REAL-TIME VOLUME RENDERING AND TRACTOGRAPHY VISUALIZATION ON THE WEB

John Congote, Esther Novo, Luis Kabongo
Vicomtech Research Center
Donostia - San Sebastian, Spain
jcongote,enovo,lkabongo@vicomtech.org

Dan Ginsburg
Children's Hospital
Boston, United States
dginsburg@upsamplesoftware.com

Stephan Gerhard
Institute of Neuroinformatics
Uni/ETH Zurich, Switzerland
connectome@unidesign.ch

Rudolph Pienaar
Harvard Medical School
Boston, United States
Rudolph.Pienaar@childrens.harvard.edu

Oscar E. Ruiz
Universidad EAFIT
Medellin, Antioquia
oruiz@eafit.edu.co

ABSTRACT

In the field of computer graphics, *Volume Rendering* techniques allow the visualization of 3D datasets, and specifically, Volume Ray-Casting renders images from volumetric datasets, typically used in some scientific areas, such as medical imaging. This article aims to describe the development of a combined visualization of *tractography* and *volume rendering* of brain T1 MRI images in an integrated way. An innovative web viewer for interactive visualization of neuro-imaging data has been developed based on *WebGL*. This recently developed standard enables the clients to use the web viewer on a wide range of devices, with the only requirement of a compliant web-browser. As the majority of the rendering tasks take place in the client machine, the effect of bottlenecks and server overloading are minimized. The web application presented is able to compete with desktop tools, even supporting high graphical demands and facing challenges regarding performance and scalability. The developed software modules are available as open source code and include MRI volume data and tractography generated by the Diffusion Toolkit, and connectivity data from the Connectome Mapping Toolkit. Our contribution for the Volume Web Viewer implements early ray termination step according to the tractography depthmap, combining volume images and estimated white matter fibers. Furthermore, the depthmap system extension can be used for visualization of other types of data, where geometric and volume elements are displayed simultaneously.

1 INTRODUCTION

Three-dimensional data can be found in several scientific fields, coming from simulation, sampling or modeling processes. Regarding the biomedical scope, several scanning techniques, such as magnetic resonance (MRI) or computerized tomography (CT), are used for storing body imaging samples as volumetric datasets formed by groups of parallel slices, where the term volumetric dataset refers to a scalar field. These datasets are usually visualized in three dimensions in order to facilitate specialists to interpret information.

Visualization of medical volumetric datasets can suitably be performed by the use of *Direct Volume Rendering* algorithms. These methods show important characteristics of datasets, even though rendering is not usually photo-realistic.The problem addressed in this paper is the visualization of tractography information obtained from dMRI (diffusion MRI) together with volume data corresponding to MRI or CT images.

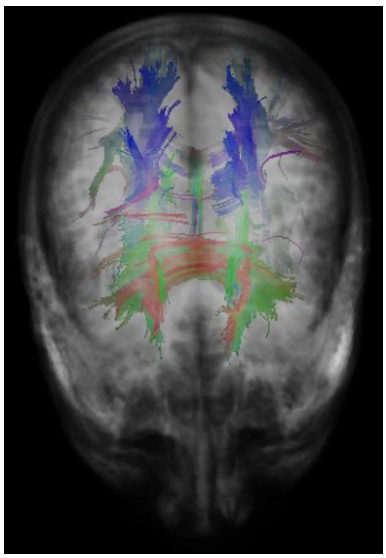


Figure 1: Combined visualization of volume rendering and tractography information on the web

In order to represent the volumetric datasets, volume rendering techniques allow the visualization of all inner characteristics of volumes at once, by projecting data into 2D images, according to the corresponding

position of a virtual camera. The main idea of the ray-casting algorithm is to launch rays from the camera into the volume, calculating the volume rendering integral along the rays. Thus, in this method, the colour and opacity of each pixel in the final image is evaluated by launching a ray in the scene from the view position, sampling the volume at discrete points along the ray and accumulating the contribution of each sample.

Our contribution is an implementation of a web rendering system for medical images, which integrates volume rendering and geometric objects within a compliant WebGL browser, based on the volume ray casting algorithm and built on previous developments [CSK+11]. Due to the technology limitations of Webgl, the improvements developed allow us to create a web application for combined visualization of volume rendering and tractography, as shown in Figure 1, being able to compete with desktop tools, supporting high graphical demands and facing challenges regarding performance and scalability.

The article is organized as follows. Section 2 presents the work related to this article, including a description of volume rendering techniques, visualization of medical images and geometry intersection. The methodology of the developed work is explained in Section 3. Then, the results accomplished are presented, and finally, Section 5 states the conclusions and future developments.

2 RELATED WORK

2.1 Volume Rendering

In computer graphics, Ray Casting is a well known direct volume rendering technique that was designed by Kajiya and Herzen [KVH84] as one of the initial developments in this area. Traditionally, three dimensional objects have been created by using surface representations, drawing geometric primitives that create polygonal meshes [Lev88], hence provoking the loss of information from one dimension.

Further developments [DCH88] accomplished the mathematical modeling of the ray casting process, based on the light's behaviour equations. Thus, the volume rendering integral was defined. A comparative between different direct volume rendering algorithms, such as Texture Mapping, Ray Casting, Splatting or Shear Warp, was presented [MHB+00]. Ray casting is a flexible algorithm that allows the implementation of acceleration methods, such as Empty Space Skipping [KW03] or *Early Ray Termination*. Early ray termination is an optimization process that establishes certain limitations in the volume, so that the samples encountered after them do not contribute to the value of the pixel.

Ray casting suitably fits GPUs' operating mode [Sch05], because of the independence of each ray that is launched to the scene, making this algorithm highly parallelizable and allowing the exploitation of GPUs' parallel architecture. For GPU ray casting, the volume element is stored in the GPU memory as a 3D texture and a fragment shader program is used in order to implement the ray casting algorithm.

A quality evaluation model was developed for comparing the different Direct Volume Rendering techniques [BBD+07]. These methods handle a higher amount of data than surface rendering techniques, therefore, the complexity of the algorithms is increased, as well as the necessary rendering time [Bru08]. Optimized volume rendering methods avoid empty spaces by introducing a volume proxy geometry [MRH08].

Web 3D Rendering

The use of the recently released WebGL standard [Mar11] leads to new methods for web 3D visualization, where most part of the computational processes are performed in vertex and fragment shaders that run on the GPU hardware. WebGL is a software library that enables HTML5-based browsers to identify clients' graphics hardware. HTML5, the latest Internet standard propose, provides native elements for audio and video. WebGL consists of a low-level imperative graphic programming API based on OpenGLES 2.0 for Javascript that enables flexibility and exploits the characteristics of advanced graphics cards. Due to the constant improvement of the performance of Javascript interpreters, the management of scene elements behaves similarly to the ones obtained by using natively compiled languages. Moreover, some WebGL extensions have been implemented in order to achieve a friendly interaction, such as SpiderGL [DBPGS10].

Several standards and proprietary solutions are currently being developed in order to fulfil the necessity of moving 3D visualization into the web [BA01], such as X3D, a standard derived from VRML that stores 3D information in a scenegraph format using XML (Extensible Markup Language). This model has been implemented in a declarative form, as an extension of HTML; X3DOM presents a framework for integrating X3D nodes into HTML5 DOM content [BEJZ09] and other alternatives have also been developed, e.g. XML3D [SKR+10]. Finally, there is a standardization for X3D in the MedX3D volume rendering model [JAC+08, PWS11].

2.2 Visualization of Medical Images

Medical visualization is a challenging scientific field because interpretation of images may lead to clinical intervention. Therefore, quality and fast interactive response are important features in this domain. Remarkable advances have occurred in medical imaging technology and applications in the past few years, supporting the possibility of sharing imaging data online across clinical and research centres and among clinicians and patients. The development of these kind of applications

is influenced by connectivity, security and resources' heterogeneity concerns.

On-server rendering can be considered a partial solution for Medical Imaging [BM07]. Moreover, several web implementations for volumetric visualization have already been presented [JAC+08], although many of these solutions require third party systems to allow visualization or their scalability is limited by the rendering server.

As medical volumetric imaging requires high fidelity and high performance, several rendering algorithms have been analyzed, leading to thread- and data-parallel implementations of ray casting [SHC+09]. Thus, architectural trends of three modern commodity parallel architectures are exploited: multi-core, GPU, and Intel Larrabee. Other approaches describe the development of web-based 3D reconstruction and visualization frameworks for medical data [SAO10]. Such applications based on X3D technology allow extending cross-platform, inter-application data transfer ability. Several applications have been implemented using web 3D rendering techniques, for example, evaluation systems at the educational level [Joh07] or medical training simulations [JROB08].

dMRI

Diffusion Magnetic Resonance Imaging (dMRI) relies on the visualization of water diffusion using data from MRI. Diverse methodologies have been presented over the last years and can be classified into two categories: Image based and Object based techniques. The first methodology divides the space in voxels and the assigned colour represents the principal diffusion direction [MAA+03]. However, tracks can not be easily identified since no segmentation of the visualization is performed, and therefore direction information is difficult to observe since voxel colour mapping is not one-to-one, *i.e.*, different directions might be represented by the same colour. Otherwise, in object based methodologies, objects, such as ellipsoids and lines, are used together with colour mapping in order to enhance visualization and give a direction sense to the representation.

Visualization of brain white matter cortical tracks is one of the most important applications of dMRI, since it allows to non-invasively visualize white matter anatomy, and detecting of anomalies [NVLM07, GKN+11]. Tractography, which refers specifically to the representation of the white matter tracks based on the water diffusion information, employs lines to represent the diffusion direction and to visualize the white matter paths. In general, lines are generated using randomly distributed seed points; together with the principal diffusion information and a prescribed interval of time, the different paths are generated. However, this representation becomes dependent on the amount and location of seed points to correctly visualize tracks [EKG06] because erroneous connections might be produced between tracks due to the existing error in data. Incorrect visualization of branching of tracks is another drawback, since only one path is generated per each seed point.

Probabilistic methodologies have been proposed [EKG06] to represent branching of white matter tracks, in which secondary seed points are included in regions in which branching is assumed. Therefore, a denser visualization is performed in those regions. An algorithm was proposed for path visualization [RSDH10], in which the different global paths are simplified by one simple curve, clustering the different paths and then using average curves to obtain one simple curve that summarizes each cluster.

2.3 Geometry Intersection

The application described in this article requires representing volume rendering and tractography together, *i.e.*, both volumetric and polygonal data have to be displayed in the same scene. There are several models for combining polygonal geometry and volume rendering. Some methods identify the intersection between rays launched in the volume rendering process and geometry [SMF00]. This technique can be optimized by creating octrees for dividing the geometric space and prove intersections correctly.

Other models try to achieve a correct visibility order for the intersections between volume and geometry [HLSR09]. Geometry has to be rendered in the first place to correctly look at the intersections of the geometry and the volume. Besides, parts that are occluded by the geometry should not contribute to the final image, not performing any ray casting at all. In order to achieve this feature, rays should terminate when they hit a polygonal object, accordingly modifying the ray length image if a polygonal object is closer to the view point than the initial ray length.

3 METHODOLOGY

In our project, the results of the Connectome Mapper are directly loaded in the browser using WebGL and JavaScript. The FreeSurfer cortical surface reconstruction binary files are loaded and processed in JavaScript and converted to WebGL vertex buffer objects for rendering. The surfaces are overlaid with per-vertex curvature values computed during the FreeSurfer processing stream. The tractography data is likewise parsed in the JavaScript code and rendered as line primitives coloured based on direction. Finally, the structural network itself is converted to JSON (JavaScript Object Notation) as an offline preprocess and loaded into the browser using JavaScript. The networks are visualized in 3D along with the fiber tracts and volumes enabling exploration of connectivity information in real-time.

The work described in this paper has been developed using volume ray casting, a widely used algorithm for

generation of 2D representations from three dimensional volumetric datasets. The obtained images are 2-dimensional matrices $I : [1,h] \times [1,w] \rightarrow \mathbb{R}^4$ (w: width and h: height, both in pixels). Each pixel is represented by a colour expressed by a four-tuple of red, green, blue and alpha real-valued components, $(R,G,B,A \in [0,1])$.

An entire volume is represented by a 3-dimensional array of real values $V : [1,H] \times [1,W] \times [1,D] \rightarrow [0,1]$ (H: Height, W: Width, D: Depth of the represented volume, all of them in positive integer coordinates). Therefore, $V(x,y,z) \in [0,1]$. The projection model used in this work is called pin-hole camera [HZ03]. The pin-hole camera technique uses intrinsic $K \in M_{3\times4}$ and extrinsic $R \in M_{4\times4}$ real-valued matrices in order to project every 3D point $p \in \mathbb{P}^3$ onto a 2D point $p' \in \mathbb{P}^2$.

The volume ray casting algorithm defines the colour for each pixel (i,j) in the image, which is also known as projection screen, I, according to the values of a scalar field $V(x,y,z)$. This scalar field is associated to the points (x,y,z) reached by rays that are originated at a certain pixel or camera, represented as C in Figure 2. A cuboid geometry is generated with coordinates $(0,0,0)$ to $(1,1,1)$. This cube represents the boundary established for the volumetric dataset. Each ray intersects with the cuboid volume V at points $p_{(i,j)}(x,y,z)$ and $q_{(i,j)}(x,y,z)$, which represent the input and output coordinates of the ray into and out from the volume, respectively.

Then, each obtained ray pq is equi-parametrically sampled. For every sampled point $s(x,y,z)$ over the ray, an approximation of the scalar field $V(s)$ is calculated, commonly by using trilinear interpolation. The sampled points also influence the colour of the originating pixel, due to the use of a composition function (Equations 1-4), where the accumulated colour A_{rgb} is the colour of the point s in the volume V, and A_a is the transparency component of the pixel, which has a value of 1 at the end of the rendering process. Given a certain set of coordinates (x,y,z) in the volume and a ray step k, V_a is the scalar value of the volume V, V_{rgb} is the colour defined by the given transfer function V_a, S represents the sampled values over the ray and O_f, L_f are the general Opacity and Light factors.

$$S_a = V_a \times O_f \times \left(\frac{1}{s}\right) \tag{1}$$

$$S_{rgb} = V_{rgb} \times S_a \times L_f \tag{2}$$

$$A^k_{rgb} = A^{k-1}_{rgb} + \left(1 - A^{k-1}_a\right) \times S_{rgb} \tag{3}$$

$$A^k_a = A^{k-1}_a + S_a \tag{4}$$

In the ray casting process performed in this work, geometry G is formed by a set of segment lines L (although G could also be represented as a set of points P or triangles T). Each segment L is defined by two points in the space. Lines are projected through projection matrices onto a different image, where the values of colour (r,g,b,a) and depth $(depth)$ are defined for each pixel (x,y).

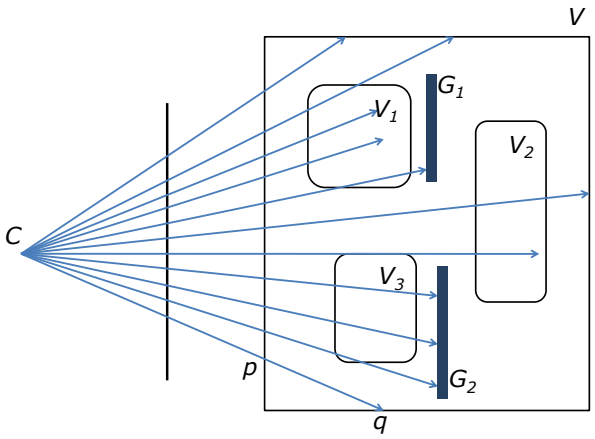


Figure 2: 2D representation of the ray casting algorithm performance (types of ray termination)

Each pq ray traverses the cuboid volume V, where both volume elements V_i and geometries G_i are rendered in the same process by modifying the early ray termination method, as depicted in Figure 2. This technique checks the alpha value for each sample of the transparency colour of the ray. If the value V_a is equal to 1, which means that the ray has reached the final colour, the remaining steps of the ray are not evaluated. Rays might terminate due to several reasons: when encountering a very dense volume (such as V_1 in fig. 2), when intersecting with a geometric element (*e.g.* with G_1) or when exiting the boundary cube, at point q.

The early ray termination model is also used to check the length of the ray and compare it to the depthmap of the figure. In conclusion, a projection of the geometry is obtained, as well as the colour and depth for each pixel in the image. This information can be compared to the length of the ray, terminating the ray when the alpha value is 1 or when the depth is equal to the geometry depth.

4 RESULTS

This section describes the accomplished implementation of a real-time web viewer for both direct volume rendering and tractography visualization. This work is based on the WebGL standard and performs the ray casting algorithm with an early ray termination optimization.

4.1 Tractography

The Connectome Mapper [GDL+11] is a publicly available software that provides a pipeline to automatically generate structural networks from raw dMRI data of the brain. Gray and white matter segmentations are obtained by processing T1 MPRAGE MRI using the Freesurfer set of tools. The Diffusion Toolkit is used later for reconstruction. A deterministic streamline

algorithm is used to obtain tractography, by generating fiber tracts of the same subject. For cortical and subcortical regions of interest, a parcellation is performed. Finally, these datasets are coregistered and a network is generated by weighting the connectivity between regions based on the fiber tracts [GGCP11].

4.2 Data Processing and Volume Interpolation

For the developed work, all the slices that correspond to a certain volume are composed into a single image, as shown in Figure 3. This image is generated by placing slices in a matrix configuration as a preprocessing step of the rendering algorithm. The size of the image stored in GPU memory could range from 4096×4096 on a PC (which can contain up to 256^3 volume) to $1024x1024$ on other devices (which can contain up to $128 \times 128 \times 64$). The screen resolutions being reduced on mobile devices it seems reasonable to scale down or even crop the volumes original dimensions in order to match the maximum GPU available memory.

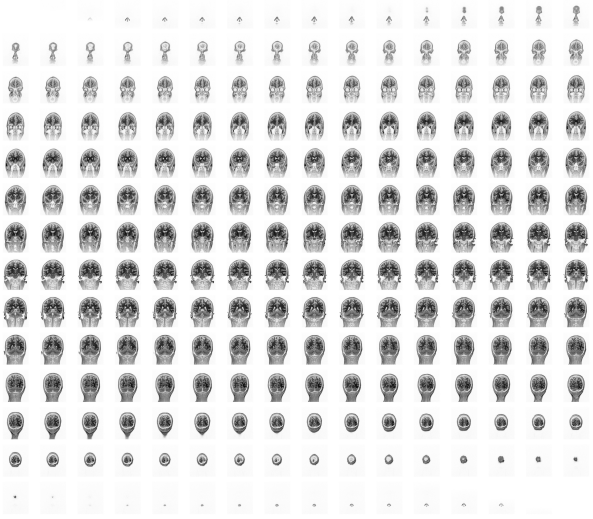


Figure 3: Brain dataset in mosaic form, read by the shader

In medical imaging, the sample bit depth is usually higher than 8 bits per pixel. This is a drawback that has to be handled for the development of web applications, where commonly supported formats are limited to 8 bits per sample. In the described experiment, information from medical datasets was reduced to 8 bits per sample.

Identification of Ray Coordinates

According to the ray casting algorithm, the displayed colours of the boundary cuboid geometry represent the coordinates at each point (x,y,z). Coordinates are stored as r,g,b colour components for each pixel. Then, the cube can be rendered in the scene from the desired view point. In order to achieve volume visualization, several steps are followed in the rendering process. First of all, the rendering of the colour cube is performed according to the depth function change.

Taking this into account, rays are defined for each point of the cube, starting at the front faces, where the virtual camera is located, and ending at the back region. The colour of every point of the cube represents the exact coordinates of the ray for each pixel in the image. The colour information is stored as 24 bit RGB values. The range of values that can be represented may seem small or imprecise for large images, but colour interpolation provides precision enough for ray coordinates. The depth information is stored in different buffers in order to obtain the corresponding depth value for each ray. Finally, the geometry is rendered and the colour and depth buffers are stored to be processed in the volume shader.

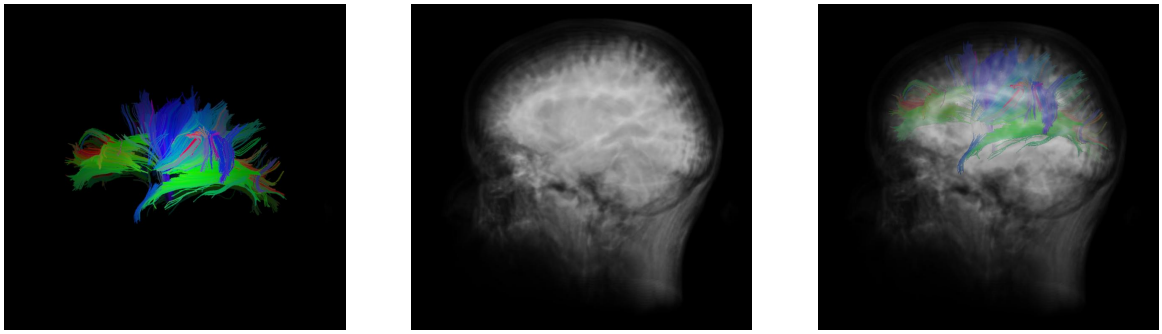
4.3 Visualization

The previously presented GPU implementation of volume rendering based on WebGL was used to develop a real-time online tractography and volume rendering viewer, accordingly to Table 1, proving this standard to be a valid technology for real-time interactive applications on the web. The results shown in the table below were accomplished when interacting with the web viewer from several computers, using the same web browser (*Chrome*) and the same number of steps, 50. For every graphic card tested, the application can be completely considered to have a real-time behaviour.

Graphic card model	Frame rate
NVidia GeForce GTX480	60 fps
NVidia GeForce GTX285	60 fps
NVidia 9600GT	28 fps
NVidia Quadro FX 3800M	20 fps
NVidia Quadro FX 880M	15 fps

Table 1: Performance of the developed viewer for different graphic cards, using Chrome as web browser, the number of steps equal to 50

In the developed work, the web viewer shows tractography information obtained from dMRI in the first place, represented in Figure 4(a). These organized fiber tracks in the white matter connect various cortical regions to each other. The tractography is represented using WebGL line primitives, where each fiber track is rendered by a set of points. The colour is assigned based on the absolute value of the unit vector pointing in the direction from the start point to the end point of the tract. The length value of each tract is stored in a per-vertex attribute together with the position and colour. The minimum tract length value is placed in a uniform variable in the vertex shader. The vertex shader determines whether the tract is longer than the minimum length to render. The entire tractography set for the brain is



(a) Tractography (i) (b) Volume Rendering (i) (c) Combined visualization (i)

Figure 4: Tractography, volume rendered image of brain T1 MPRAGE MRI and combined visualization on the web

efficiently rendered using a single draw call with one vertex buffer object. Thus, no dynamic geometry generation is performed in JavaScript.

Direct volume rendering of MRI data (Figures 4(b)) is developed simultaneously with the tractography. The volume renderer loads the MRI dataset from the server into a tiled 2D texture. Then, ray-tracing is performed in the shader in order to obtain the volume rendering. This implementation of a volume rendering system for the Web is based on the Volume Ray-Casting algorithm. Since the algorithm is implemented in WebGL, the reached visualization speed is similar to native applications, due to the use of the same accelerated graphic pipeline. The algorithm simulates 3D data by using a 2D tiling map of the slices from the volume maintaining trilinear interpolation and runs entirely in the client.

In the developed Web viewer, shown in Figure 5, the tractography and the volume rendering from brain MRI data can be represented separate or simultaneously, as depicted in Figures 4(c). Several features can be modified at runtime, by adjusting the provided sliders. Tractography's position can be changed according to the three main axes and fiber tracks can be seen more clearly by reducing the volume opacity. Finally, the minimum tract length can also be modified.

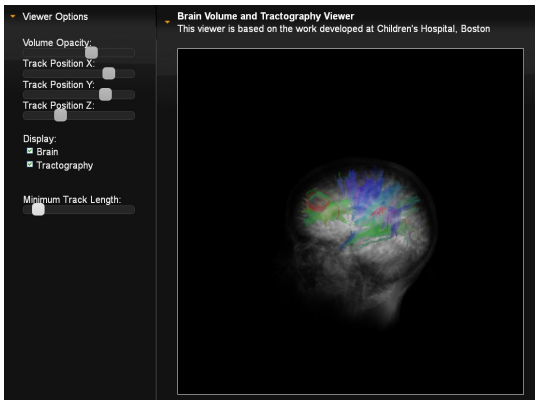


Figure 5: Volume rendering and tractography web viewer (sliders available for configuration)

5 CONCLUSIONS AND FUTURE WORK

This paper describes the successful implementation of remote visualization of medical images based on WebGL[1]. Interaction with remote medical images was limited by many technical requirements, but the emergence of recent standards such as WebGL and HTML5 allow the development of applications that enable clients to access images without downloading them, maintaining data in a secure server and being able to perform functions, *e.g.* registration, segmentation, *etc.*, in a web context. These technologies empower web browsers to handle 3D graphics naturally. Thus, modern browsers support a wide range of applications, from simple rendering of two dimensional images to complex manipulation of 3D models.

The achieved visualization of volume rendering and tractography on the web, used for the implementation the presented viewer (shown in Figure 5), has demonstrated the capabilities of complex volume rendering visualization in web browsers, as well as the potential of WebGL for interactive visualization of neuroimaging data. Combined representation of volume rendering of brain T1 MRI images and tractography in real time has been accomplished. The main strength of the WebGL standard used here is the ability to provide efficient access to GPU rendering hardware with no special client-side software requirements, except for a compatible browser. Thus, this platform has great potential for imaging tools, particularly those providing web-based interfaces for automatic pipelining of image data processing.

In the work explained herein, the early ray termination algorithm was modified in order to combine volume and geometric elements in a seamless way. Thus, the developed software modules, which are available as open source code, successfully implement early ray termination step according to the tractography depthmap,

[1] `http://www.volumerc.org/demos/brainviewer/webgl/brain_viewer/brain_viewer.html`

performing a combination between volume images and estimated white matter fibers.

6 ACKNOWLEDGEMENTS

This work was partially supported by CAD/CAM/CAE Laboratory at EAFIT University and the Colombian Council for Science and Technology -COLCIENCIAS-. Everyone who has contributed to this work is also gratefully acknowledged.

7 REFERENCES

[BA01] Johannes Behr and Marc Alexa. Volume visualization in vrml. In *Proceedings of the sixth international conference on 3D Web technology*, Web3D '01, pages 23–27, New York, NY, USA, 2001. ACM.

[BBD+07] Christian Boucheny, Georges-Pierre Bonneau, Jacques Droulez, Guillaume Thibault, and Stéphane Ploix. A perceptive evaluation of volume rendering techniques. In *Proceedings of the 4th symposium on Applied perception in graphics and visualization*, APGV '07, pages 83–90, New York, NY, USA, 2007. ACM.

[BEJZ09] Johannes Behr, Peter Eschler, Yvonne Jung, and Michael Zöllner. X3dom: a dom-based html5/x3d integration model. In *Proceedings of the 14th International Conference on 3D Web Technology*, Web3D '09, pages 127–135, New York, NY, USA, 2009. ACM.

[BM07] Bojan Blazona and Zeljka Mihajlovic. Visualization service based on web services. *29th International Conference on*, pages 673–678, 2007.

[Bru08] S. Bruckner. *Efficient Volume Visualization of Large Medical Datasets: Concepts and Algorithms*. VDM Verlag, 2008.

[CSK+11] John Congote, Alvaro Segura, Luis Kabongo, Aitor Moreno, Jorge Posada, and Oscar Ruiz. Interactive visualization of volumetric data with webgl in real-time. In *Proceedings of the 16th International Conference on 3D Web Technology*, Web3D '11, pages 137–146, New York, NY, USA, 2011. ACM.

[DBPGS10] Marco Di Benedetto, Federico Ponchio, Fabio Ganovelli, and Roberto Scopigno. Spidergl: a javascript 3d graphics library for next-generation www. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 165–174, New York, NY, USA, 2010. ACM.

[DCH88] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '88, pages 65–74, New York, NY, USA, 1988. ACM.

[EKG06] H.H. Ehricke, U. Klose, and W. Grodd. Visualizing mr diffusion tensor fields by dynamic fiber tracking and uncertainty mapping. *Computers & Graphics*, 30(2):255–264, 2006.

[GDL+11] S. Gerhard, A. Daducci, A. Lemkaddem, R. Meuli, J.P. Thiran, and P. Hagmann. The connectome viewer toolkit: an open source framework to manage, analyze, and visualize connectomes. *Frontiers in Neuroinformatics*, 5, 2011.

[GGCP11] Daniel Ginsburg, Stephan Gerhard, John Edgar Congote, and Rudolph Pienaar. Realtime visualization of the connectome in the browser using webgl. *Frontiers in Neuroinformatics*, October 2011.

[GKN+11] A.J. Golby, G. Kindlmann, I. Norton, A. Yarmarkovich, S. Pieper, and R. Kikinis. Interactive diffusion tensor tractography visualization for neurosurgical planning. *Neurosurgery*, 68(2):496, 2011.

[HLSR09] Markus Hadwiger, Patric Ljung, Christof R. Salama, and Timo Ropinski. Advanced illumination techniques for gpu-based volume raycasting. In *ACM SIGGRAPH 2009 Courses*, pages 1–166. ACM, 2009.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, second edition, 2003.

[JAC+08] N W John, M Aratow, J Couch, D Evestedt, A D Hudson, N Polys, R F Puk, A Ray, K Victor, and Q Wang. Medx3d: Standards enabled desktop medical 3d. *Studies In Health Technology And Informatics*, 132:189–194, 2008.

[Joh07] Nigel W. John. The impact of web3d technologies on medical education and training. *Computers and Education*, 49(1):19 – 31, 2007. Web3D Technologies in Learning, Education and Training.

[JROB08] Yvonne Jung, Ruth Recker, Manuel Olbrich, and Ulrich Bockholt. Using x3d for medical training simulations. In *Web3D '08: Proceedings of the 13th international symposium on 3D web technology*, pages 43–51, New York, NY, USA, 2008. ACM.

[KVH84] James T. Kajiya and Brian P Von Herzen.

Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18:165–174, January 1984.

[KW03] J. Kruger and R. Westermann. Acceleration techniques for gpu-based volume rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 38–, Washington, DC, USA, 2003. IEEE Computer Society.

[Lev88] Marc Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8:29–37, May 1988.

[MAA+03] Y. Masutani, S. Aoki, O. Abe, N. Hayashi, and K. Otomo. Mr diffusion tensor imaging: recent advance and new techniques for diffusion tensor visualization. *European Journal of Radiology*, 46(1):53–66, 2003.

[Mar11] Chris Marrin. *WebGL Specification*. Khronos WebGL Working Group, 2011.

[MHB+00] M. Meißner, J. Huang, D. Bartz, K. Mueller, and R. Crawfis. A practical evaluation of popular volume rendering algorithms. In *Proceedings of the 2000 IEEE symposium on Volume visualization*, pages 81–90. Citeseer, 2000.

[MRH08] Jörg Mensmann, Timo Ropinski, and Klaus Hinrichs. ccelerating volume raycasting using occlusion frustums. In *IEEE/EG Volume and Point-Based Graphics*, pages 147–154, 2008.

[NVLM07] P.G.P. Nucifora, R. Verma, S.K. Lee, and E.R. Melhem. Diffusion-tensor mr imaging and tractography: Exploring brain microstructure and connectivity. *Radiology*, 245(2):367–384, 2007.

[PWS11] Nicholas Polys, Andrew Wood, and Patrick Shinpaugh. Cross-platform presentation of interactive volumetric imagery. Departmental Technical Report 1177, Virginia Tech, Advanced Research Computing, 2011.

[RSDH10] N. Ratnarajah, A. Simmons, O. Davydov, and A. Hojjat. A novel white matter fibre tracking algorithm using probabilistic tractography and average curves. *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2010*, pages 666–673, 2010.

[SAO10] S. Settapat, T. Achalakul, and M. Ohkura. Web-based 3d visualization and interaction of medical data using web3d. In *SICE Annual Conference 2010, Proceedings of*, pages 2986–2991. IEEE, 2010.

[Sch05] Henning Scharsach. Advanced gpu raycasting. *Proceedings of CESCG*, 5:67–76, 2005.

[SHC+09] Mikhail Smelyanskiy, David Holmes, Jatin Chhugani, Alan Larson, Douglas M. Carmean, Dennis Hanson, Pradeep Dubey, Kurt Augustine, Daehyun Kim, Alan Kyker, Victor W. Lee, Anthony D. Nguyen, Larry Seiler, and Richard Robb. Mapping high-fidelity volume rendering for medical imaging to cpu, gpu and many-core architectures. *IEEE Transactions on Visualization and Computer Graphics*, 15:1563–1570, November 2009.

[SKR+10] Kristian Sons, Felix Klein, Dmitri Rubinstein, Sergiy Byelozyorov, and Philipp Slusallek. Xml3d: interactive 3d graphics for the web. In *Proceedings of the 15th International Conference on Web 3D Technology*, Web3D '10, pages 175–184, New York, NY, USA, 2010. ACM.

[SMF00] Marcelo Rodrigo Maciel Silva, Isabel Harb Manssour, and Carla Maria Dal Sasso Freitas. Optimizing combined volume and surface data ray casting. In *WSCG*, 2000.