Interactive, in-browser cinematic volume rendering of medical images

Jiayi Xu[a], Gaspard Thevenon[a], Timothee Chabat[a], Matthew McCormick [a], Forrest Li[a],Tom Birdsong[a],Ken Martin[a], Yueh Lee[b], and Stephen Aylward[a]

[a]Kitware, Inc.; [b]The University of North Carolina at Chapel Hill

ARTICLE HISTORY
Compiled September 6, 2022

ABSTRACT
The diversity and utility of cinematic volume rendering (CVR) for medical image visualization have grown rapidly in recent years. At the same time, volume rendering on augmented and virtual reality systems is attracting greater interest with the advance of the WebXR standard. This paper introduces CVR extensions to the open-source visualization toolkit (vtk.js) that supports WebXR. This paper also summarizes two studies that were conducted to evaluate the speed and quality of various CVR techniques on a variety of medical data. This work is intended to provide the first open-source solution for CVR that can be used for in-browser rendering as well as for WebXR research and applications. This paper aims to help medical imaging researchers and developers make more informed decision when selecting CVR algorithms for their applications. Our software and this paper also provide a foundation for new research and product development at the intersection of medical imaging, web visualization, XR, and CVR.

KEYWORDS
Scientific visualization; Cinematic rendering; Visualization toolkit; open source

1. Introduction

Visualization is the foundation of human-computer interaction. This has led to consumer XR (augmented reality and virtual reality) technologies advancing at an amazing rate in recent years. For example, the Oculus Quest 2 (Meta, Inc.) sold over 14.8 million units between October 2020 and June 2022, outpacing total sales for the XBox Series X|S. In general, XR sales grew by 97% in 2021 and are projected to grow by over 240% in 2022.

Just as consumer (gaming) adoption of GPUs in the 1990s was the catalyst for 3D medical volume rendering becoming clinically routine, we anticipate that the consumer adoption of XR systems will spur their clinical use over the next few years. Therefore, we have been advancing the visualization algorithms necessary for medical XR applications in the open-source visualization libraries: VTK (the visualization toolkit, www.vtk.org) and vtk.js (VTK rewritten in JavaScript for in-browser applications, https://github.com/kitware/vtk-js). We anticipate that XR systems will increasingly build upon web technologies, such as vtk.js, as exhibited by the consortium of businesses rallying behind the WebXR standard (Kokelj et al. 2018). We envision a future

CONTACT Jiayi Xu Email: jiayixux@gmail.com

with WebXR dominating the medical XR domain, including, for example, for digital twin and surgical simulation applications (Noguera and Jiménez 2016).

This paper introduces our research to develop open-source, cinematic volume rendering (CVR) algorithms that can run in web browsers or within the WebXR framework, using commonly available GPUs. The arrival of increasingly powerful GPUs set forth the development of photo-realistic rendering algorithms (Fellner 2016). However, many CVR systems are closed-source, and they either require high-end hardware or deliver rendering at non-interactive frame rates. In this paper, we explore various combinations of CVR techniques for visualizing CT and ultrasound volumes using readily available GPUs. Timing results and a two-alternative force choice experiment were conducted to compare rendering methods. While preliminary and focusing on preference rather than performance, these studies provide clear pathways for future CVR research and application development.

2. Background

2.1. In-browser cinematic volume rendering

CVR is a physically-based rendering technique. It integrates light contribution from many directions to simulate multiple scatters. Compared to conventional volume rendering (Lichtenbelt et al. 1998), CVR images have higher density, more natural illumination, and therefore are often judged to be of "better quality" and "more effective" for XR applications. In certain clinical settings, CVR has been shown to reveal pathologies that are otherwise hidden in traditional volume rendering (Eid et al. 2017). Studies have also shown that CVR helps clinicians to comprehend anatomy faster and more accurately (Elshafei et al. 2019).

One major challenge of CVR is that it is computationally expensive. Since there is no upper limit on the number of rays to trace, rendering one image can take several minutes to several hours, and the process restarts after every user interaction. The computational time of CVR must be strictly managed for it to be used clinically, where interactivity is essential.

In-browser CVR has additional memory limitations compared to desktop applications. Therefore, we did not include any rendering techniques that would surpass those memory limits. Typically, in-browser JavaScript applications should be limited to 2 gigabytes of memory, albeit this limit varies by platform and browser.

While CVR can enhance rendering quality, that still depends heavily on the quality of the data itself. As a result, in contrast to CT and MRI, ultrasound CVR presents greater challenges due to its low resolution, intensity inhomogeneities, structured noise, and other imaging artifacts. We will compare and discuss how CVR may improve ultrasound volume rendering in the user preference study presented in this paper.

2.2. Premises

This paper focuses on ray-casting based rendering because it is a mature foundation for a diverse set of algorithms and it is an ongoing basis of research (Jönsson et al. 2013). We used a simplified version of the volume rendering equation by omitting emission and background radiance terms and using discrete sampling to estimate the

integral:

$$L(x, \vec{\omega}_o) = \int_{x_0}^{x} \sigma_s(x') L_{in}(x', \vec{\omega}_o) T(x', x) dx' \tag{1}$$

In the equation, $\sigma_s(x')$ denotes the scattering coefficient, $L_{in}(x', \vec{\omega}_o)$ is the incoming light contribution from direction $\vec{\omega}_o$, $T(x', x)$ represents transmittance, and we integrate along the ray between the two bounds x_0 and x defined by volume bounding box.

Literature sometimes draws parallels between CVR and multiple scatter. However, several of the CVR techniques that are discussed in this paper are less computationally demanding than multiple scatter and yet still bring out details that are not present in conventional volume rendering, such as surface gradient, depth, and occlusion. More specifically, we explore three features that are commonly used in CVR: gradient-based surface model, directional occlusion, and multiple scatter, plus some related techniques that do not fit into these standard categories.

The discussed rendering algorithms are implemented in vtk.js. Vtk.js is a web-based scientific visualization library written in JavaScript, which mirrors its C++ counterpart, the VTK library (Schroeder et al. 2006). Medical data is formatted as topological and geometrical regular array of voxels, each with an assigned color and opacity. All rendering in this paper is driven by WebGL 2.0.

3. Volumetric illumination techniques

3.1. Gradient-based shading

Gradient-based shading is one of the most widely used method to capture surface reflection. It is based on the normalized surface gradient, and in terms of medical data, the gradient of scalar values. Surface lighting has many advantages: relatively low time complexity compared to scattering, realistic surface lighting effect, and easy adaptation to any lighting scenario. It allows clinicians to quickly capture information such as surface orientation and reflection. However, gradient-based shading only takes adjacent voxels into account, thus is unable to capture the scattering effects of translucent materials or media occlusions. For gradient-based method, we replace the in-scattering term in equation (1) with the Phong Shading model (Phong 1975), formulated as:

$$L_{in} = L_{phong} = k_a i_a + \sum_{m}^{lights} A(k_d i_{m,d}(N \cdot L) + k_s i_{m,s}(R \cdot V)^{\alpha}) \tag{2}$$

where k_a, k_d, k_s denote ambient, diffuse and scatter coefficients; i_a, $i_{m,d}$, $i_{m,s}$ are colors for ambient, diffuse and scatter lights; N, L, R, V represents surface normal, light direction, light reflection, and view direction respectively. α denotes specular power.

We also modified the Phong model: for a point light source, users can define its cone angle, or an attenuation factor so shades deepen at locations farther away from the light source.

3.2. Volumetric shading

Volumetric shading is another relatively efficient technique to capture shadow information, sometimes referred to as shadow ray or secondary scatter. At each ray-cast sample location, a secondary ray branches off towards the light source. Production rendering often uses this method to check whether a point is occluded from the light source. For translucent media, such as medical volume data, shadow ray also carries information about the transmittance along the way and produces a global shadow effect. Perceptual studies have shown that volumetric shadow improves understanding of spatial relations and object size (Wanger et al. 1992).

$$L_{in}(x', \vec{\omega}_o) = L_{vs} = \sum_{m}^{lights} \int_{x_0}^{x} \Phi(x', \vec{\omega}_i, \vec{\omega}_o) L_m T(x_m, x') dx' \tag{3}$$

The transmittance term $T(x_m, x')$ in our shading equation (3) is computed at run time rather than using a shadow map in order to reduce memory occupancy. $\Phi(x', \vec{\omega}_i, \vec{\omega}_o)$ is the Henyey-Greenstein phase function for in-scatter and out-scatter directions. The lighting effect is summed over all light sources and their incident radiance L_m. We integrate the shadow ray from sample position x_0 to end position x. Users can also adjust the maximum proportion of the shadow ray. When the reach of the shadow ray is relatively small, the rendering method is equivalent to local ambient occlusion technique with one sampled direction (Hernell et al. 2010).

3.3. Multiple scatter

We have attempted and tested several multiple scatter algorithms that integrate contributions from tertiary rays and beyond, including the Monte Carlo analytical method with importance sampling (Kajiya and Von Herzen 1984; Robert and Casella 2010) and the Path Integration method (Premoze et al. 2003). In all cases, the in-browser renderings remained interactive only if we apply a conservative number of iterations. However, limiting the number of iteration also constrained the visual improvements we hoped to gain from multiple scattering. At a greater number of iterations, the renderings were no longer interactive, hence we chose not to include multiple scatter in our studies. However, real-time multiple scattering in CVR should be available in vtk.js in the not-too-distant future, as we continue to optimize its implementation.

3.4. Other techniques

During our experiments, the above algorithms demonstrated issues that inspired several adaptations. Two of those adaptations are explained in detail next. Additionally, we implemented numerous other techniques to improve performance and/or rendering quality, such as jitter, dither (Danskina and Hanrahan 1992), and dynamically changing sampling distance.

First, we added support for replacing the scalar gradient in the above methods with a density gradient. A density gradient calculates the vertex gradient based on opacity rather than scalar values. We discovered that scalar gradient may incidentally flip the normal of a surface, and if only one side of the surface has reflection, then the shadow might be cast at the wrong location. An example of this error is shown in Figure 1. The green dot represents the position of the light source, which is in-between the camera

and the volume. Because the scalar-gradient normal direction is flipped, the front of the cavity is dim (Figure 1 center column). This error is corrected by using a density-gradient normal (Figure 1 right column). It is also easier to normalize density-gradient magnitude because opacity has a fixed range, whereas the scalar value ranges will vary.

We also found that using gradient opacity makes it easier to define visualization parameters that generate realistic-looking medical images. One significant challenge for realistic rendering is finding the suitable color and opacity transfer functions. This search usually requires a lengthy trial and error process. Gradient opacity disables the rendering of voxels when their gradient magnitude is below a threshold, and therefore edges and surfaces are accentuated without having to precisely manipulate the opacity transfer function. As shown in Figure 1, images on the top row, which use manually tweaked transfer functions, are similar to those at the bottom, which have uniform transfer functions. Note that enabling both gradient opacity and density gradient will lead to three times more gradient look up. If gradients are computed at run time, then there is a linear increase in computational time.

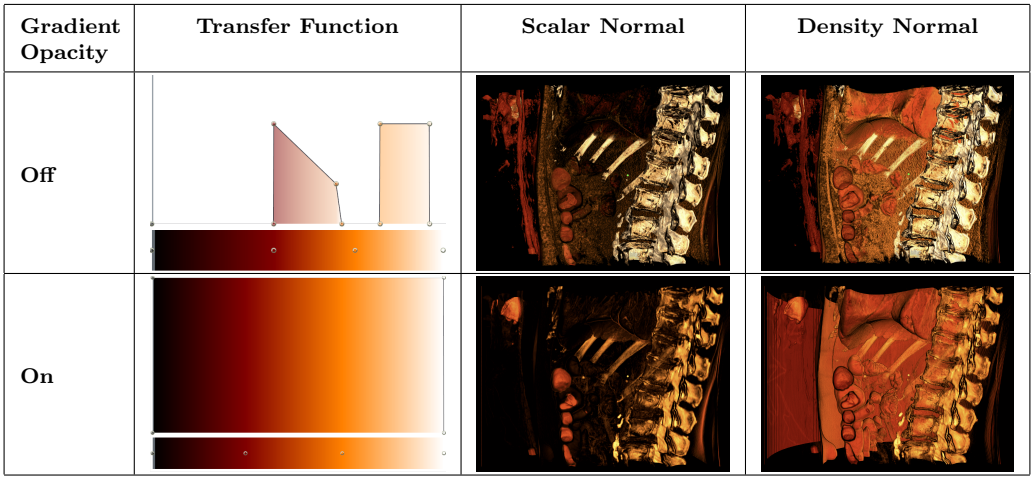


Figure 1. Demonstration of gradient opacity functionality and two computational methods for vertex gradient. The center column computes gradient based on scalar value and the right column uses density (opacity). The bottom uses gradient opacity with uniform opacity function while the top row use custom opacity function. Data is from The Cancer Imaging Archive (2021) published by Clark et al. (2013).

Second, rather than separating gradient-based and volumetric shading, we theorized that the hybrid of these two methods would allow us to adjust a visualization to balance an image's local and global features and noise. Gradient-based shadow reflects local details while volumetric shadow creates global ambient occlusion. This method has been implemented by Kroes et al. (2012), though we adopted the original formula to increase speed and reduce speckle. Our blending coefficient uses interpolation rather than stochasticity, as written in equation (4b), to reduce artifacts introduced by random sampling.

$$c_{blend} = v(1 - e^{|g|}\alpha_{x'}) \tag{4a}$$

$$L_{in} = (1 - c_{blend})L_{phong} + c_{blend}L_{vs} \tag{4b}$$

Equation (4a) is the formula for the blending coefficient c_{blend}. The coefficient changes dynamically at each sample location based on the user-defined blending factor

v which ranges between 0 and 1, gradient magnitude g, and opacity value $\alpha_{x'}$.

4. Methods

4.1. Quantitative speed evaluation

Balancing quality and speed is essential for creating a smooth experience with in-browser interactive rendering, and speed is especially important for most medical visualizations tasks. To quantitatively compare the rendering speed of the algorithms, we computed the first render time and the interactive render time of select techniques on multiple GPUs. First render time is the time span starting from creating the WebGL context to getting the first rendered image. Interactive render time is the average time of rendering one frame when the user rotates or moves the volume. In everyday use, Vtk.js optimizes the interactive frame rate by dynamically adjusting sampling distance to achieve a desired run-time frame rate; however, that option was disabled in all benchmark tests to report accurate render times.

For these quantitative evaluations, as well as for the preference study presented in the next subsection, we used three datasets: medium size chest CT scan (214MB), small size fetus 3D ultrasound scan (161MB) from TomoVision (2022), and an even smaller size head CT scan (15.6MB). Five different sets of parameters were applied to each data representing five rendering techniques: (1) direct volume rendering with ray casting, (2) gradient based shading, (3) volumetric shading with full shadow ray, (4) volumetric shading with partial shadow ray, and (5) a mixture of gradient based shading with volumetric shading. An overview of rendering results is included in Figure 2.

The parameters specific to each rendering method were optimized for each dataset, but all lighting related parameters were held consistent to ensure a fair comparison. The consistent lighting conditions, however, did result in images based on volumetric methods appearing darker, since they include global occlusion. Furthermore, to accommodate less powerful GPUs in the speed test, we slightly reduced the rendering quality compared to those shown in Figure 2 by reducing the ray-cast sampling distance. For all experiments, the canvas size is fixed to 500x500 so a consistent number of fragments is processed by the GPUs.

4.2. Perceived quality evaluation

To evaluate the perceptual quality of the rendering methods, we conducted an anonymous two-alternative-forced-choice study. We arranged the five renderings of each dataset into random pairs of two and presented those pairs to 13 participants. Participants were asked to select the rendering that they judged to have "higher quality" in each pair. We did not design a performance study to evaluate each visualization method's utility for a specific medical task. A performance study would require many more experimental conditions and specialized participants. Instead, our preference study was meant to reveal broad speed-vs-quality insights.

5. Results

Results for the quantitative evaluations are summarized in Figure 3 and Figure 4. Figure 3(a) plots the interactive time from three brands of GPU that are commonly found

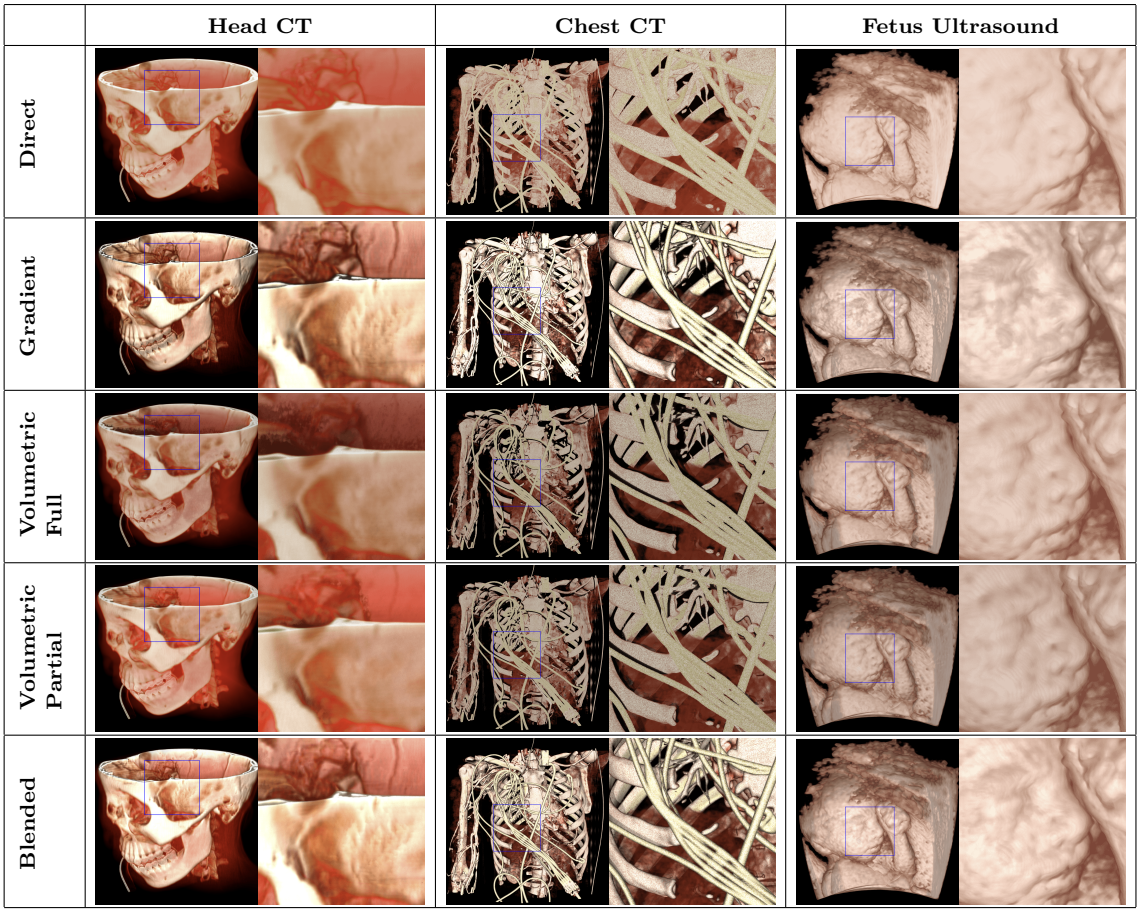


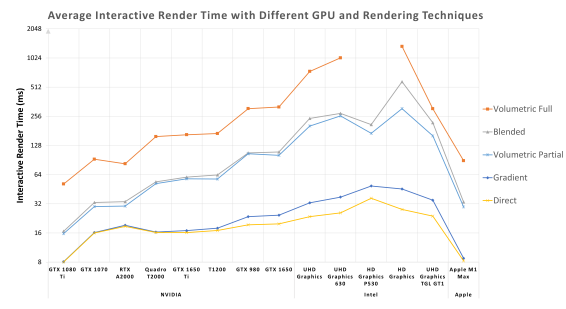
Figure 2. Rendering overview for all datasets and rendering methods included in the studies.

in laptops and PCs: NVIDIA, Intel, and Apple. It is worth noting that computationally demanding rendering, such as volumetric shadow with full shadow ray, did lead to test failures when the GPU was not powerful enough, as indicated by the missing entry in Figure 3(a). However, we experienced test failures only with Intel GPUs, and none with NVIDIA GPUs. Figure 3(b) shows the interactive time for 3 datasets that were averaged across all GPUs. Figure 4(a) summarizes the first render time for different GPUs, which exhibit a trend that is largely different from the interactive render time.

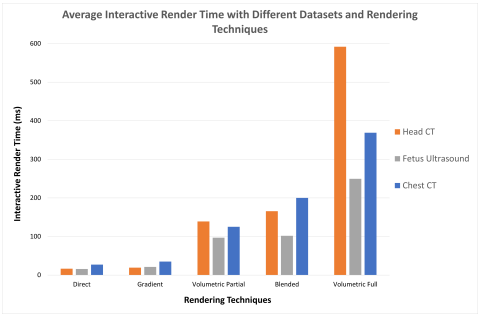
Results for the qualitative preference study are outlined in Figure 5. The figures show the percentage of vote each rendering technique received in the two-alternative-forced-choice study in which each method was compared to every other.

6. Discussion

This work strongly indicates that real-time, in-browser and WebXR rendering with CVR is possible with the open-source vtk.js toolkit. Previously, real-time CVR was limited to closed-source commercial solutions and expensive hardware. Now, researchers are able to explore a variety of CVR techniques and compare their merits. The GPU that delivered the best performance in this study, the NVidia GTX 1080 Ti, was released in 2017. There are now an abundance of higher performing GPUs available on the market that will drive even faster renderings.

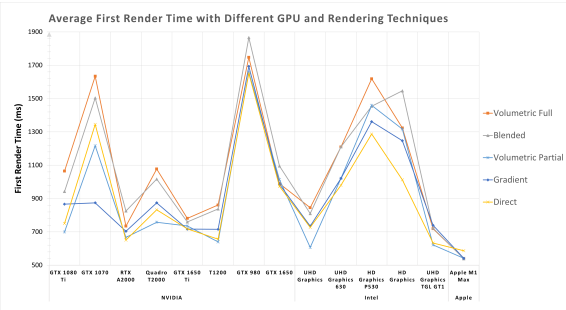


(a) Interactive render time for various GPUs and rendering techniques. Each data point reports the time average over 3 datasets. The missing entry for *HD Graphics P530 - Volumetric Full* represents a test failure due to computational requirements exceeding the GPU's capabilities.

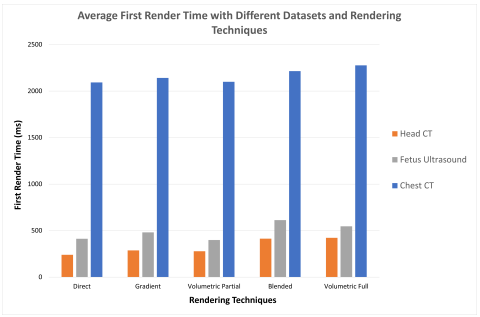


(b) Interactive render time for 3 datasets and rendering techniques. Each data entry reports the time average over all GPUs.

Figure 3. Integrative render time summary



(a) First render time for various GPUs and rendering techniques. Each data point reports the time average over 3 datasets.



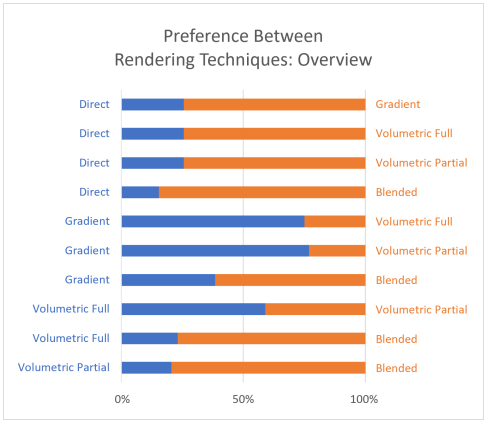
(b) First render time for 3 datasets and rendering techniques. Each data entry reports the time average over all GPUs.

Figure 4. First render time summary

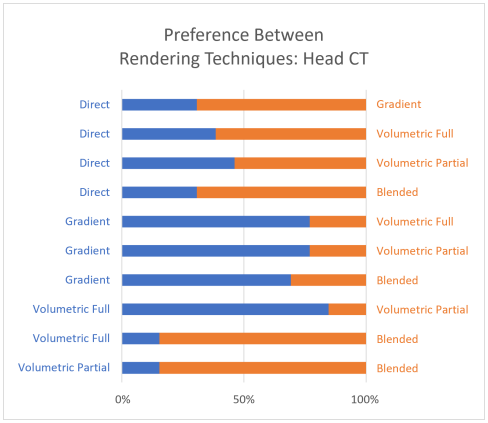
It is initially a surprise that the smallest data, head CT, was roughly 1.5 times slower to render compared to our largest data, chest CT (Figure 3(a). The difference can be explained by their opacity transfer functions. The chest CT data consists mostly of bones and only a small portion of soft tissue, making it more surfacic and opaque. However, the head CT used a transfer function that produced a thick layer of translucent tissues surrounding the skull. As a result, a much greater number of points were sampled when a ray is traced across the head CT, while the same process for chest CT could terminate much earlier, as soon as the accumulated opacity reaches 1.0. This reveals that the chosen opacity transfer function has a high impact not only on visualization quality but also on rendering speed.

Besides GPU power, more factors are at play in determining the first render time, as it involves buffer creation in CPU, as well as streaming the data to the GPU before they are processed. To predict the pre-processing time, we need to further decompose and trace the entire rendering pipeline. Volume data pre-processing speed is mostly unrelated to the rendering method, since the data points for each GPU are well clustered together. Nonetheless, we can still see a clear pattern in Figure 4(b), which aligns with our expectation that first render time increases linearly with data size.

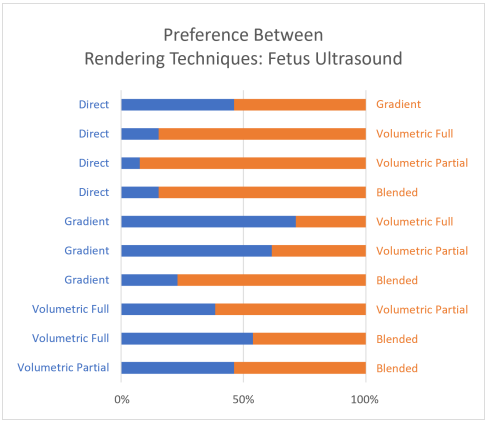
The tendencies in the preference study are similar across types of volumes, but users are less decided on which rendering brings out the "highest quality" rendering of



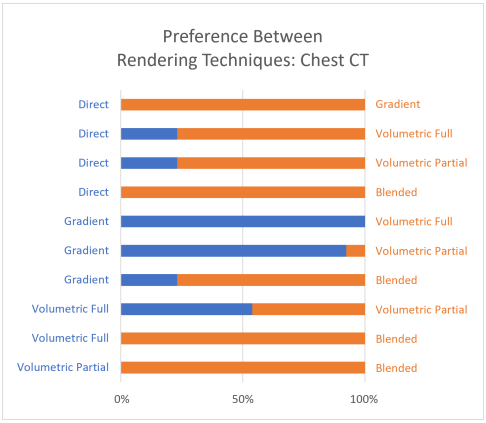
(a) Preference report summed over all datasets.



(b) Preference report for head CT volume data.



(c) Preference report for fetus Ultrasound data.



(d) Preference report for chest CT data.

Figure 5. Two-alternative-first-choic study: users were asked to choose 1 out of the 2 presented images for 10 combinations of rendering methods.

ultrasound data, as shown in Figure 5(c). When it comes to chest CT in Figure 5(d), the bias for specific rendering techniques becomes much stronger. For example, since gradient shading is able to bring out sharp details better than the other techniques, it makes the biggest difference on surface-like, opaque structures as shown in the chest CT data. This advantage is less noticeable for ultrasound volume data, for which there is almost equal preference for the volumetric and the hybrid method.

In general, the preference study indicates that there is not a single magical formula for creating the best CVR; however, Figure 5(a) reveals that users prefer every CVR technique over direct volume rendering. For CT scans, gradient-related techniques were more often preferred than global shading alone. For ultrasound data, choosing suitable color and opacity transfer functions is vital, before adding any CVR features. Often, masking is also necessary to block out redundant volume or noise in ultrasound data. We also noticed that gradient-based methods often imposed an overwhelmingly metallic feeling and made the ultrasound data look unrealistic. Additionally, it is also harder to grasp the sense of the depth or orientation of ultrasound data. We hypothesize that a multiple scatter method could drastically improve ultrasound rendering due to its ability to bring out the density and sub-surface properties of the volume, but as previously noted, we are in the process of optimizing vtk.js' multiple scatter method to achieve reasonable rendering rates. For both CT and ultrasound data, ambient lighting and volumetric shading offered benefits. Surprisingly, the more computationally and

memory demanding volumetric shadow rendering was generally considered to have lesser positive effect compared to gradient shadow. Unsurprisingly, the hybrid method was favored over individual gradient or volumetric shading.

7. Future Work

At the time of writing this paper, we have added a preliminary version of local ambient occlusion (LAO) technique for volume rendering (Hernell et al. 2010), which was not included in the two studies presented in this paper. LAO samples an arbitrary number of neighboring voxels in an arbitrary number of directions. The number of pixels to consider is defined as kernel radius, and the number of directions is defined as kernel size. Unlike volumetric methods, LAO creates softer shadows, and it stands mid-way between local and global shading. Additionally, volumetric shading might occlude parts of a volume completely, and LAO avoids such undesirable effect. Figure 6 includes images taken using a hybrid of gradient and LAO methods with increasing kernel radius.

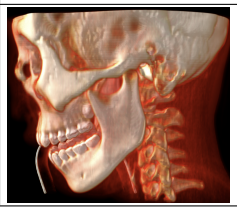
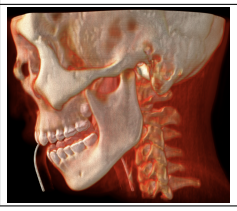
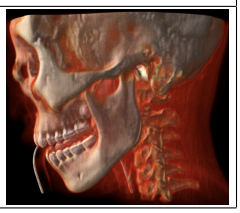
Kernel Radius	**1**	**3**	**5**	**7**
Image				

Figure 6. Images taken from a combination of gradient-based shading and LAO. Kernel radius represents the number of voxels traversed to calculate LAO.

As with LAO, many other CVR algorithms are not fully covered by this paper. Although there is abundant literature on CVR for production rendering, more extensive research needs to be done to determine how well they interact with medical data. The two studies we conducted were not clinically oriented. The expertise of doctors and clinicians must be combined with clearly defined clinical tasks and metrics, as well as state-of-the-art GPUs, to make final conclusions on CVR's usability in clinical settings.

Beyond testing CVR on laptops and PCs, we have also successfully generated interactive CVR renderings on XR devices, including the Looking Glass holographic display (Looking Glass Factory, Inc.), Oculus Quest 2, and mobile AR. To advance CVR rendering for WebXR in medical applications, the driving clinical tasks must be defined. We hope that the open-source software and studies presented in this paper will accelerate the discovery of those driving clinical tasks.

8. Open-Source Software

The CVR methods presented in this paper are freely available as part of the open-source vtk.js library, under the Apache 2.0 license: `https://kitware.github.io/vtk-js/index.html`. The vtk.js library includes several examples of CVR rendering in WebXR, such as `https://kitware.github.io/vtk-js/examples/`

`WebXRHeadGradientCVR.html`. The code and data for the observer and performance studies presented in this paper are also available: `https://github.com/KitwareMedical/vtk.js-render-tests`.

Funding

This work was funded, in part, by the NIH via NIBIB and NIGMS R01EB021396, NIBIB R01EB014955, NCI R01CA220681, and NINDS R42NS086295.

References

Clark K, Vendt B, Smith K, Freymann J, Kirby J, Koppel P, Moore S, Phillips S, Maffitt D, Pringle M, et al. 2013. The Cancer Imaging Archive (TCIA): Maintaining and Operating a Public Information Repository. Journal of Digital Imaging. 26(6):1045–1057. doi:10.1007/s10278-013-9622-7.

Danskin John, Hanrahan Pat. 1992. Fast algorithms for volume ray tracing. In Proceedings of the 1992 workshop on Volume visualization (VVS '92). Association for Computing Machinery, New York, NY, USA, 91–98. https://doi.org/10.1145/147130.147155

Dappa E, Higashigaito K, Fornaro J, Leschka S, Wildermuth S, Alkadhi H. 2016. Cinematic rendering – an alternative to volume rendering for 3D computed tomography imaging. Insights into Imaging. 7(6):849–856. doi:10.1007/s13244-016-0518-1.

Ebert LC, Schweitzer W, Gascho D, Ruder TD, Flach PM, Thali MJ, Ampanozi G. 2017. Forensic 3D Visualization of CT Data Using Cinematic Volume Rendering: A Preliminary Study. American Journal of Roentgenology. 208(2):233–240. doi:10.2214/ajr.16.16499.

Eid M, De Cecco CN, Nance JW, Caruso D, Albrecht MH, Spandorfer AJ, De Santis D, Varga-Szemes A, Schoepf UJ. 2017. Cinematic Rendering in CT: A Novel, Lifelike 3D Visualization Technique. American Journal of Roentgenology. 209(2):370–379. doi:10.2214/ajr.17.17850. [accessed 2019 Mar 18].

Elshafei M, Binder J, Baecker J, Brunner M, Uder M, Weber GF, Grützmann R, Krautz C. 2019. Comparison of Cinematic Rendering and Computed Tomography for Speed and Comprehension of Surgical Anatomy. JAMA Surgery. 154(8):738. doi:10.1001/jamasurg.2019.1168.

Fellner FA. 2016. Introducing Cinematic Rendering: A Novel Technique for Post-Processing Medical Imaging Data. Journal of Biomedical Science and Engineering. 09(03):170–175. doi:10.4236/jbise.2016.93013.

Glemser PA, Engel K, Simons D, Steffens J, Schlemmer H-P, Orakcioglu B. 2018. A New Approach for Photorealistic Visualization of Rendered Computed Tomography Images. World Neurosurgery. 114:e283–e292. doi:10.1016/j.wneu.2018.02.174.

Hernell F, Ljung P, Ynnerman A. 2010. Local Ambient Occlusion in Direct Volume Rendering. IEEE Transactions on Visualization and Computer Graphics. 16(4):548–559. doi:10.1109/tvcg.2009.45.

Jönsson D, Sundén E, Ynnerman A, Ropinski T. 2013. A Survey of Volumetric Illumination Techniques for Interactive Volume Rendering. Computer Graphics Forum. 33(1):27–51. doi:10.1111/cgf.12252.

Kajiya JT, Von Herzen BP. 1984. Ray tracing volume densities. ACM SIGGRAPH Computer Graphics. 18(3):165–174. doi:10.1145/964965.808594.

Kokelj Z, Bohak C, Marolt M. 2018. A web-based virtual reality environment for medical visualization. IEEE: 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).

Kroes T, Post FH, Botha CP. 2012. Exposure Render: An Interactive Photo-

Realistic Volume Rendering Framework. Zuo X-N, editor. PLoS ONE. 7(7):e38586. doi:10.1371/journal.pone.0038586.

Lichtenbelt B, Crane R, Naqvi S. 1998. Introduction to volume rendering. Upper Saddle River, Nj: Prentice Hall.

Noguera JM, Jiménez JR. 2016. Mobile volume rendering: Past, present and future. IEEE transactions on visualization and computer graphics. 22(2):1164–1178. doi:10.1109/TVCG.2015.2430343

Phong BT. 1975. Illumination for computer generated pictures. Communications of the ACM. 18(6):311–317. doi:10.1145/360825.360839.

Premoze S, Ashikhmin M, Shirley P. 2003. Path Integration for Light Transport in Volumes. Eurographics Workshop on Rendering. doi:10.2312/EGWR/EGWR03/052-063.

Robert CP, Casella G. 2010. Monte Carlo statistical methods. New York: Springer.

Schroeder WJ, Martin KM, Lorensen WE. 2006. The visualization toolkit : an object-oriented approach to 3D graphics ; [visualize data in 3D-medical, engineering or scientific ; build your own applications with C++, Tcl, Java ord Python/ Will Schroeder ; Ken Martin ; Bill Lorensen. New York, Ny: Kitware.

The Cancer Imaging Archive. 2021-. Version 1. Applied Proteogenomics OrganizationaL Learning and Outcomes (APOLLO) Research Network. (2021). [updated 2021 Nov. 24; accessed 2022 Mar. 22]. Data from the Applied Proteogenomics OrganizationaL Learning and Outcomes Lung Adenocarcinoma cohort [APOLLO-5-LUAD] Collection [Data set]. https://doi.org/10.7937/BDM9-4623

HOME - TomoVision. tomovisioncom. [accessed 2022 Jul 14]. https://tomovision.com/index.html.

Usher W, Pascucci P. 2020. Interactive Visualization of Terascale Data in the Browser: Fact or Fiction? IEEE 10th Symposium on Large Data Analysis and Visualization (LDAV). p. 27–36

Wanger LR, Ferwerda JA, Greenberg DP. 1992. Perceiving spatial relationships in computer-generated images. IEEE Computer Graphics and Applications. 12(3):44–58. doi:10.1109/38.135913.