Interactive visualization of DICOM volumetric datasets in the Web
Providing VR experiences within the web browser

Ander Arbelaiz[1], Aitor Moreno[1], Luis Kabongo[1,2], Helen V. Diez[1] and Alejandro García Alonso[3]
[1]*Vicomtech-IK4, Paseo Mikeletegi, 57, 20009 Donostia / San Sebastián, Spain*
[2]*Biodonostia Health Research Institute, Donostia / San Sebastián, Spain*
[3]*University of the Basque Country, Paseo Manuel de Lardizabal 1, 20018 Donostia / San Sebastián, Spain*
{*aarbelaiz, amoreno, lkabongo, hdiez*}*@vicomtech.org, alex.galonso@ehu.es*

Keywords: Volume Rendering, Medical imaging, DICOM, X3DOM, Ubiquitous platforms, WebGL, WebVR.

Abstract: Recently the possibility to visualize interactively volumetric datasets in the Web has opened new methods of exploration and sharing of 3D images coming from different fields. At the same time, VR technologies are gaining momentum in the society, where several HMD's are ready to be bought. This paper presents how volumetric datasets represented as DICOM images can be loaded and visualized interactively in a WebVR compatible setup. DICOM images are loaded from local or remote repositories into X3D volume rendering nodes, which are displayed in the VR devices using WebVR technology. The results show that WebVR and X3D are compatible web technologies that can be joined together to provide easy and extensible tools to interact with DICOM datasets. Some enhancements for the interactive VR and non-VR experiences are presented.

1 Introduction

DICOM datasets are the *de-facto* interchange file format in the medical field. Volumetric datasets are naturally represented as sets of images, covering different phases of the medical procedures. Typically, CT and MRI scans are composed of 2D slices that can be visualized and analyzed with specialized software. These slices can be processed and inspected individually, but they provided more information if they are considered as a whole volumetric dataset.

The introduction of volume rendering techniques in desktop platforms was a milestone in the medical field. New methodologies and algorithms were implemented to take into account the 3D information embedded in the set of 2D slices that compose the scans.

In the last years, the web paradigm has introduced new visualization, sharing and interaction schema. Web technologies have been always a way to democratize the technological advances to the public, and therefore, new open standards were defined to harmonize the web ecosystem: The X3D (Web3DConsortium, 2014), WebGL (Khronos, 2016) and WebVR (WebVR, 2016) standards are relevant players in this standardization process.

X3D standard defines a set of nodes for volumetric datasets, but they do not materialize the precise functionality that the users require to interact with the volumetric information. X3DOM (Fraunhofer IGD, 2014) provides a WebGL based implementation of the X3D standard, and it can be used to visualize complex 3D scenes in the Web.

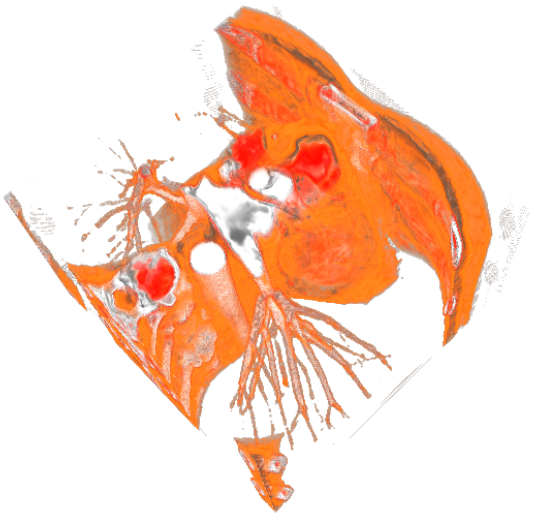

Figure 1: Volume rendering example showing the *AGE-CANONIX* volumetric dataset taken from the *OsiriX* repository (OsiriX, 2016).

```
<ImageTextureAtlas id="atlas">
  <canvas id="voxelCanvas"></canvas>
</ImageTextureAtlas>
```

```
<ImageTextureAtlas id="atlas" url="data:"></ImageTextureAtlas>
<script>
document.getElementById("atlas").setAttribute(
  'url',
  document.getElementById("voxelCanvas").toDataURL()
);
</script>
```

Figure 2: Integration mechanisms devised to transfer the DICOM content into the X3DOM framework. Left) direct utilisation of a *canvas*. Right) load mechanism using *toDataURL* function.

The introduction of new VR devices provides new means to create virtual experiences for the users. Typically, state-of-the-art gaming companies ship their standalone products, fully tested to provide the best immersive experiences. This kind of developments are very oriented to the specific product and very costly to implement, relying in the commercial success to recover the investment.

On the other hand, web technologies provide a rich multimedia ecosystem to anyone in the world with access to the Internet: text, audio, video and interactive 3D thanks to WebGL-based technologies. WebVR is the next step in this technological chain, aiming to provide support to the current head-mounted displays (HMD) in the market through the Web.

We identify web technologies as a key to give easy access to volume visualizations for both specialists and non-specialists in different scientific fields. The incorporation of the latest generation of Web APIs like WebVR has opened new technological challenges. This work presents how volumetric datasets, represented as DICOM files and stored locally or remotely, can be interactively visualized in web applications composed of the latest web technologies (see Fig. 1). The preliminary research findings show that this combination enhances the user experience in the interaction with volumetric datasets, hiding the development complexity under a clean and clear declarative X3D scene.

This paper is structured as follows. The next section provides a summary of the related work. Sect. 3 presents the work carried out to provide a combination of volume rendering on the web with WebVR using the X3DOM framework. Finally, this paper concludes with the conclusions and future work.

2 Related work

This section is divided into two parts: the first one describes previous work regarding the volume rendering raycasting algorithm and the second one describes how the DICOM datasets can be loaded using JavaScript libraries.

2.1 Volume rendering algorithm

Volumetric visualization has been extensively studied over the years. Nowadays it is used on a variety of fields, but especially in medicine. Different volume visualization techniques can be found in the literature; our work focuses in ray-casting. Originally introduced by Kajiya and Herzen (Kajiya and Von Herzen, 1984), ray-casting method was later translated to the GPU by Kruger and Westermann (Kruger and Westermann, 2003).

Our work is integrated in the X3DOM framework under the volume rendering component (Arbelaiz et al., 2016). Volumetric rendering is performed with a single-pass ray-casting algorithm. In a nutshell, in the scene a 3D unit cube is rendered and it will be used to place the actual volume data during the ray-casting. Using the programmable pipeline provided by WebGL, a single pair of vertex and fragment shaders is used. In the vertex shader, each vertex position of the cube is multiplied by the `ModelView` matrix and the `Projection` matrix, transforming the vertices into clip space.

In the fragment shader, the ray traversal is actually computed. When the triangles of the cube are rasterized into fragments, the interpolated vertices of the unitary cube represent 3D texture coordinates. From the inverse `ModelView` matrix the camera position can be obtained. By subtracting the interpolated vertex position with the camera position the ray direction can be determined. Taking both the interpolated position as the ray origin and the ray direction, using a fixed length loop statement in the fragment shader the ray traversal is created. For an in depth description of the single-pass approach, we refer the reader to Mobeen et al. (Mobeen and Feng, 2012).

The ray traversal is discretized into a series of steps. At each step the position of the ray is used as a 3D texture coordinates to fetch the volume data. However, WebGL does not support 3D textures, as first stated by Congote et al. (Congote et al., 2011) this can be overcome using a texture atlas (*ImageTex-*

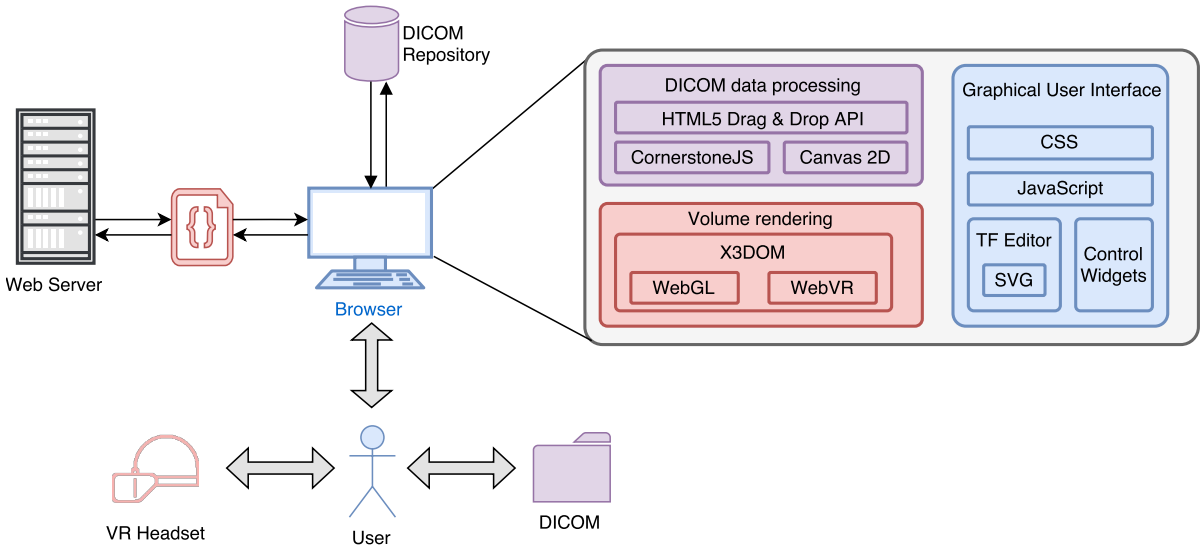

Figure 3: Architecture and modules of the web application. DICOM images are stored in a local or remote repository. The web application is loaded from Internet and presented to the user in non-VR mode. The web application process the DICOM dataset to create the *ImageTextureAtlas* and prepare the X3D scene. WebVR is used to display the VR mode to the users.

tureAtlas): the cross sectional slices that compose the volume are tiled into a matrix configuration to compose a single 2D texture.

2.2 DICOM Datasets Visualization

Volumetric data is often used in the medical field in a large variety of situations: from research and diagnosis to educational purposes. In terms of visualization interactivity and usability, mobile platforms should provide the same tools as their desktop counterpart.

In pursuit of a ubiquitous medical volumetric visualization, the support of DICOM file format is necessary as DICOM is the standard *de-facto* that the software of the medical imaging devices uses to store their scans.

Medical imaging devices do not only produce the actual set of 2D slices. They are linked with a large amount of metadata related to the patient health information and other medical procedures. DICOM is the medical image standard to store and transfer all this information from and between imaging devices and medical image storage repositories (Fernandez-Bayó et al., 2000). The wide utilization of DICOM by all manufacturers had a major impact on usability of the file format. Resulting sometimes in a variable set of tags to be read, interpreted and combined in order to achieve coherent restitution of the images for the final user.

Cornerstone JavaScript library (Cornerstone, 2016) provides a set of functions to read and interact with the 2D set of slices stored in DICOM files and it relies on *dicomParser* to load DICOM tags, including pixel data.

The combination of *dicomParser* with the volume rendering nodes defined in a X3D scene provides a general and ubiquitous solution to the problem: the slices pixel data is extracted from the DICOM file and then, a texture atlas is created and linked to the required texture field of the VolumeData X3D node.

Fig. 2 shows two ways to accomplish the information transformation from DICOM files to the X3DOM framework. The first one is to use a `<canvas>` node defined inside the *ImageTextureAtlas* node and then, using JavaScript, fill the canvas with a texture atlas.

The second integration method is to use the same auxiliary `<canvas>` node but defined out of the *ImageTextureAtlas* node. The *VolumeData* node uses an empty *ImageTextureAtlas* (no real URL is given). The JavaScript loader draws the atlas texture in this canvas as before, and then, the correct URL is provided as a *DataURL*, which means that the whole volumetric information is encoded and passed in the URL.

3 Web technologies architecture

This work presents how volumetric DICOM datasets composed of 2D slices can be interactively visualized in modern desktop browsers using exclusively open web technologies. The preliminary results provide an interactive VR user experience by presenting an adequate web interface that interacts dynamically with the volume rendering process.

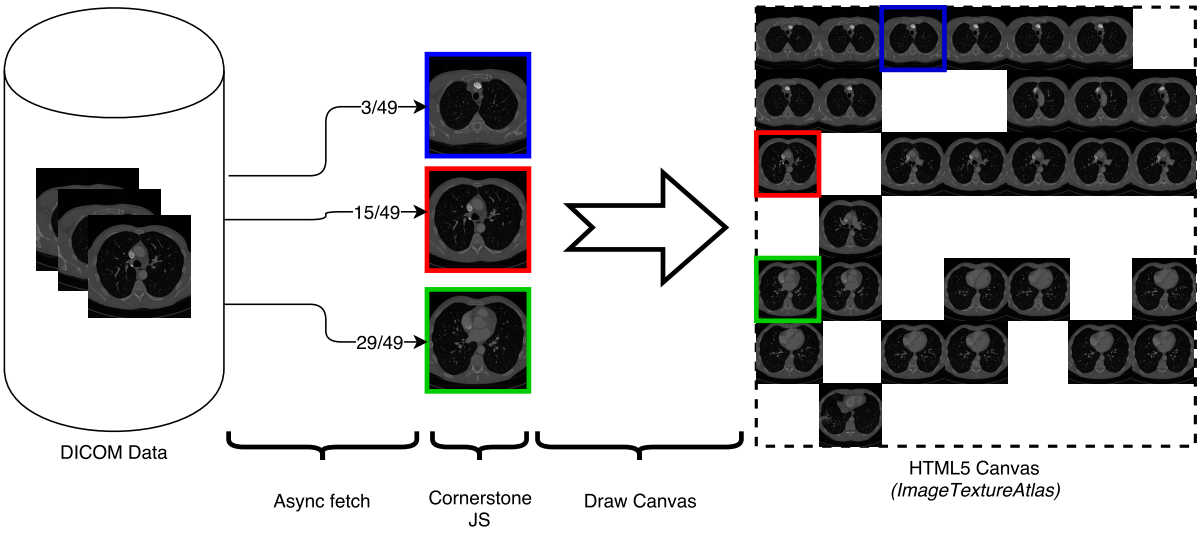

Figure 4: Asynchronous loading process from DICOM images to *ImageTextureAtlas*. Image shows an intermediate state where the dataset is not fully loaded and hence, some subtiles are not filled into the final HTML5 *canvas*.

In this section, we describe the architecture (see Fig. 3), modules and their relationships among the selected open web technologies. In our work, the current stack of Web technologies are WebGL, WebVR, X3DOM, JavaScript, SVG and CSS. These tools provide us with necessary base technology to develop web VR and interactive applications to the users.

Next Subsection 3.1 introduces the technological approach to load DICOM datasets into data structures ready to be used with WebGL and X3DOM. Subsection 3.2 presents some interactive modifications that can be performed to the original DICOM datasets in order to select the *Window Level*. Subsection 3.3 shows our interactive Transfer Function editor, capable of fine tuning the rendering output of the volume rendering algorithms. Finally, Subsection 3.4 presents how volumetric datasets can be visualized from the inside of the volume, instead of the typical visualization from outside the bounding box of the dataset.

3.1 Asynchronous atlas generation

The construction of the texture atlas is a technical constraint of the Web based volume rendering algorithm. An integration of the native Drag and Drop HTML5 API into our web application provides an easy and transparent approach for the generation of the required *ImageTextureAtlas* (see Fig. 4). Without this functionality, the atlas creation must be performed at the server in a pre-processing step. Then, it can be transfered to the client device as an *ImageTextureAtlas* image. With the proposed drag and drop functionality we can avoid this step .

The DICOM data files can contain the volume data in a single file, whereas, usually each DICOM file represents a single slice with the additional metadata information. As stated in Sect. 2.2 using the *dicomParser* open source library we can parse the volume data and compose the atlas in the background. In our prototype, the rendering canvas is equipped with drag and drop functionality. The user can drag a set of DICOM files hosted in their own device filesystem and drop them in the rendering canvas. The received files will be ordered by filename and composed into an atlas which is rendered in a hidden HTML5 2D *canvas*.

This feature is specially interesting, because it does not enforce users to adapt the volume data files to a certain file format. As a consequence, it makes it compatible with other application that output their results as DICOM or other browser compatible common image formats, such as: JPG or PNG.

3.2 DICOM metadata

DICOM files contain rich information that can be used in the UI presentation. Some metadata attributes can be added to the UI (capture date, acquisition machine, software version...) and others give information about the dataset itself, i.e., the number of slices and their resolution go normally with the pixel resolution (8, 10, 12 or more bits per channel). Taking into account that the display devices we are targeting can only display RGBA (8 bits per channel), a region of interest is defined with the aid of *window width* and

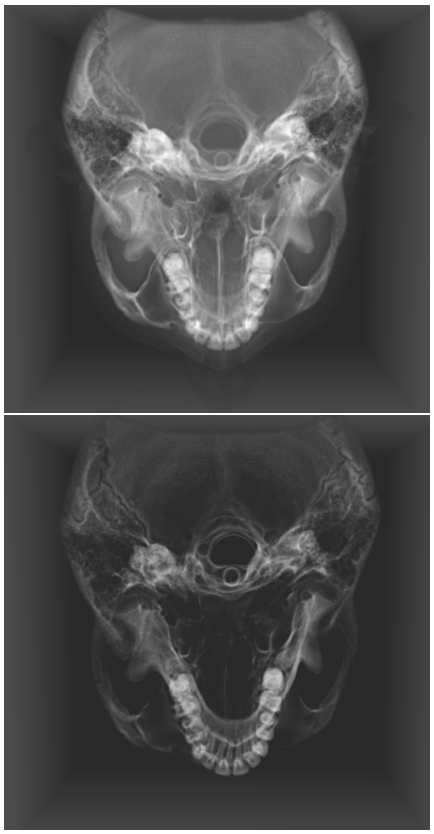

Figure 5: *INCISIX* dataset from the *OsiriX* repository (OsiriX, 2016) with different window levels. On the top *window center*= 395 and *window width*= 2068. On the bottom, *window center*= 1056 and *window width*= 1489.

window center levels from the DICOM data.

These values specify a linear conversion from stored pixel values in the DICOM file to values to be displayed on the screen. To change these levels, we provide dynamic sliders to the users. The user can manually discard or extend the desired image data range by modifying the *window width* and *window center* levels. As a result, the contrast between structures within the volume can be adjusted. Fig. 5 shows the *INCISIX* dataset from *OsiriX* library (OsiriX, 2016) with different window levels.

3.3 Interactive transfer function

In volume rendering, applying a transfer function (TF) is one of the most common techniques to illustrate a volume. A TF is a lookup function that maps each scalar value of the dataset [0-255] with a given color and opacity. Typically, a set of predefined and general TF can be used (from red to green to blue, rainbow schema...) but it is very common to find domain-specific TF's (like in the weather radar information) that are widely accepted by the experts of that domain. Additionally, it is quite common to offer the possibility to load a predefined TF, but with the possibility of customizing it interactively.

In our web application, we have developed a web TF editor based on the scalable vector graphics (SVG) technology. In Fig. 6 the TF editor can be seen at the bottom of the captures. In this interactive chart, an histogram of the volume data values is plotted. Then the user can add control points and assign a color to each of them. Colors will be interpolated between control points and the opacity will be calculated in function of the control points height. As a result of the user interaction, a 255 pixel width 1D RGBA image is generated. This image is directly passed as a texture input to the GPU and therefore, any changes in the TF editor have an immediate effect in the volume rendering visual output.

3.4 Inside exploration of volume data

Our work aims to the interactive visualization under VR technologies, where the user can perceive the stereoscopic effect of the 3D scene. In other words, the user can perceive distances and what is near and far in the volumetric information. During the inspection of the dataset, the user can move freely around the dataset, and therefore, the dataset can be located further or closer to the user. When the user gets closer to the volumetric object, it would be very intrusive for the virtual experience if the object suddenly disappears. As the volume rendering algorithms render the output on the surface of a 3D cube, if the user enters inside this cube, nothing would be seen.

Some modifications of the raycasting algorithms have been developed to allow correct visualizations of the volumetric dataset even if the user enters the 3D cube. An inside exploration allows the user to easily discern the internal composition of the volume rather than looking at the cross-sectional 2D images.

In our implementation we have allowed the exploration of the volume by dynamically changing the initial position of the ray origin (see Fig. 7). First the camera position is obtained from the inverse `ModelView` matrix. Then using the maximum and minimum boundaries of the cube, it can be determined whether or not the camera is inside the volume. If the camera is outside the cube, the ray origin is assigned as the interpolated vertex position from the output of the vertex shader (`varying vertex` position), if not, the ray origin is the camera position.

Without changing the ray origin, the camera face of the cube will be clipped, making it impossible to examine the inside. Thus, back face culling must be

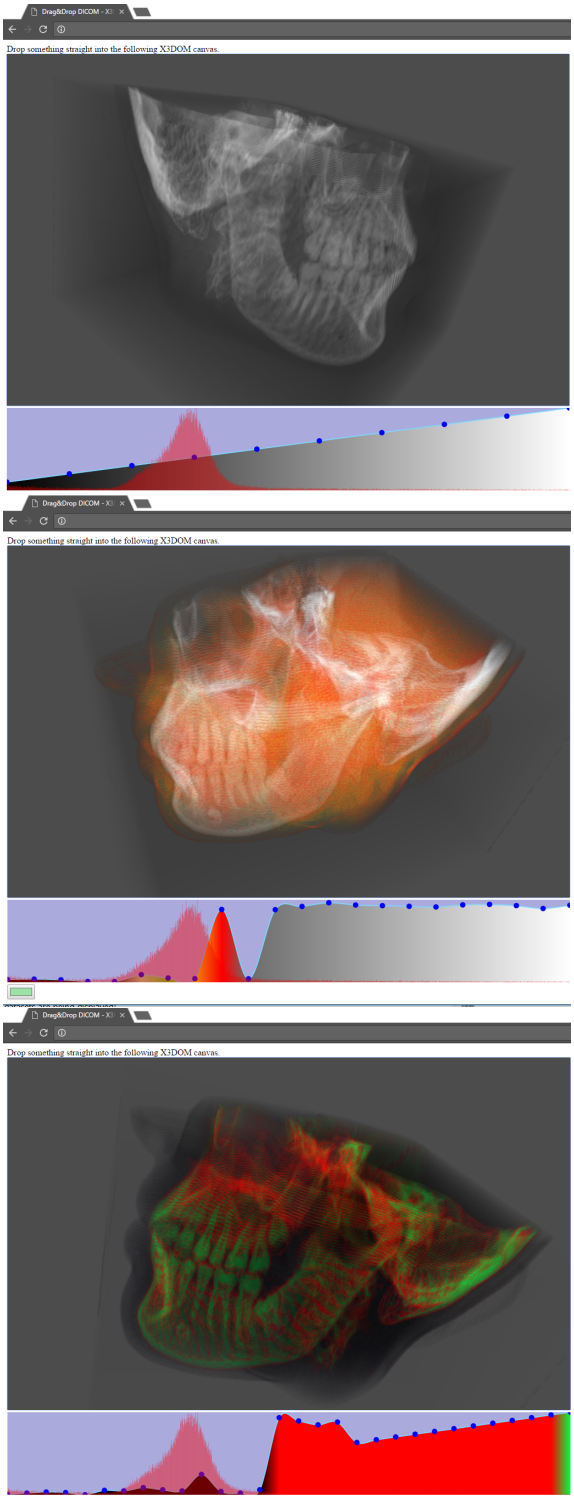

Figure 6: Volume rendering web application interface prototype in non-VR mode.

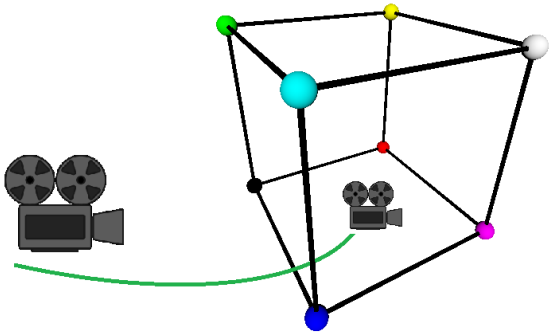

Figure 7: The visualization from the inside of the dataset requires that the user moves the virtual camera location into the cube that holds the volumetric dataset. *Zoom* functionality can be triggered with the wheel mouse.

disabled when internal exploration is required. When the camera is moved into the cube, the ray-casting direction for each ray is obtained by subtracting the interpolated back face vertex position with the camera position.

3.5 WebVR interactive application

The creation of new VR oriented content with the WebVR (WebVR, 2016) framework is simplified to the extent that only knowledge of HTML, JavaScript, CSS and related web technologies are required to develop VR experiences. But even with this simplicity, teachers, researchers and other non-technological-aware target groups are not prepared for such a task.

To solve this issue, X3DOM provides a WebGL implementation of the X3D representation. The declarative nature of X3D scene suits better for most of the people who want to deploy VR experiences but lack the technological capabilities of doing everything by themselves. We have combined our proposed medical functionalities implemented in the X3DOM framework and test them along with WebVR.

Our preliminary demonstrations have been tested with the Oculus Rift DK2, under Windows 10, with the 1.5 and 1.6 runtimes. A developer build of the Chrome browser with WebVR support (Chrome, 2016) has been used to test the VR developments.

The technical limitations regarding this setup will be fully overcome when the WebVR 1.0 implementations of the standard reaches the release version of the most common browsers: Firefox, Chrome, Safari and Edge. Additionally, it is expected that mobile versions (for Android and iOS) would be available with WebVR support.

Once the setup is running and a X3D volume rendering scene is declared, loading a VR experience is as easy as loading a URL. Then the VR experience is

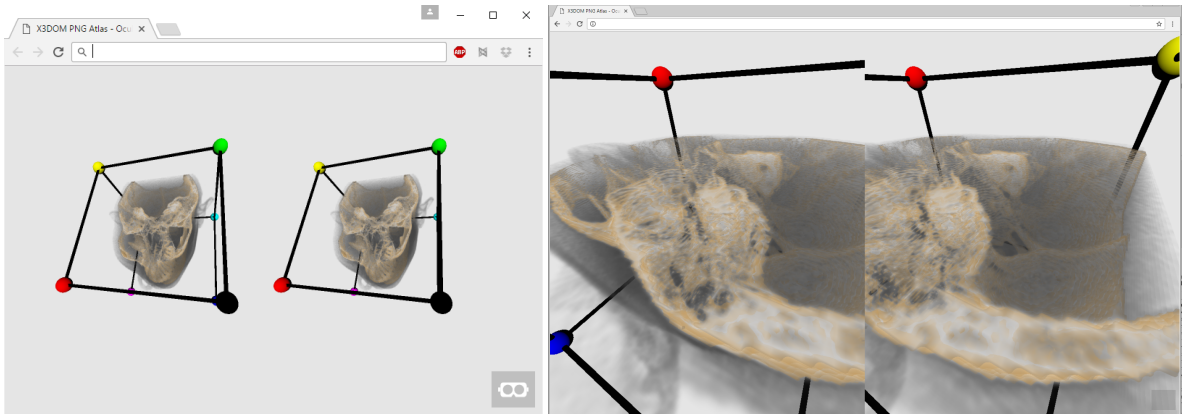

Figure 8: Web based volume rendering in a VR scene with WebVR. Using the *INCISIX* dataset from the *OsiriX* repository (OsiriX, 2016). Left: inspection from a far point of view. Right: Visualization from a closer point of view. The wireframe cube and the solid colored spheres help the users to know their location at any time.

started by clicking in a *Enter VR* button or the *white goggles* icon, see Fig. 8.

3.6 VR experience enhancements

One of the key elements in the success of the VR is to control how the VR experiences are perceived by the users. In this regard, the transitions from non-VR to VR and backwards should be controlled. In addition to this, any sudden change in the scene should be carefully treated to avoid intrusive pop-up effect in the user field of view.

In our prototypes, DICOM files (through Cornerstone JavaScript library) are loaded asynchronously, i.e, each slice of the volumetric dataset is retrieved as soon as it is loaded by the library. Therefore, initially the 3D scene could be empty, which is something to avoid in order to reduce stress to the users: being *floating* in empty space is against ergonomic rules regarding VR (Rebenitsch, 2015).

To solve this potential situation, the empty 3D cube where the volume will be loaded is surrounded with a 3D wireframe with colored solid spheres in the corners of the cube (see Fig. 8). This solution offers two benefits: *i)* the scene is not empty while the dataset is being loaded and *ii)* the colored spheres give visual clues of the orientation of the scene that can be used in case the users got lost. In our tests, this 3D visual clue has been proved very helpful in VR environment, but also in non-VR setups.

The HMD's like Oculus Rift are more focused in immersive environments where the user is transported to a virtual world that can be navigated and explored in *first person*. In our case, the typical navigation and exploration of the volumetric dataset is more compatible with the *orbit* navigation style, which conflicts somehow with the nature of the VR. We refer the reader to this collection of publications (Christie and Olivier, 2009) to get an insight of the different camera styles in virtual environments.

Our solution to cope with this situation was to map the VR interaction capabilities to the actions that should be carried out when the volumetric datasets are being displayed:

- Mouse: Typically in VR, the mouse rotates the user in the world. In our case, the mouse rotates the 3D volumetric dataset around its center, like in the *orbit* navigation style.

- Arrow Keys: Typically in VR, the keys allow to move freely in the virtual scene (in walking or flying mode). In our case, the keys have been disabled. In order to get closer to the volume or to inspect it from the inside, the wheel mouse is used to *zoom* into the scene.

- Head tracking: The modern HMD's provide information about where the user is looking at. We kept this functionality as it would be very intrusive to remove this functionality.

- Information display: In VR mode we remove all the UI elements (TF editor, window level management...) in order to provide a clean VR experience. Only the *Exit VR* button is present. In preliminary tests we added another button to reset the viewpoint in case the user got lost, but ultimately, it was discarded. It was easier for the user to get out VR mode in that extreme case.

The resulting VR experience allows to inspect the volumetric dataset from any point in an easy and straightforward manner.

4 Conclusions

This work has introduced our preliminary efforts to bring VR and non-VR volumetric visualizations on the Web by using a combination of open web technologies: WebGL, WebVR and X3D. DICOM files are supported through the Cornerstone JavaScript library, providing a custom piece of code to construct a texture atlas in the client. A X3DOM implementation of the volume rendering nodes has been used to render the texture atlas.

Some UI elements and visual clues have been added to help the users to interact with the volumetric information (*Window Level* modification, interactive Transfer Function editor, 3D wireframe, visualization from inside the volume and VR compatible *orbit* navigation style). Our preliminary research activities have dealt with situations regarding the VR and non-VR environments and the transitions among them. Our results show that there is room to improve the Human-Computer Interaction within the web environment and the current and forthcoming collection of HMD devices: Oculus Rift, HTC Vive, Microsoft Hololens, Samsung GearVR, PlayStation VR, Google Cardboard and Daydream...

The presented mixture of web technologies provide a real ecosystem that can facilitate the deployment of VR experiences of volumetric datasets for experts in their corresponding fields, but unaware of the technological advances under the hood. This situation will democratize the access to the information and the distribution and sharing of volumetric visualization among different user groups.

Our next steps will be oriented to the further testing of UI elements with real users through subjective surveys. Additionally, we will explore the possibility to add 3D interaction widgets into VR mode through the combination of gesture based recognition (using devices like Leap Motion (Leap Motion, 2016), see Fig. 9) and novel navigation modes.

Figure 9: Leap Motion can be attached to the Oculus Rift in order to provide gesture recognition of the user's hands.

REFERENCES

Arbelaiz, A., Moreno, A., Kabongo, L., and García-Alonso, A. (2016). X3DOM volume rendering component for web content developers. *Multimedia Tools and Applications*, pages 1–30.

Christie, M. and Olivier, P. (2009). Camera Control in Computer Graphics: Models, Techniques and Applications. In *ACM SIGGRAPH ASIA 2009 Courses*, SIGGRAPH ASIA '09, pages 3:1–3:197, New York, NY, USA. ACM.

Chrome (2016). Chrome developer builds with WebVR support. https://webvr.info/get-chrome/.

Congote, J., Segura, A., Kabongo, L., Moreno, A., Posada, J., and Ruiz, O. (2011). 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. ACM.

Cornerstone (2016). JavaScript library to display interactive medical images including but not limited to DICOM. https://github.com/chafey/cornerstone.

Fernandez-Bayó, J., Barbero, O., Rubies, C., Sentís, M., and Donoso, L. (2000). Distributing Medical Images with Internet Technologies: A DICOM Web Server and a DICOM Java Viewer. *RadioGraphics*, 20(2):581–590. PMID: 10715352.

Fraunhofer IGD (2014). X3DOM: Open-source framework and runtime for 3D graphics on the Web. http://www.x3dom.org.

Kajiya, J. T. and Von Herzen, B. P. (1984). Ray Tracing Volume Densities. *SIGGRAPH Comput. Graph.*, 18(3):165–174.

Khronos (2016). WebGL: OpenGL ES 2.0 for the Web. https://www.khronos.org/webgl/.

Kruger, J. and Westermann, R. (2003). Acceleration Techniques for GPU-based Volume Rendering. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 38–, Washington, DC, USA. IEEE Computer Society.

Leap Motion (2016). Truly immersive VR technology. https://www.leapmotion.com/.

Mobeen, M. M. and Feng, L. (2012). High-performance volume rendering on the ubiquitous webgl platform. In *Proceedings of the 2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, HPCC '12, pages 381–388, Washington, DC, USA. IEEE Computer Society.

OsiriX (2016). DICOM image sample sets. http://www.osirix-viewer.com/resources/dicom-image-library/.

Rebenitsch, L. (2015). Managing cybersickness in virtual reality. *XRDS*, 22(1):46–51.

Web3DConsortium (2014). Extensible 3D (X3D) basic example archives. http://www.web3d.org/x3d-resources/content/examples/Basic/VolumeRendering/.

WebVR (2016). Bringing virtual reality to the web. http://webvr.info/.