOLS

Rendering medical data in 4D presents its own sets of problems and limitations. All of the issues relevant to 3D volume rendering are present in 4D rendering, with added challenges of much more data and rendering multiple volumes simultaneously. There are also new visualization strategies that must be created to reveal important information in the data. In 3D volume rendering, there is one set of volumetric data for the entire scene. In 4D rendering, there is one set of volumetric data for each sample time causing the amount of data to increase linearly with the number of samples taken. Thus, methods are required to handle the increase in data and allow rendering at interactive speeds.

There are three commonly used open source 4D volumetric visualization tools,Osirix, MeVisLab [10], and Va3D. OsiriX is an FDA approved volume renderer that started as an academic project and then moved to open source development supporting Mac OS X and iOS. The MeVisLab tool is built in a modular fashion to allow developers and researches to be able to extend the software's abilities by adding or replacing modules. The module for 4D data visualization is unfortunately restricted to fluid flow visualization, which is not entirely helpful for visualizing fMRI data. Vaa3D is a cross platform tool that uses iso-surface rendering instead of direct volume rendering algorithms.

VIPRE-fMRI was created to address this need for a single cross platform 4D volume rendering application programming interface (API). Figure 28 shows an overview of the VIPRE-fMRI library architecture. The boxes at the bottom of the figure represent the lowest level systems and libraries being used.

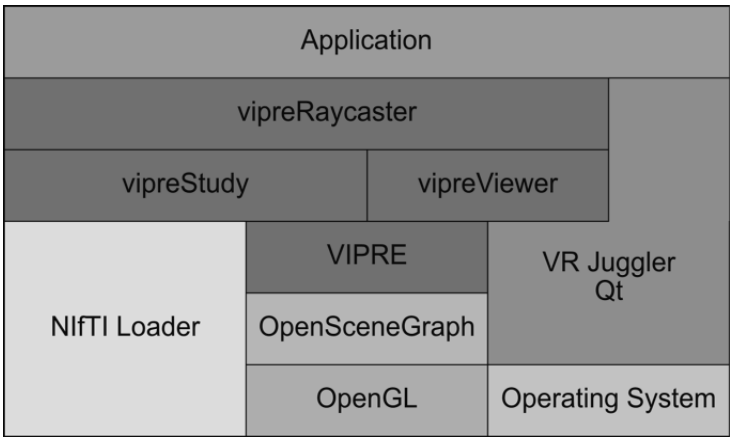

Figure 4. Simplified library architecture

The third-party APIs were combined into VIPRE-fMRI, a cross platform 4D volume rendering API. The original VIPRE was proposed by Noon [2] but was limited to volume rendering a single 3D volume. This work extends VIPRE by adding 4D volume raycasting. VIPRE-fMRI supports Windows, Linux, and Mac OS X. The rendering is abstracted from the developer, allowing different windowing APIs like Qt, Cocoa Touch, and VR Juggler to be used on their respective hardware platforms.

V. DESKTOP IMPLEMENTATION

The desktop application was built off the same raycasting renderer code with the only modifications being the windowing system used to display the results. Qt is a freely available user interface library that was used as the windowing system to display the raycasting results. OSG's window independence allows VIPRE-fMRI to be rendered directly within an OpenGL widget provided by Qt. No other changes were required to the VIPRE-fMRI rendering codebase.

A graphical user interface was created around the Qt OpenGL widget to all users to adjust things like coloring and tissue density through the use of interface elements like button and slider bars. The mouse and keyboard interactions were also handled by Qt and passed to the VIPRE-fMRI renderer as needed. An example of the desktop application viewing fMRI data can be seen in Figure 5.

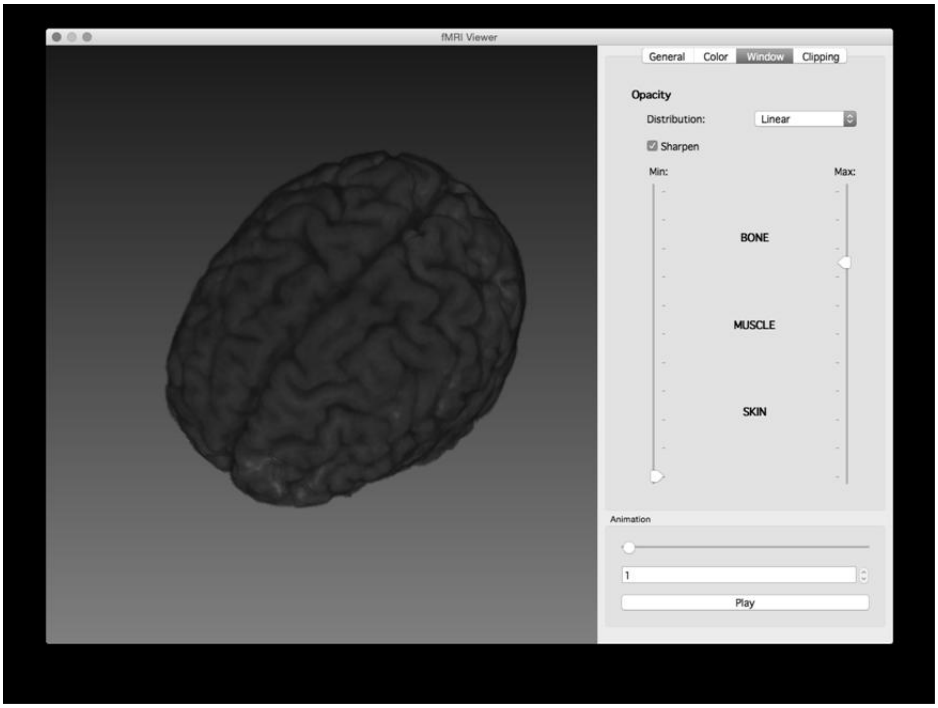

Figure 5. Desktop application visualizing a human brain

VI. MOBILE RAYCASTING

The Metal shading language was introduced for iOS devices with support for 3D textures and a focus on low level control of the graphics pipeline. The initialization of the application required the setup of the Metal graphics pipeline. This includes steps such as initializing the vertex buffers, textures, and compiling the shaders for use. Shaders are small programs that can be written to run on a GPU and is where the volume rendering logic is written. The render loop consisted of passing in the vertex buffers, textures, and uniforms to the shader to process. A shader uniform is simply a value passed into the shader from the application. In general, it is desirable to move as many operations as possible into the initialization step to allow faster drawing in the render loop step. The volume raycasting pipeline, as implemented using Metal, can be seen in Figure 6.The pipeline is broken up into the Application Compilation, Application Initialization, and Render Loop steps.

Traditionally, a shader accepts the volumetric data as an input in the form of a texture or a set of textures. In this research, each volume is passed into the shader as a 3D texture. Each texel in the texture represents a single voxel's intensity value. The high-resolution structural data is stored in a single 3D texture and the low-resolution functional data is stored in a series of 3D textures where each texture represents a single snapshot in time.

Loading data is another time consuming process that is best performed upon initialization. When loading the volume data, there were two options for storing it. The first was to store each volume as a 3D texture and the second was to store the volume as an array of values. Once the graphics pipeline is initialized, the rendering loop can begin drawing. The shader starts by determining the start and end points of the ray for the current pixel. It is important to calculate the start and end points of the ray to determine the number of times the ray must sample. The number of samples the ray takes is directly proportional to rendering time.The shader then starts sampling at the start point and continues along the ray until it reaches the end point. The ray is sampling in 3D world space, but the volume data is defined as a 3D texture with its own coordinate system. Sampling the volume requires both the ray and the volume texture to be in the same coordinate space.

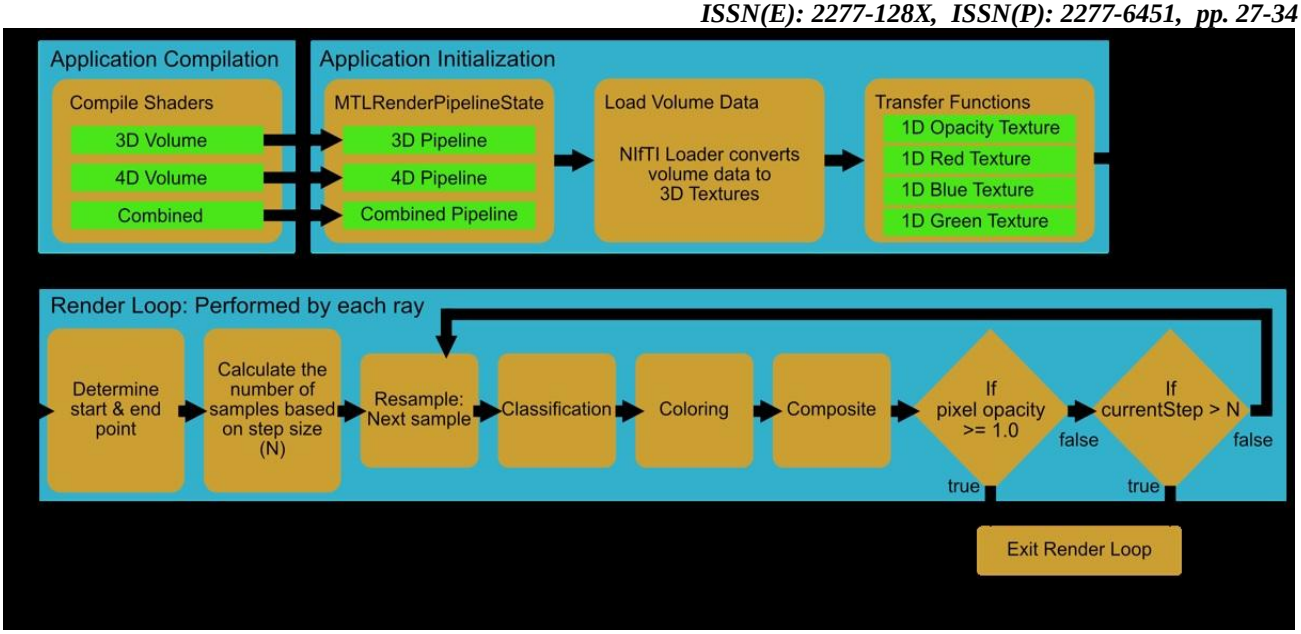

Figure 6: Volume raycasting pipeline implementation using Metal.

The converted ray can then be used in the ray casting loop to step along the ray and sample the volume at different points. The intensity value can then be used to sample the opacity transfer function texture for the correct opacity. If the opacity is not greater than zero, the ray moves to the next sample. If the opacity is greater than zero, the intensity value is used to determine a color using the color transfer function texture. The opacity and color are then combined to form the sample's contribution to the final pixel color. This sampling process continues until the end of the ray is reached or the pixel color becomes completely opaque.

VII. 4D VOLUME RENDERING WITH METAL

Moving from 3D rendering to 4D rendering required creating a different shader to handle changing volume data. The interest in 4D data comes from the change in activity from the baseline at different moments in time. Therefore, it was necessary to create a fragment shader that accepted two 3D textures, one representing the baseline activity and the second representing the time step of interest. Mapping the sample point in 3D world space to a point on each texture in their own space. Figure 7 shows a 2D example of the mapping issue. The left-side of the figure shows the structural (blue) and the functional (red) textures that represent the voxel data of their respective volumes. Both textures use a normalized coordinate system with (0,0) being the upper left corner and (1,1) being the lower right corner. Each box in the image represents a voxel. The right-side of the image shows the world space where both volumes are aligned and scaled with the functional volume being scaled up equally in both directions. The world space uses a coordinate system where the center of the volumes is at (0,0) and the height and width of the structural volume is 10. If the renderer samples the volume at world space location (0,0), the mapped texture position would be (0.5, 0.5) for both the structural and the functional texture. However, if the renderer samples the volumes at (5,-4), the sample falls outside the bounds of the functional volume but not the structural volume. Therefore, the structural texture is sampled at (1, 0.9) and the value for the functional volume is set to zero as it does not exist at that point.

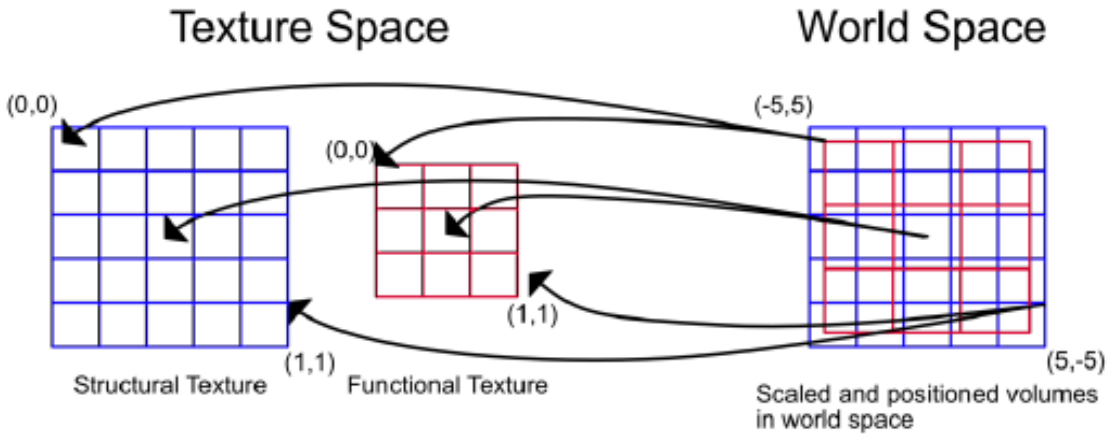

Figure 7: Mapping sampled in 3D world space to 2D textures.

Once both volumes are correctly sampled, the intensity values must be classified and colored. The structural volume's intensity value went through the same opacity and coloring operations as previously described for 3D rendering. The functional data went through the same coloring process as described above for 4D rendering where the activity level is irectly proportional to the intensity of a single color, in this case red. The structural and functional color components were then added together to obtain a final color value for that sample. This process was then repeated for every sample along the ray. The result can be seen in Figure 8 where the structural data is colored using a blue coloring scheme and the functional data is represented in red.

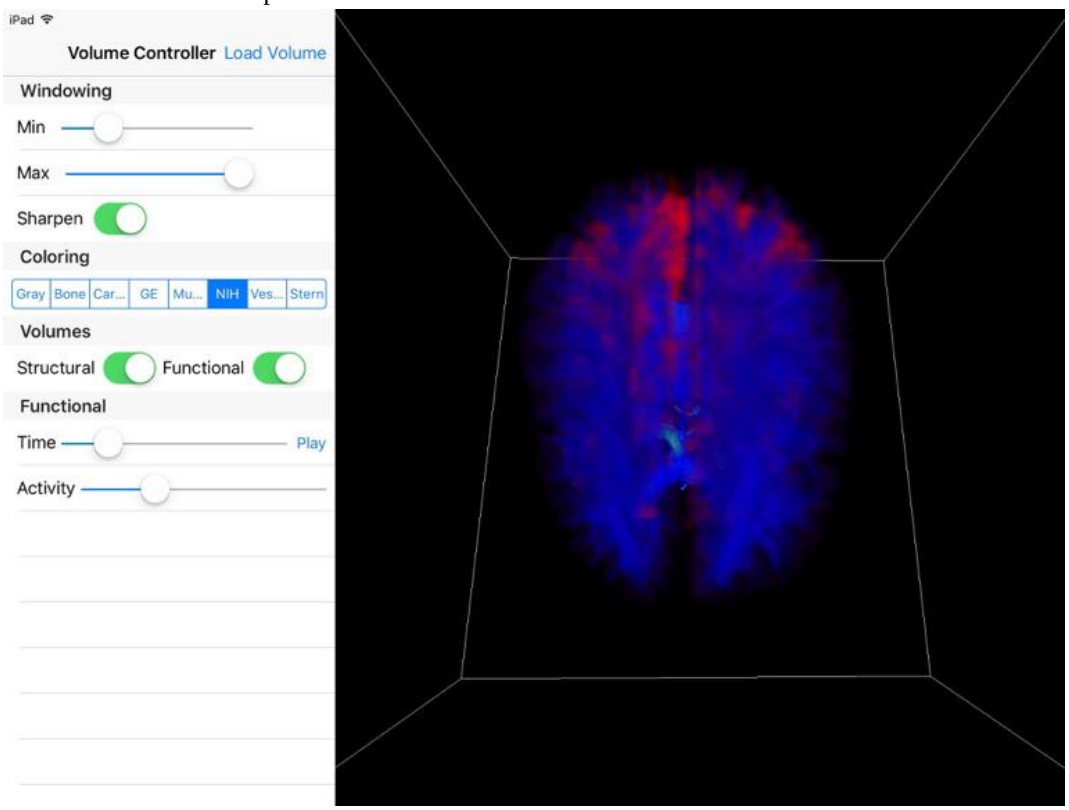

Figure 8: Combined structural and functional data for the brain

VIII. CONCLUSION

This research presented VIPRE-fMRI, a cross platform 4D fMRI volume rendering library for visualizing fMRI data on multiple immersive VR hardware platforms. Two sample applications were built to test VIPRE-fMRI. The first platform was the world's highest resolution six-sided VR CAVEÔ and the second platform was commodity desktop computer. The most difficult aspect of building a cross-platform 4D volume rendering library was selecting tools (software libraries and languages) that would work across all the variety of systems. OpenGL and the GLSL shader language were chosen for this reason. Frame rates show the library to be able to perform raycasting at 60 fps on commodity desktop hardware and 20 frames per second on a 96 node graphics cluster. The frame rates could be increased with several optimizations. Currently, the raycasting shader is a single program with multiple computationally expensive conditional statements.

This also looked at the feasibility of implementing a real-time 3D and 4D fMRI volume rendering method on a mobile device given the recent updates in graphic computational performance and new graphics languages. The application initialization,frame rates, and memory footprint results indicate that it is feasible to implement a raycasting method on a mobile device, but the frame rates are still not consistently high enough to be considered real-time. This is especially true for the 4D functional data which saw low frame rates. However, the performance will improve as hardware continues to improve.

IX. FUTURE WORK

Volume raycasting for mobile devices has been a challenge due to the limited hardware computational power and a lack of support for 3D textures. Mobile devices are generally underpowered compared to desktop computers because they intentionally sacrifice computational power for size to keep devices thin and light and provide ample battery life. Raycasting samples generally fall between voxels requiring some method of interpolation to obtain a correct value. Interpolating 3D positions within a set of 2D textures required a prohibitively large number of graphics operations.

REFERENCES

[1] D.J. Goodenough, K.E. Weaver, Overview of Computed Tomography, IEEE Trans. Nucl.Sci. 26 (1979) 1661–1667. doi:10.1109/TNS.1979.4330458.

[2] P.A. Bottomley, Nuclear magnetic resonance: Beyond physical imaging: A powerful new diagnostic tool that uses magnetic fields and radio waves creates pictures of the body's internal chemistry, IEEE Spectr. 20 (1983) 32–38. doi:10.1109/MSPEC.1983.6369002.

[3] K.K. Shung, M.B. Smith, B. Tsui, Principles of Medical Imaging, Academic Press, Inc.,San Diego, CA, 1992.

[4] S. Roettger, S. Guthe, D. Weiskopf, T. Ertl, W. Strasser, Smart Hardware-Accelerated Volume Rendering, in: IEEE TCVG Symp. Vis., 2003: pp. 231–238.

[5] W.M. Hsu, Segmented ray casting for data parallel volume rendering, in: Proc. 1993 IEEE Parallel Render. Symp., 1993: pp. 7–14. doi:10.1109/PRS.1993.586079.

[6] K. Ma,Parallel volume raycasting for unstructured grid data on distributed memory architectures,in:IEEE Symp . Parallel Render.1995,1995:pp.23–30.doi:10.1145/218327. 218333.

[7] K.-L.M.K.-L. Ma, T.W. Crockett, A scalable parallel cell-projection volume rendering algorithm for three-dimensional unstructured data, in: Proc. IEEE Symp. Parallel Render.,1997: pp. 95–104. doi:10.1109/PRS.1997.628300.

[8] F. Di Salle, E. Formisano, D.E. Linden, R. Goebel, S. Bonavita, a Pepino, et al., Exploring brain function with magnetic resonance imaging., Eur. J. Radiol. 30 (1999) 84–94.http://www.ncbi.nlm.nih.gov/pubmed/10401589.

[9] P.T. Fox, M.E. Raichle, Focal physiological uncoupling of cerebral blood flow and oxidative metabolism during somatosensory stimulation in human subjects., Proc. Natl.Acad. Sci. U. S. A. 83 (1986) 1140–1144. doi:10.1073/pnas.83.4.1140.

[10] A. Hennemuth, O. Friman, C. Schumann, J. Bock, J. Drexl, M. Huellebrand, et al., Fast Interactive Exploration of 4D MRI Flow Data, 7964 (2011) 79640E–79640E–11.doi:10.1117 /12.878202.

[11] C.J. Noon, A Volume Rendering Engine for Desktops, Laptops, Mobile Devices and Immersive Virtual Reality Systems using GPU-Based Volume Raycasting by, Iowa State University, 2012.

