Real-time volume rendering of 3D data using semi-automatic transfer function

Shashank Chandgude(2304656)
Hemanth Nagapudi(2285874)

***Abstract*—Real-time volume rendering is a cutting-edge technique that swiftly generates and displays three-dimensional images from volumetric data. This technology finds widespread applications across diverse domains, including medical imaging, scientific research, computer-aided design (CAD), virtual reality (VR), augmented reality (AR), and various industries. This research delves into the integration of a semi-automatic transfer function to enhance 3D visualization, providing users with interactive and intuitive control over rendering settings. The study's emphasis lies in applications such as medical imaging and scientific research, where precise visualization of complex 3D structures is paramount for analysis and understanding.**

***Index Terms*—Real-time volume rendering, 3D visualization, Intuitive controls, Interactive Rendering**

INTRODUCTION

THIS Real-time volume rendering is an innovative technique revolutionizing the visualization of 3D data across various domains. This method swiftly generates and presents compelling three-dimensional images from volumetric datasets, facilitating the rapid exploration of intricate structures. Its widespread applications span medical imaging, scientific research, computer-aided design (CAD), and virtual reality, where it improves disease diagnosis, aids in research understanding, streamlines design processes, and contributes to efficient reservoir management.

Volume rendering techniques intricately hinge on the chosen grid for data representation, displaying remarkable adaptability across structured and unstructured grids. While applicable to diverse grid types, their primary realm of influence lies in uniform grids, especially within the realm of three-dimensional imagery. In the context of uniform grids, voxels assume the pivotal role of the elemental rendering unit, driving the visualization process for intricate three-dimensional datasets. This emphasis on grid type not only underscores the versatility of volume rendering techniques across varied data structures but also highlights a prevailing focus on uniform grids, where voxels emerge as central players in the visualization of complex three-dimensional datasets.

The versatility of real-time rendering extends to entertainment, gaming, and education, making an impact in areas like astronomy, architectural design, flight simulation, and environmental monitoring. As it continues to evolve, real-time volume rendering emerges as a state-of-the-art tool with the potential for significant contributions to the future of 3D data visualization.

The incorporation of color transfer functions in this methodology distinguishes it from grayscale alternatives, providing a more intricate representation of 3D structures. Unlike single-channel intensity mapping, which may struggle to capture the inherent complexity of volumetric data, this approach offers a more effective solution.

For instance, consider a medical imaging scenario where grayscale intensity mapping might inadequately convey the subtleties of different tissue types. In contrast, the usage of color transfer functions allows for a nuanced depiction of diverse anatomical structures. This not only enhances the diagnostic potential but also facilitates a more intuitive and informative exploration of volumetric data. As we embark on the exploration of real-time volume rendering with color transfer functions, our approach unfolds as a transformative paradigm, outperforming alternatives and providing a dynamic, insightful, and efficient solution for the rapid and interactive visualization of volumetric data.

BACKGROUND

In this section, we aim to provide a background overview of some relevant previous works focused on real-time volume rendering. Some works like [1] present a system for integrating transfer function design results from multiple users in volume visualization. [1] also introduces a 2D representation of the transfer function feature space called a transfer function map, constructed using a Multi-Dimensional Scaling (MDS) projection of collected transfer function samples. Some insightful points were made in the name of the challenges of transfer function design and the benefits of the proposed transfer function map in [1]. A two-level interactive interface for designing transfer functions (TFs) for volume visualization was under proposition in [2]. A combination of semi-automatic TF generation methods, such as boundary emphasis, stochastic evolutive search, and manual TF specification aided by dual domain interaction was demonstrated in this work. Thus, [2] aims to provide users with a flexible and efficient tool for TF design and data exploration in volume visualization. The use of multi-dimensional transfer functions in volume rendering to extract specific material boundaries and convey subtle surface properties was made relevant by [3]. This particular work introduced direct manipulation of widgets to make specifying such transfer functions intuitive and convenient for the users. It also described how modern graphics hardware enabled the ability to interactively render with multi-dimensional transfer functions. Some other works such as [4] provide a comprehensive overview of the state of the art in transfer functions for direct volume rendering. The fundamentals of transfer

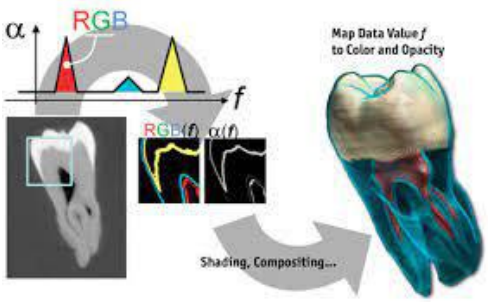

Fig. 1. Illustration over process of Volume Rendering

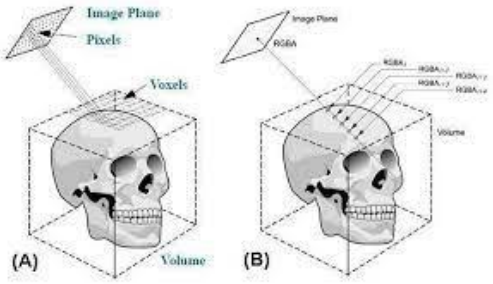

Fig. 2. Demonstration of ray-casting algorithm for volumetric visualization a) Significance of voxels b)RGBA accumulation

functions, their role in volume rendering, and their dual role as a material classifier and carrier of optical properties were also discussed in [4] and some other similar works. There were a few works discussing Semi-automatic transfer functions. [5] in particular discusses the Semi-automatic generation of transfer functions for direct volume rendering and also puts emphasis on visualizing material boundaries in volume data. An introduction to a method for generating transfer functions based on the relationship between data values and their first and second-directional derivatives along the gradient direction was also discussed in detail in [5].

Methodology and Implementation

1. Local setup

To go through with this study, we made a visual studio code setup with GPU support for smoother rendering and visualizations. We have used PyQt5 and vtk libraries with python to implement these visualizations.

PyQt serves as a bridge between Python and Qt, a collection of C++ libraries and development tools offering a versatile foundation for creating graphical user interfaces (GUIs). Qt goes beyond GUIs, providing tools for networking, threads, regular expressions, SQL databases, SVG, OpenGL, XML, and various other robust features, making it a comprehensive toolkit for diverse programming needs. VTK, or Visualization Toolkit, is an open-source software system that provides a powerful set of tools for 3D computer graphics, image processing, and visualization. Widely utilized in scientific data visualization since its establishment in 1993, VTK is actively developed and maintained by Kitware Inc. Written in C++, it offers flexibility with wrappers for Python, Java, and Tcl, catering to various programming requirements.

2. Dataset

For our visualization task, we used a fullhead dataset consisting of raw image data(fullhead.raw) and an accompanying metaImage file(fullhead.mhd). The metaImage file helps in interpreting the raw image file for our purposes. We utilise composite raycasting, which involves the integration of color and opacity information along each ray to generate the final pixel color, resulting in a visually coherent and meaningful representation of the 3D volume. This technique is commonly used in fields such as medical imaging, scientific visualization, and computer graphics. The composite step involves combining these sampled properties using a blending algorithm, often referred to as alpha compositing. α-compositing takes into account both the color and opacity (alpha) values of the sampled points. The algorithm blends colors based on the opacity, allowing for the transparent combination of different colors along the ray. This blending process is crucial for producing realistic and visually informative representations of complex 3D structures within volumetric data.

3. Implementation

We now use Python and vtk to visualize our fullhead dataset. We use PyQt5 to build a render window as shown in the figure below.

3.1 Application Structure: The application is structured using the PyQt5 framework, featuring a primary window that integrates a VTK rendering widget and a right panel housing user controls. The graphical user interface (GUI) is meticulously organized to enhance user interaction with visualization components. QFrame() establishes the frame, QHBoxLayout() configures the main layout, and QWidget() facilitates the addition of widgets to the main window. QVTKRenderWindowInteractor() is employed to enable users to interact with the rendered results.

3.2 VTK Initialization: The initialization of the VTK rendering environment takes place in the init_vtk_widget method. This encompasses the setup of crucial components such as the VTK renderer, rendering window, and interactor. Additionally, a black outline is incorporated to visually outline the boundaries of the loaded volume, providing a foundation for subsequent visualization. QFrame() establishes the frame, QHBoxLayout() defines the main layout, and QWidget() is utilized for adding widgets to the main window. QVTKRenderWindowInteractor() facilitates user interaction with the rendered outcomes.

3.3 File Loading: The application empowers users to load 3D data files in diverse formats, including MHD, VTK, or RAW, through a file browser dialog. The vtk.vtkMetaImageReader() is employed for reading MHD file formats within the project. Following loading, fundamental data information, such as the scalar range, is displayed,

offering users essential context about the dataset.

3.4 Transfer Function Controls: Comprehensive control over transfer functions is provided, enabling users to interactively adjust color, opacity, and gradient settings. The color transfer function is configured using RGB control points, offering a visual representation of data characteristics. Opacity and gradient transfer functions can be finely tuned using sliders through the functions change_opacity() and change_gradient(). The update_transfer_function() method assimilates the updated color, opacity, and gradient values, subsequently re-rendering the window to showcase the modified results.

3.5 Volume Rendering: An integral feature of the application is volume rendering, implemented through the comp_raycasting() method. This method establishes the necessary components for volume rendering, incorporating a GPU-based ray-casting mapper named vtk.vtkGPUVolumeRayCastMapper() and transfer functions. Users can conveniently toggle volume rendering on or off using a checkbox, providing flexibility in visual exploration.

3.6 Scalar Value Interaction: The application facilitates dynamic updates to the scalar value, allowing users to explore the dataset at different scalar levels. The update_scalar_value method prompts users for a new scalar value using the Qt.QInputDialog.getInt() method, and the visualization is subsequently updated accordingly. This feature enhances user engagement and fosters a more detailed exploration of the dataset.

Discussion on the Results

We have integrated informative visual representations depicting the outcomes of our code implementation, which applies a semi-automatic transfer function to visualize the fullhead dataset. Fig.3 provides an illustration of volume rendering with default configurations, offering an initial insight into the dataset. In Fig.4, we introduce a custom-programmed input popup box tailored to solicit user input for scalar values, enhancing user interaction and data exploration. Figures 5, 6, and 7 exemplify the implementation of color sliders, allowing users to input RGB values for the color transfer function. When employed in conjunction with scalar value input, these sliders facilitate the precise configuration of control points for a nuanced color transfer function.

Moreover, Fig.8 delineates the functionality of an opacity input slider, providing users with the capability to define control points for the opacity function in tandem with scalar values. This feature empowers users to fine-tune opacity settings, contributing to a more refined and expressive visualization. Lastly, Fig.9 demonstrates the application of a gradient input slider, affording users the ability to set gradient opacity values. The integration of these interactive elements enhances the adaptability of the visualization process.

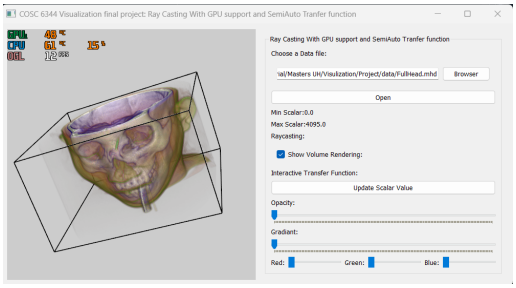

Fig. 3. Volume rendering of fullhead dataset

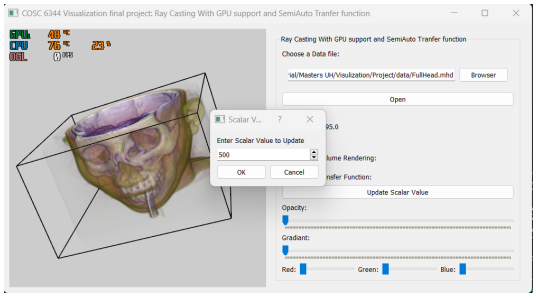

Fig. 4. Input popup box for scalar value input

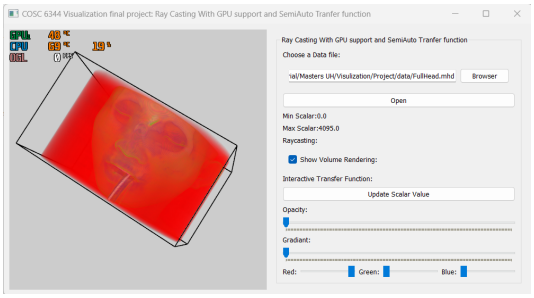

Fig. 5. Testing color slider Red

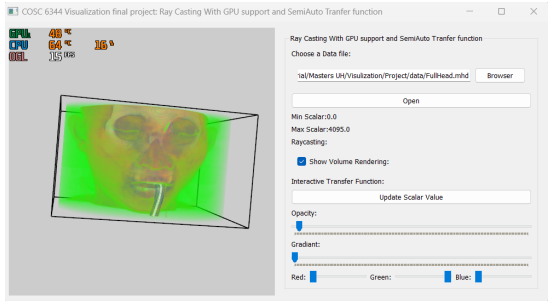

Fig. 6. Testing color slider Green

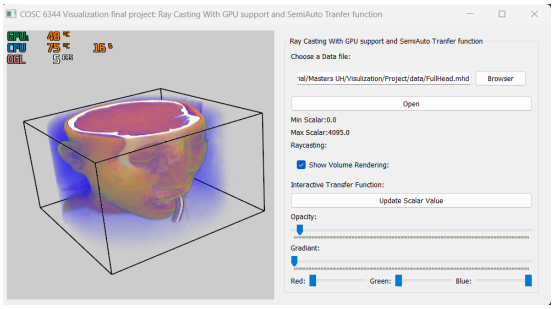

Fig. 7. Testing color slider Blue

Conclusion and Future Work

This study has successfully demonstrated the feasibility and utility of volume rendering through a semi-automatic transfer

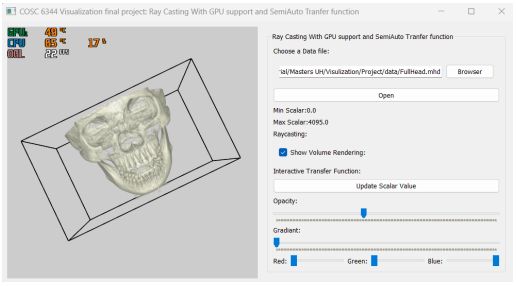

Fig. 8. Testing opacity slider

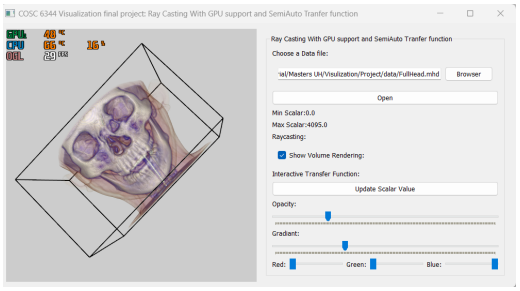

Fig. 9. Testing Gradient slider

function. The implementation involves user input for scalar values and corresponding RGB color information, enabling the creation of a customized semi-automatic transfer function. The same user-driven approach applies to opacity transfer functions, where interactive input is gathered through the rendering window, allowing users to specify scalar values and corresponding opacity levels.

The application of these techniques has been showcased on a 3D dataset representing full head data, illustrating the efficacy of volume rendering using semi-automatic color and opacity transfer functions. Notably, user input empowers control over the opacity of specific boundaries, with users determining which portions are to be rendered opaque by providing scalar data values corresponding to the boundary points.

This interactive and user-centric methodology holds promise for various applications, particularly in fields where precise and adaptable visualization of complex 3D structures is paramount. By enabling users to influence rendering parameters in real-time, our approach contributes to a more intuitive and tailored exploration of volumetric data. As a result, this research not only underscores the practicality of semi-automatic transfer functions in volume rendering but also points towards a user-centric paradigm that could redefine the landscape of 3D data visualization.

Future research in real-time volume rendering encompasses some key areas such as using automatic transfer function optimization using machine learning. This manner of optimization using machine learning to automate and refine transfer function design results in better adaptability to diverse datasets and reduces manual tuning efforts. Second, Dynamic Transfer Functions for Adaptive Rendering are being studied for real-time adjustments based on volume data or viewing changes, enhancing usability. Lastly, research focuses on developing Efficient GPU Algorithms for Large-Scale Data, addressing challenges posed by complex volumetric datasets in fields like medical imaging for optimal real-time rendering on modern GPUs.

Contributions

Shashank Chandgude: Conducted research on the semi-automatic transfer function, exploring its implementation in VTK and PyQT5. Implemented the setup and modification of UI elements, including the addition of color, opacity, and gradient sliders. Integrated a scalar value pop-up message window to prompt user input. Also implemented the update transfer function method, ensuring real-time rendering updates as values were modified. Developed opacity and gradient change functions and incorporated a method for obtaining scalar values from users. In the final stages, played a crucial role in documenting the project and contributed significantly to the report and presentation.

Hemanth Nagapudi: Delved into research on GPU-based volume rendering and ray casting. Skillfully modified the base code for ray casting, transitioning it to utilize a GPU-supported ray-casting mapper. Implemented color change functionality, allowing colors to adjust dynamically with slider value updates. Contributed in documentation adding the in-depth background explanation of Volume rendering and ray casting. Also, prepared slides for the presentation, ensuring a comprehensive and well-communicated project overview.

References

[1] H. Guo, W. Li, and X. Yuan, "Transfer Function Map," 2014 IEEE Pacific Visualization Symposium, Yokohama, Japan, 2014, pp. 262-266, doi: 10.1109/PacificVis.2014.24.

[2] J. Kniss, G. Kindlmann and C. Hansen, "Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets," Proceedings Visualization, 2001. VIS '01., San Diego, CA, USA, 2001, pp. 255-562, doi: 10.1109/VISUAL.2001.964519.

[3] G. Kindlmann and J. W. Durkin, "Semi-automatic generation of transfer functions for direct volume rendering," IEEE Symposium on Volume Visualization (Cat. No.989EX300), Research Triangle Park, NC, USA, 1998, pp. 79-86, doi: 10.1109/SVV.1998.729588.

[4] Pinto, Francisco I& Freitas, Carla (2008) "Volume visualization and exploration through flexible transfer function design." Computers I& Graphics. 32.420-429.10.1016/j.cag.2008.04.004.

[5] Ljung, Patric, Jens H. Krüger, Eduard Gröller, Markus Hadwiger, Charles D. Hansen and Anders Ynnerman. "State of the Art in Transfer Functions for Direct Volume Rendering." Computer Graphics Forum 35 (2016): n. pag.

[6] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. 1988. Volume rendering. In Proceedings of the 15th annual conference on Computer Graphics and Interactive Techniques (SIGGRAPH '88). Association for Computing Machinery, New York, NY, USA, 65–74. https://doi.org/10.1145/54852.378484

[7] Pfister, Hanspeter, Arie Kaufman, and Frank Wessels. 1995. Towards a scalable architecture for real-time volume rendering. In Proceeding of the 10th Eurographics Workshop on Graphics Hardware: August 28-29, 1995, MECC, Maastricht, ed. W. Straßer, 123-130. Aire-La-Ville, Switzerland: Eurographics Association.

[8] S. Weiss and R. Westermann, "Differentiable Direct Volume Rendering," in IEEE Transactions on Visualization and Computer Graphics, vol. 28, no. 1, pp. 562-572, Jan. 2022, doi: 10.1109/TVCG.2021.3114769.

[9] J. Kruger and R. Westermann, "Acceleration techniques for GPU-based volume rendering," IEEE Visualization, 2003. VIS 2003., Seattle, WA, USA, 2003, pp. 287-292, doi: 10.1109/VISUAL.2003.1250384.

[10] S. Bista, J. Zhuo, R. P. Gullapalli and A. Varshney, "Visualization of Brain Microstructure Through Spherical Harmonics Illumination of High Fidelity Spatio-Angular Fields," in IEEE Transactions on Visualization and Computer Graphics, vol. 20, no. 12, pp. 2516-2525, 31 Dec. 2014, doi: 10.1109/TVCG.2014.2346411.

[11] U. D. Bordoloi and H. . -W. Shen, "View selection for volume rendering," VIS 05. IEEE Visualization, 2005., Minneapolis, MN, USA, 2005, pp. 487-494, doi: 10.1109/VISUAL.2005.1532833.