Virtual Reality Interactive Tool for 3D Scalar Field Rendering

Mai Trinh, Jiechang Guo

Abstract—Scientific data visualization provides aid to improve scientists' interpretation of large data sets. Displaying complex data in visual form with a clear and intuitive approach is an important task in enhancing the understanding and exploration of big data. Thus, this work presents a virtual reality application and interactive tool for 3D scientific data visualization in immersive environments. In particular, we implement several visualization techniques including direct volume rendering, iso-surface and 3D cut-planes for 4 different scientific data sets and render it in a virtual reality headset. Natural and direct user interaction is integrated into the proposed system allowing the user to directly interaction with the 3D volume data.

Index Terms—Virtual Reality, Visualization, Volume Rendering, 3D Scalar Field, Scientific Visualization

1 Introduction

Data visualization has traditionally focused on 2D platform such as screen and paper. While many popular visualization toolkits such as ParaView [1] and VTK [2] have been used to create data visualization for large dataset, it still lacks the ability for direct and natural interaction. Data visualization in an immersive environment such as virtual reality (VR) addresses this issue. VR generates environment with scenes and objects that appear to be real, making users feel as if they are immersed in their surroundings. It allows for a new way to explore complex datasets using emerging display and interaction. Users can potentially walk into the data and study from every perspective. VR offers 360° view to display data. Thus, data visualization utilizing VR can offer insight that traditional technologies cannot.

In this work, we present a VR application and interactive tool for 3D scalar field visualization. We utilized the VR headset, Oculus Quest 2 (Meta Quest 2) as the main target platform. While wearing the VR headset, users can explore the rendered volume inside an immersive VR environment and interacts with the 3D volume with intuitive gestures. The contributions of this work are summarized as follows:

- Develop a data conversion of 3D volumetric data into texture for rendering
- Implement an Unity shader to perform direct volume rendering, iso-surface, and 3D cut-planes and render 3D scientific data visualization in VR
- Design and implement friendly user interface and intuitive, direct user interaction in an immersive VR environment

In the remainder of this paper, we outline existing toolkits for 3D scientific data visualization in both traditional platform and immersive environment (Section 2). The detailed description of our methodology and implementation are provided in Section 3. Additionally we also report on the contribution of each team member. Section 4 shows the results as well as challenges/limitations of our current application. Possible improvement and extension fo the work are outlined in Section 5. Finally, we conclude the report in Section 6.

• *M. Trinh and J. Guo are with the Department of Computer Science, University of Houston, Houston, TX, 77004.*
E-mail: mtrinh2@uh.edu, guo23@uh.edu

2 Related Work

2.1 Scientific Visualization

Scientific visualization is the process of graphically illustrate real or simulated scientific data to facilitate the understanding and exploring data insights. Primary methods for visualizing 3D scientific data include 3D cut planes, iso-surfaces and volume rendering. Typically, volumetric data comprises of multiple 2D image slices captured by a CT, MRI, or Xray machine. Volume rendering maps 3D volumetric data to an 2D image by mapping voxel values to a location in data space. Popular applications such as ParaView [1] and VTK [2] provides fully integrated tools for manipulating and displaying scientific data. However, such applications support traditional 2D environment and utilizes monitor, mouse or keyboard as user input which poses intrinsic limitations when exploring 3D volumes. The dimensional restriction in 2D interface poses significant drawbacks in manipulating and perceiving 3D volumes and drastically hinders user experience.

2.2 Visualization in Immersive Environments

Virtual reality environments has been shown to be an effective approach for enhancing visualization to explore and understand 3D data. VR systems allow users to explore the spatial surrounding freely thus motivate users to explore volumetric data in new ways. Cordeil *et al.* introduces an immersive analytic toolkit for multidimensional, information data visualization in immersive environments [3]. Usher *et al.* presents a VR system to trace neurons in microscope scans and allow users to interact with neurons directly in 3D [4]. O'Leary *et al.* integrates the state-of-the-art visualization toolkit VTK in a virtual reality environment with near real-time updates and efficient interaction [5]. Given its ability

for a natural and direct interaction, our application aims to visualizing 3D scalar field data using volume rendering, iso-surface and 3D cut-planes in an immersive VR environment.

3 Method & Implementation

This section describes in detail the proposed methodology and implementation details of our application. The implementation of visualization techniques is adapted from the work of [6].

3.1 System Setup

Our system is designed for Oculus Quest 2 as the target implementation platform and powered by a Windows personal computer with Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz and NVIDIA GeForce GTX 1060 6GB via Meta Quest Link and USB 3.0 connector. The application is developed using C# and Unity3D version 2021.3.

3.2 Data Preparation

3.2.1 3D Voxel Data

3D volumetric data uses 3D voxel data. Unlike traditional computer graphics that uses surface data to represent a model in 3D, a voxel data is more focus on the internal part of the model. A volume which is a uniform 3D array stores data value in voxels. As shown in Fig. 1, 3D volumetric data is a stack of images or slices (typically taken from CT scan, MRI, or XRay image) with 2D arrays of data values represented by each pixel [7].

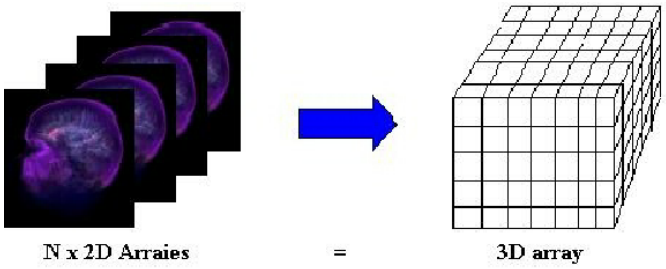


Fig. 1. 3D Volume Data Representation, illustrated by J. Zhou and K. Tönnies [7]

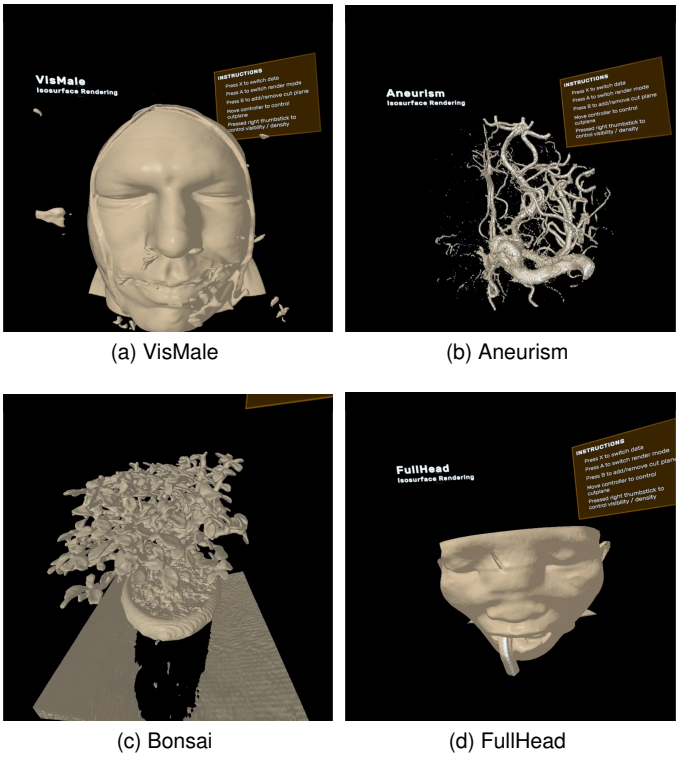
3.2.2 Data Processing

Data Format. To show the proof of concept, we outsource 4 different data sets taken from [6], [8]. Each data set contains two files including ".raw" and ".ini". For ".ini" file it records useful information such as the dimension of the data and the format it is stored, in order to read binary data from ".raw". Each ".raw" file stores an array of density value. The configuration of each data set used in our application is shown in Table 1. Fig. 2 shows example of each data set.

Data Conversion. In this project, a convert script is provided for user to convert the data sets used in vtk including ".mhd" and ".vtk" to the appropriate data format required by the system. The pseudo-code for converting vtk data file to raw format is outlined in Algorithm. 1. Once the raw data is loaded in the system, it is saved as 3D texture and prepared to be processed in Unity shader. Each pixel value is the normalized density value from 0 to 1. In order to enable the light effect, a gradient texture is computed during the data processing step. The data texture and gradient texture are passed to the shader when rendering a cube, which is the container for the volume data.

TABLE 1
Data Configuration

Data	dimx	dimy	dimz	skip	format	size(KB)
FullHead	256	256	94	0	int32	24064
VisMale	128	256	256	0	uint8	8192
Aneurism	256	256	256	0	uint8	16384
Bonsai	256	256	256	0	uint8	16384



(a) VisMale (b) Aneurism

(c) Bonsai (d) FullHead

Fig. 2. Screenshot of each data used in our application as proof of concept.

Algorithm 1 Converting vtk data file to raw format

```
Read data using vtkReader
get scalars array from point data
get range of the scalar field
get the dimension of the data
create .ini file and .raw file
while .init file is open do
    write dimension to file and set format as Int32
end while
while .raw file is open do
    for each s in range of scalar values do
        write s as binary to the raw file
    end for
end while
```

3.3 Visualization Techniques

The visualization algorithm was adapted from [6] with major modifications to make our system applicable for rendering and interacting in virtual reality environment.

3.3.1 Direct Volume Rendering

Direct volume rendering (DVR) is a process that maps 3D volumetric data to 2D image without extra computation (i.e. intermediate geometry). There are several ways to compute direct volume rendering including ray-casting, splatting and texture-mapping. For this project, we focus on the ray-casting method.

Raycasting. Raycasting is a fundamental method for generating DVR. An illustration of the ray-casting technique is shown in Fig. 3. For each pixel, it cast a ray with direction from the camera, uniformly sample along the ray with given maximum steps, for each sample get the density value from 3D texture and finally do color composition based on the density and different color composition techniques.

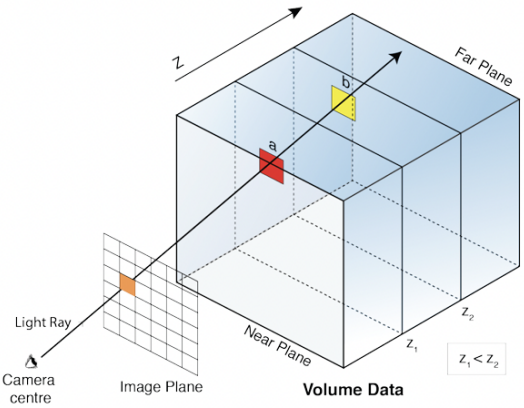


Fig. 3. Raycasting illustration by Kalshetti *et al.* [9]

The process to practically implementing direct volume rendering in Unity3D is described as follows:

- Create a cube in the scene, the cube will be the container for a volume.
- Then the cube will be rendered following a rendering pipeline [10], as shown in Fig. 4. The color for each pixel in the screen is decided during fragment shader where we get the density value from data texture and calculate light from gradient texture.

The pseudo-code for a general ray-casting method is outlined in Algorithm. 2. The color of the fragment is decided based on different color composition strategies (alpha composition, local maximum intensity projection and maximum intensity project) according to its density and gradient. This variation is specified in line 14, and will be discussed in detail in the later section.

3.3.2 Color composition

Maximum Intensity Projection. One of the simple and fast color composition technique is maximum intensity projection (MIP) which use the highest density of sampled data along the ray. This technique typically highlights the component with a higher density like teeth and bone (Fig. 5a).

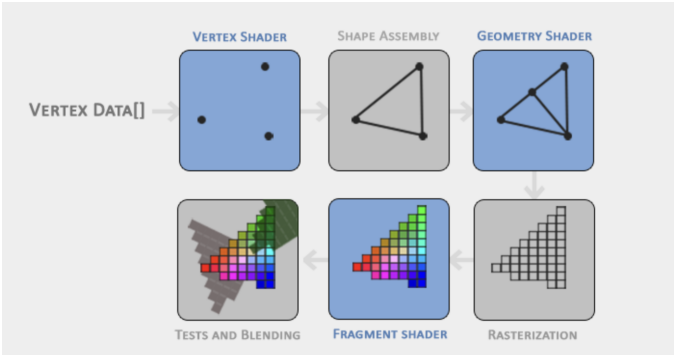


Fig. 4. Rendering pipeline illustrated by [10]. The volume rendering is processed in fragment shader.

Algorithm 2 General Raycasting in fragment shader

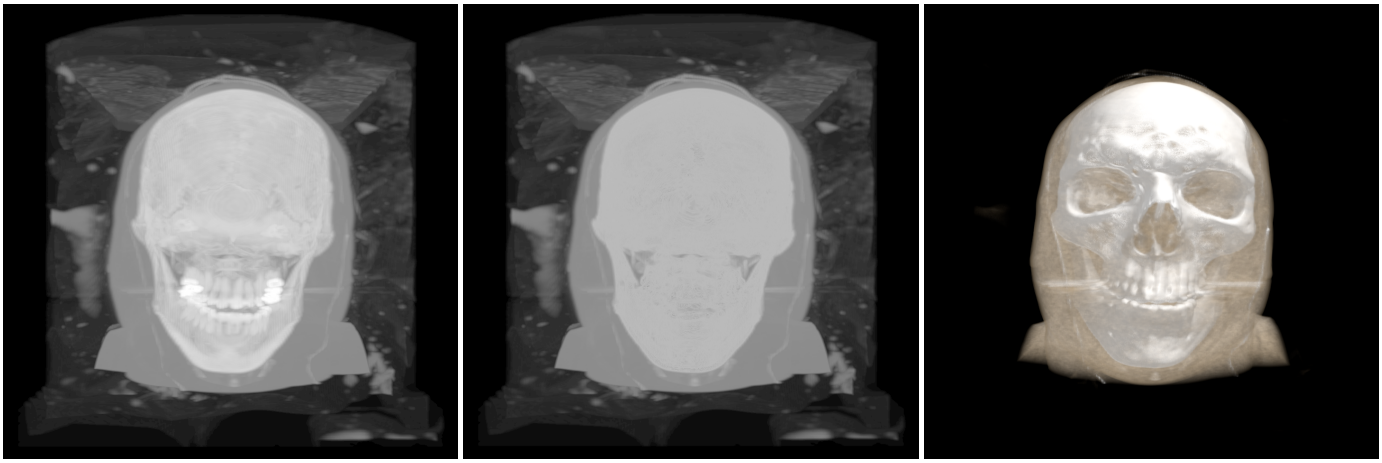
```
1:  for each fragment do
2:      init ray along view direction with vertex local position
3:      sampling along the ray
4:      init color for this fragment
5:      for each step along the ray do
6:          if current position is cut out by cut plane then
7:              continue
8:          end if
9:          get density of current position from data texture
10:         if density is out of the visibility window then
11:             continue
12:         end if
13:         get gradient of current position from gradient texture
14:         use density and gradient to decide fragment's final color
15:     end for
16: end for
```

Local Maximum Intensity Projection. Another type of color composition strategies is local maximum intensity projection (LMIP). LMIP technique is faster than MIP. It will use a density equal to or higher than an user-selected threshold to do the final color composition (Fig. 5b).

Alpha Composition. A more sophisticated method for color compositing is alpha composition (Fig. 5c). This technique accumulated the color value of the current voxel with the color from the previous voxel. A linear interpolation technique was used here to interpolate the RGB color based on the current alpha value. A new alpha value is also computed based on the previous alpha value. The update equations for RGB color and alpha value are as follows:

$$color.rgb = curr.rgb + (1.0 - curr.a) * prev.rgb \quad (1)$$

$$color.a = curr.a + (1.0 - curr.a) * prev.a \quad (2)$$



(a) MIP composition (b) LMIP composition (c) Alpha composition

Fig. 5. Direct Volume Rendering with different composition techniques on male head data. DVR using alpha composition strategy shows the best result with clear partition of features such as bones and teeth.

3.3.3 Iso-surface Construction

In traditional approaches such as VTK, iso-surface construction uses the marching cube method, which is time-consuming and computationally expensive. For our application, the implementation of iso-surface rendering is done by simply drawing the first voxel data with a density higher or equal to the user-defined iso-value. One problem with this method is when there is noise in the data, there will be small mesh components. To achieve a better visual effect, we enable lighting on the surface, that is the diffuse light reflection on the surface. The normal for calculating light reflection is stored in the gradient texture. As mentioned previously, the gradient texture is computed during the data preparation process. An example of iso-surface rendering is shown in Fig. 6.

3.3.4 Cut-planes

Cut-plane is basically a 2D plane highlighting a certain plane section of the volume data. To enable direct interaction in this system, cut-planes are implemented for users to control and observe the inside of the volume. This process consists of 2 steps. First, we must specify the plane position and orientation. Second, once we have the plane normal, all of the voxels that are in front of the user-controlled plane will be disabled. This cut-out test is being done for each voxel as shown in line 6 of Algorithm 2. For each frame, the global transform matrix of the plane is set to the shader of the volume container used to transform the vertex from local space to the plane's space, and then check if the vertex is in front of the plane. Examples of cut-plane are shown in Fig. 8.

3.3.5 Transfer function

Transfer function is used to emphasize features of interest in the data such as bones with high density are white and opaque while the skin with low density is brown and almost invisible. The transfer function for DVR is encoded in the transfer function for 2D texture with matrix of dimension 512 x 2. For each pixel in the image, the first row is the actual lookup table for color. Here we use the density

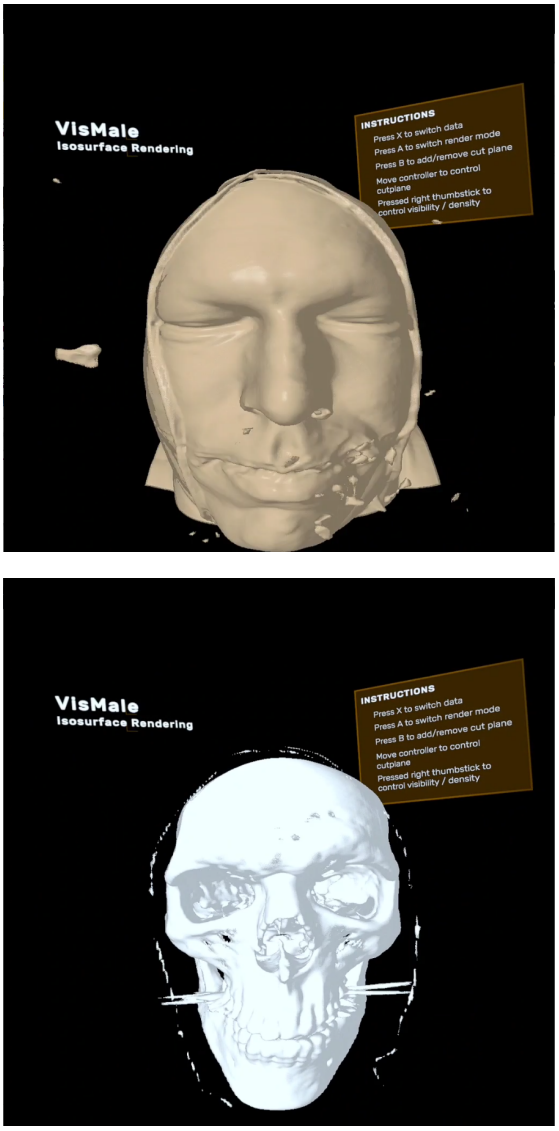


Fig. 6. Iso-surface rendering of VisMale data set using different iso-value.

value to sample the color and opacity from the transfer function texture. Since Unity does not support 1D texture, we need to add a second dummy row to satisfy the 2D texture requirement. The control points used to determine the transfer function are shown in Table 2. The color lookup table is then computed based on the control points.

TABLE 2
Color Control Points for Transfer function

Density	r	g	b
0.0	0.11	0.14	0.13
0.2415	0.469	0.354	0.223
0.3253	1.0	1.0	1.0

Density	alpha
0.0	0.0
0.1787	0.0
0.2	0.024
0.28	0.03
0.4	0.546
0.547	0.5266

3.4 User Interface and Interaction

For the implementation of the immersive environment, we employed the Unity 3D Game Engine developed by Unity Technologies [11] in combination with XR Interaction Toolkit [12] as it provides a full-featured graphical interface and easy-to-use APIs for controller input and UI interactions. VR interface is designed relatively simple for the moment, with the rendered 3D volume data displays in front of the viewing direction and an instruction panel specifying the controller input functionality. Two single-handed XR controllers are used to capture the user's input with the provided mapping as illustrated in Fig. 7. The controller input specifications are defined as:

- Press ***X*** to switch to different data set
- Press ***A*** to change rendering mode (i.e. DVR with alpha composition, DVR with MIP, DVR with LMIP, iso-surface)
- Press ***B*** to add or remove cut plane
- Press and move right thumbstick to control visibility (for DVR) or density (for iso-surface) value

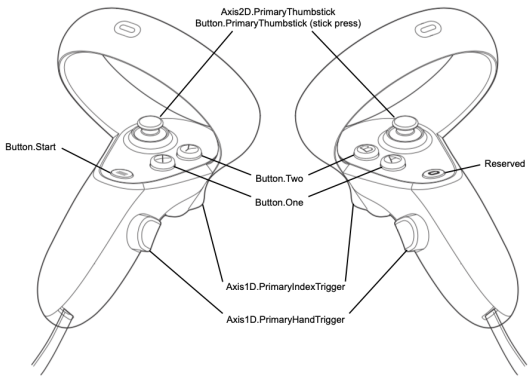
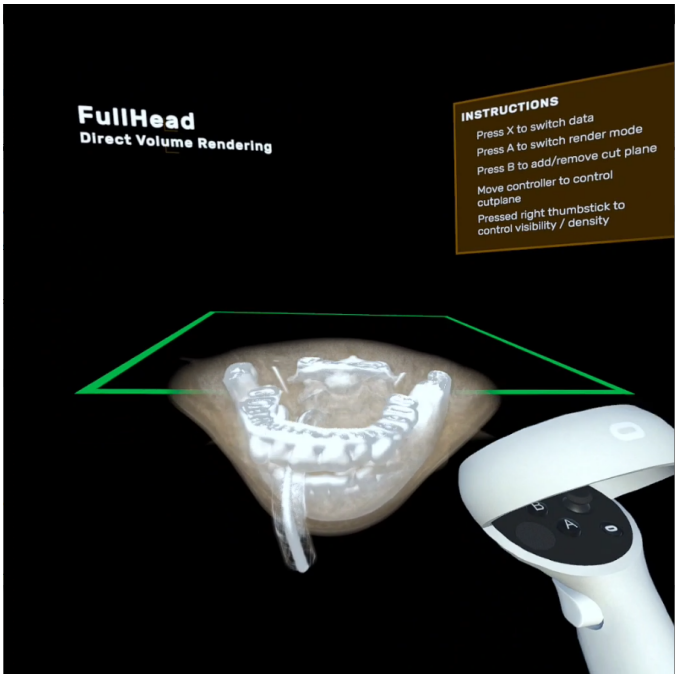
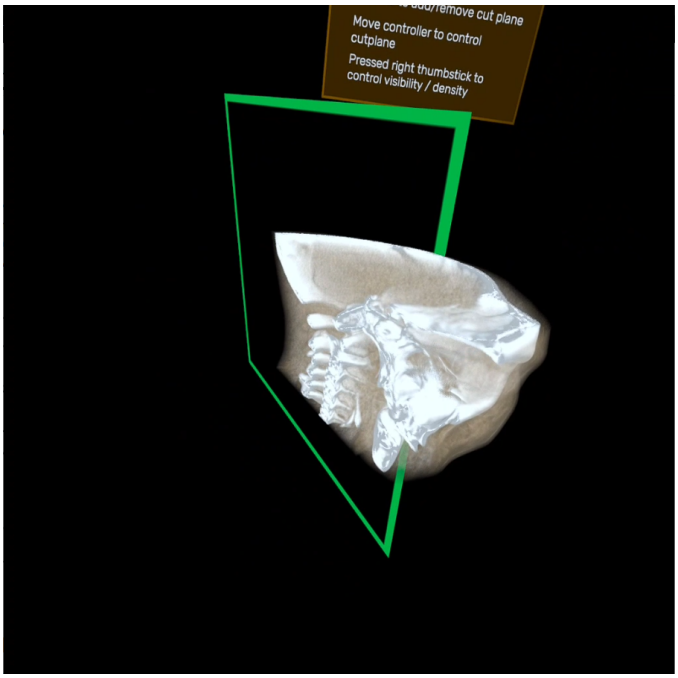


Fig. 7. VR Controllers virtual mapping (accessed as individual controllers), illustrated by [13]. Button.One of the left controller is mapped to X button. Button.One of the right controller is mapped to A button. Button.Two of the right controller is mapped to B button.

When enabled, the cut-plane is attached to the right controller, which means it can be moved and oriented using user's hand movement while holding the controller (Fig. 8).



(a) Front view



(b) Side view

Fig. 8. Simulation results for the VR environment while moving the controller to visualize the cut-plane.

In addition, the VR headset provides its own head tracking functionality with six degrees of freedom thus enabling the user to move around on a room scale without the need to set up additional sensors. We utilized this functionality to our own benefits through a 360° view of the rendered visualization by simply walking around the object or walking toward the object for closer observation.

3.5 Team Members and Contribution

TABLE 3
Member & Task Distribution

Member	Role
Jiechang Guo	Data processing
	Implement visualization techniques
	Debugging and testing of application
Mai Trinh	Set up development environment for Unity + Oculus
	Integrate Unity application into VR
	Implement user interaction and GUI design
	Debugging and testing of application

4 Results and Discussion

In comparison with traditional approach which uses indirect manipulation on 2D environment, our VR application offers a direct manipulatino of the object in 3D environment. Instead of using mouse and keyboard to input value and manipulate the field of view, we use the VR controller and headset to capture the user input and manipulate the viewing direction (Table. 4). Fig. 9 shows an example of visualization 3D scientific data using traditional approach on 2D environment. As we can see, the user must use a mouse to manipulate the viewing direction to be able to view different side of the visualization. Different checkbox options are used to enable different modes of rendering (such as iso-surface, cut-plane, DVR). The user selects different check box to specify which cut plane they want to view, and uses slider to visualize the different cut-plane. As for our VR application, we provide a more intuitive approach. The user can change the rendering mode using a single button on the controller. The cut-plane is manipulated based on user's hand movement while holding the controller. The manipulation of different viewing direction is done by simply wearing the VR headset and walking around the object.

TABLE 4
Comparison

	Traditional	Our VR Application
Interaction Type	Indirect manipulation	Simulated direct manipulation
Platform	2D environment (paper, monitor)	3D environment
User Input	Mouse, keyboard	Controller, VR Headset

Overall, our application shows a promising result while allowing user to interact with 3D scientific data in an immersive environment. Free from using mouse and keyboard, user is now able to explore the the data in new ways without any restriction (beside the spatial constraint of the physical environment). Manipulation and visualization of the data from every point of view become intuitive and realistic.

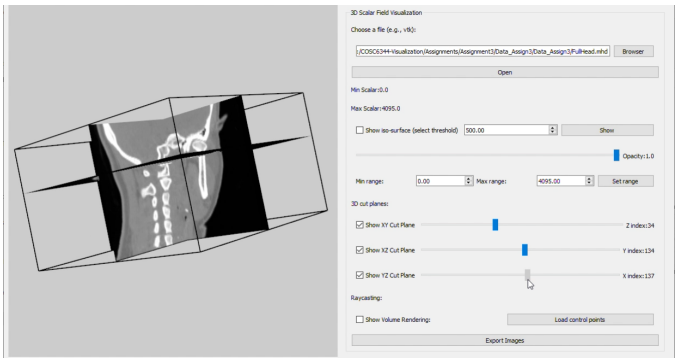


Fig. 9. Example of 3D scientific data visualization using traditional approach on 2D environment, taken from Assignment 3 of the Visualization course.

4.1 Limitation

While showing its promising result, limitation exists in our system. The challenges for direct volume rendering, in combination with virtual reality, is the required powerful hardware. DVR uses a computational expensive visualization algorithm, thus the current application utilizes GPU to achieve performance sufficient for rendering visualization. This condition leads to the requirement of external PC with GPU support. To efficiently develop a standalone VR solution that uses solely the VR headset, optimization needs to be made. One possible approach is to implement cache-aware algorithm on the VR headset's CPU.

5 Future Work

In the future, we would evaluate the feasibility of these features for the VR application with a target group of users such as medical students to further enhance user experience. Moreover, our current application uses a simple linear transfer function based on density to decide color and opacity. However, this can pose an issue when different parts of the data have similar densities, thus multidimensional transfer function would be needed to sufficiently highlight specific part of the data.

6 Conclusion

In this paper, we presented a virtual reality application and interactive tool for 3D scientific data visualization in immersive environments. Visualization techniques including direct volume rendering, iso-surface, and 3D cut-planes were integrated for 3D scalar field data. Simple interface with natural and direct user interaction were enabled to provide a full VR experience. In combination with the current imaging technology, virtual reality has shown its potential in aiding users' understanding of complex data.

References

[1] J. P. Ahrens, B. Geveci, and C. C. Law, "Paraview: An end-user tool for large-data visualization," in *The Visualization Handbook*, 2005.
[2] W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit–An Object-Oriented Approach To 3D Graphics*, 4th ed. Kitware, Inc., 2006.

[3] M. Cordeil, A. Cunningham, B. Bach, C. Hurter, B. H. Thomas, K. Marriott, and T. Dwyer, "Iatk: An immersive analytics toolkit," in *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2019, pp. 200–209.
[4] W. Usher, P. Klacansky, F. Federer, P.-T. Bremer, A. Knoll, J. Yarch, A. Angelucci, and V. Pascucci, "A virtual reality visualization tool for neuron tracing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 994–1003, 2018.
[5] P. O'Leary, S. Jhaveri, A. Chaudhary, W. Sherman, K. Martin, D. Lonie, E. Whiting, J. Money, and S. McKenzie, "Enhancements to vtk enabling scientific visualization in immersive environments," in *2017 IEEE Virtual Reality (VR)*, 2017, pp. 186–194.
[6] M. Lavik, "UnityVolumeRendering," 2021. [Online]. Available: https://github.com/mlavik1/UnityVolumeRendering
[7] J. Zhou and K. Tönnies, "State of the art for volume rendering," Institute for Simulation and Graphics, Magdeburg, Germany, Tech. Rep., 08 2003.
[8] L. Q. Campagnolo, "Volume Rendering Data," 2021. [Online]. Available: https://github.com/lquatrin/volume_rendering_data
[9] P. Kalshetti, P. Rahangdale, D. Jangra, M. Bundele, and C. Chattopadhyay, "Antara: An interactive 3d volume rendering and visualization framework," *CoRR*, vol. abs/1812.04233, 2018. [Online]. Available: http://arxiv.org/abs/1812.04233
[10] J. de Vries. Hello Triangle. https://learnopengl.com/Getting-started/Hello-Triangle.
[11] U. Technologies. Unity Engine. https://unity.com/.
[12] ——. XR Interaction Toolkit. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@1.0/manual/index.html
[13] M. Quest. Map Controllers. [Online]. Available: https://developer.oculus.com/documentation/unity/unity-ovrinput/