Immersive 3D Medical Visualization in Virtual Reality using Stereoscopic Volumetric Path Tracing

Javier Taibo* Jose A. Iglesias-Guitian†

Universidade da Coruña
CITIC - Centre for ICT Research

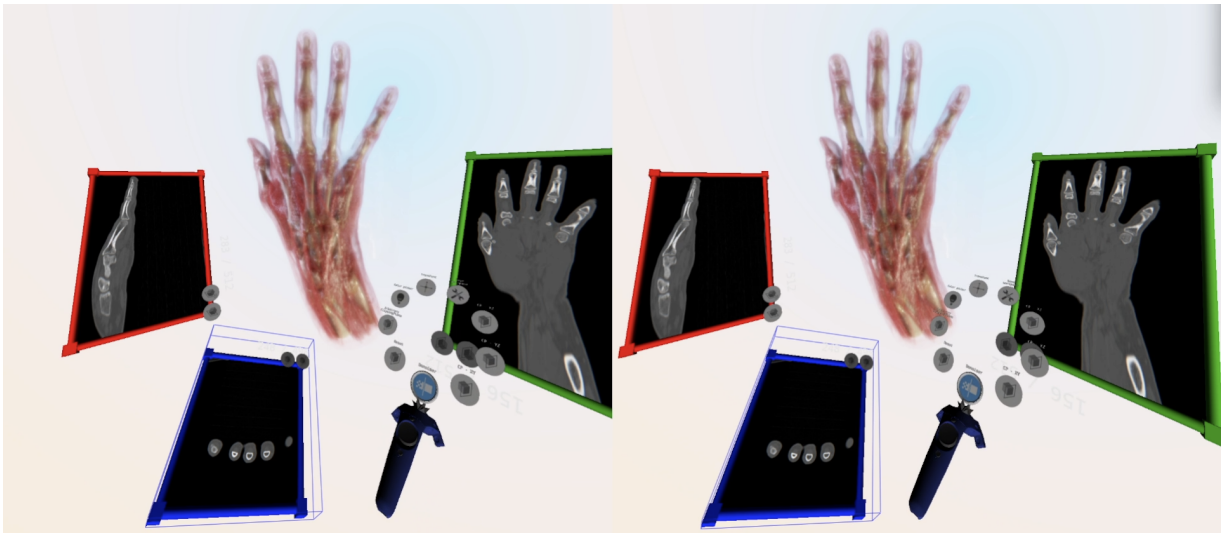

Figure 1: Stereoscopic interactive session rendering a photorealistic 3D reconstruction of the HAND CT scan.

ABSTRACT

Scientific visualizations using physically-based lighting models play a crucial role in enhancing both image quality and realism. In the domain of medical visualization, this trend has gained significant traction under the term cinematic rendering (CR). It enables the creation of 3D photorealistic reconstructions from medical data, offering great potential for aiding healthcare professionals in the analysis and study of volumetric datasets. However, the adoption of such advanced rendering for immersive virtual reality (VR) faces two main limitations related to their high computational demands. First, these techniques are frequently used to produce pre-recorded videos and offline content, thereby restricting interactivity to predefined volume appearance and lighting settings. Second, when deployed in head-tracked VR environments they can induce cybersickness symptoms due to the disturbing flicker caused by noisy Monte Carlo renderings. Consequently, the scope for meaningful interactive operations is constrained in this modality, in contrast with the versatile capabilities of classical direct volume rendering (DVR).

In this work, we introduce an immersive 3D medical visualization system capable of producing photorealistic and fully interactive stereoscopic visualizations on head-mounted display (HMD) devices. Our approach extends previous linear regression denoising to enable real-time stereoscopic cinematic rendering within AR/VR settings. We demonstrate the capabilities of the resulting VR system, like its interactive rendering, appearance and transfer function editing.

*e-mail: javier.taibo@udc.es
†e-mail:j.iglesias.guitian@udc.es

Index Terms: Computing methodologies—Virtual reality—;——Computing methodologies—Ray tracing—; Human-centered computing—Graphical user interfaces—;—

1 INTRODUCTION

The importance of medical volume visualization for dissecting and comprehending three-dimensional (3D) anatomy is undeniable. Direct volume rendering (DVR) is a computer graphics technique often used in medical imaging to reconstruct and render 3D visualizations directly from cross-sectional images, i.e. CT or MRI scans, without the need for intermediate surface reconstruction or segmentation. Modern DVR has evolved into advanced ray-casting implementations that harness the computational power of parallel graphics processing units (GPUs) to expedite the visualization process [6]. Such evolution has also led to exciting extensions of DVR into physically-based rendering algorithms [14, 39]. For example, Monte Carlo volumetric path-tracing (VPT) allowed photo-realistic three-dimensional reconstructions from cross-sectional images, i.e. CT or MRI scans [16, 35]. This approach has taken significant traction under the name of cinematic rendering (CR) [26, 29].

The integration of augmented and virtual reality (AR/VR) technology into medical visualization provides additional cues for the identification of critical structures. Tracking capabilities of HMDs can enhance the perception of volumetric visualizations, as users can freely move around the reconstructed volume and examine it from every angle.

The accessibility to head-mounted display (HMD) systems for medical visualization unlocks new opportunities, presenting novel avenues for diagnosis [12], surgery planning [24], and medical education [27]. However, the seamless integration of DVR, CR, and other advanced volume rendering techniques with VR presents its own challenges. One of the principal challenges lies in achieving

the high refresh rates necessary for an optimal VR experience. VR necessitates consistent and rapid update rates to mitigate the emergence of cybersickness symptoms, a discomforting and disorienting phenomenon [17]. These requirements become even more challenging as we consider the integration of physically-based rendering, i.e., CR or VPT, typically used for progressive [16, 21, 33] or offline volume rendering [9]. Progressive rendering approaches render a quick preview during interaction and refine the image once the interaction stops. In VR, however, the interaction never stops, and those techniques are not directly applicable to head-tracked environments since they produce noticeable temporal flicker.

In this work, we propose a novel method to integrate stereoscopic Monte Carlo VPT while achieving the demanding high refresh rates imperative to prevent VR-induced discomfort. Our main objective is to provide users with an experience that is not only comfortable but also innovative and deeply engaging when interacting with volumetric data. Our approach enables users to employ common interactions typically associated with direct volume rendering (DVR) and cinematic rendering (CR) while delivering updates of photorealistic 3D reconstructions rendered in real-time with physically-based Monte Carlo VPT, tailored explicitly for stereoscopic VR.

Our main contributions can be summarized as follows:

- We introduce a new virtual reality rendering framework that adapts previous weighted linear regression denoising to support interactive volumetric path tracing on stereoscopic HMDs. Our approach requires only one sample-per-pixel, and extends previous work with a more precise stereo reprojection leveraging dual motion vectors.

- We showcase a practical virtual reality implementation of a visualization system in the medical domain, demonstrating feasible interactive rendering and appearance manipulation for 3D medical data in real time. Our approach addresses common interactive operations for DVR within an AR/VR setting, achieving unprecedented quality for interactive cinematic rendering within commodity VR settings.

2 Related Work

2.1 Immersive DVR visualizations for virtual reality

Immersive virtual reality (VR) can serve as a fruitful platform for enhancing direct volume rendering (DVR) visualizations. We can find, from large tiled display walls and projection environments using 3D shutter glasses [7], to light-field displays enhancing depth-perception with eye-naked setups [2, 11]. For instance, drawing inspiration from the CAVE [3], Jadhav and Kaufman [12] proposed a dedicated visualization workbench for radiologists combining head tracking and shutter glasses to interact with DVR visualizations.

Stereoscopic volume rendering is often based on volume ray-casting [1] to generate high-quality stereoscopic images. Nowadays, there is a wide variety of implementations aimed at amortizing the costs of DVR for stereoscopic HMD devices [4, 34, 37]. For example, Waschk and Krüger [37] introduced a method to adjust image quality during rendering without suffering a perceptual loss, based on adjusting rendering resolution in the peripheral vision [20]. Unlike cited prior works implementing traditional DVR for VR, our work differs in achieving higher quality for fully interactive cinematic rendering (VPT) on stereoscopic HMD devices and still supporting most common DVR manipulations.

2.2 Denoising global illumination for DVR

The perceptual benefits of applying more advanced illumination algorithms for 3D scientific visualizations are being studied both in computer graphics [18] and in the medical visualization domain [5, 26, 29]. In the beginning, researchers pursued DVR implementations that supported various effects, i.e., dynamic ambient occlusion and color bleeding [13, 28, 39]. This trend also found its counterpart in VR, where e.g., Scholl et al. [31] extended DVR with shadow rays and global illumination. However, since DVR techniques in scientific visualization historically relied on ray marching algorithms, i.e. Rezk-Salama [30], they typically suffer from undesired bias. The extension of unbiased Monte Carlo (MC) path-tracing to volumetric participating media is commonly known as volumetric path tracing (VPT) [25].

Kroes et al. [16] and Liu et al. [19] demonstrated that MC progressive VPT using GPUs could achieve interactive frame rates for DVR unbiased rendering. These progressive rendering solutions can produce quick previews during interaction and refine the image once the interaction stops. However, in VR the interaction never stops, and those techniques are not directly applicable to head-tracked environments as the temporal flicker introduced by MC noise would significantly increase the cybersickness symptoms [37]. Thus, recent work focused on different sampling and filtering techniques to reduce the MC noise. For instance, Radziewsky et al. [35] proposed a progressive estimate for joint importance sampling of visibility information and to suppress the occurrence of fireflies. Adaptive temporal sampling for VPT [21] attacked the problem by adaptive temporal reprojection and dedicating more samples on critical regions. Unfortunately, their suggested sampling budgets exceeded our available resources for current HMD resolutions. Other authors also proposed offline denoising based on deep neural networks [9] and a weighted linear regression scheme for real-time denoising [10], all in the context of medical data visualization to reduce MC noise and temporal flicker.

Given our sampling budget limitations, limited to a maximum of one sample-per-pixel (spp) in average, we decided to build on top of a Kroes et al. [16] progressive VPT framework and extend previous real-time weighted linear regression denoising [10] to work on stereoscopic HMD devices.

In summary, we propose an immersive 3D medical visualization system based on stereoscopic volume rendering on commodity HMD devices. Our approach can render fully interactive photorealistic 3D visualizations of medical data in real-time, using MC-VPT with one sample-per-pixel (spp) and enabling its use for VR applications thanks to a new spatio-temporal denoiser for stereoscopic VPT. We believe our approach can offer a deeply engaging VR experience, mitigating the risk of cybersickness symptoms caused by the temporal flicker induced by MC noise and overcoming the limitations that have previously hindered akin DVR interactivity for cinematic rendering (VPT) approaches.

3 System Overview

In this section, we aim to provide a general description of our immersive 3D visualization system and its associated rendering pipeline, shown in Fig. 2. Our ultimate goal is to create an engaging experience for 3D medical visualization within immersive VR. Consequently, our system shares similar requirements to most VR applications. First, we must render stereoscopic images at high-resolutions for each eye, commonly exceeding 1024×1024 per eye (resulting in combined resolutions of 2K and over). Moreover, our system must operate with high frame rate, ideally 60-90 Hz, to prevent user discomfort.

Additionally, our immersive system needs to be designed to address the following challenging tasks simultaneously: a) generating photorealistic, high-quality 3D stereoscopic visualizations in VR, using computationally intensive volumetric path tracing (VPT); and b) supporting familiar visualization tools utilized for DVR within the new rendering pipeline. These tasks pose a formidable challenge, especially when considering the aforementioned requirements imposed by VR visualization.

With this demanding scenario in mind, the architecture of the system (see Fig. 2) is structured around four main modules: i) the

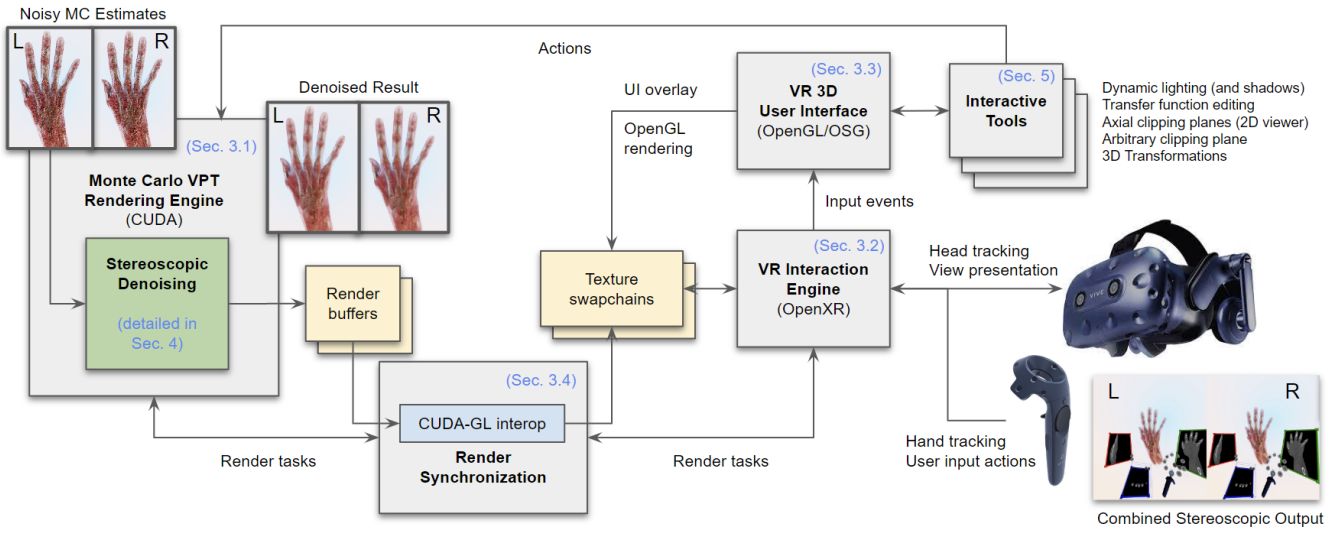

Figure 2: General overview of our immersive 3D stereoscopic rendering system. Our system enables the creation of photorealistic 3D stereoscopic visualizations in VR, while providing users with common visualization tools and widgets for exploring volumetric data, akin to DVR but enhanced with cinematic rendering quality and low-latency response times as required by HMD devices.

VPT rendering engine, ii) the VR interaction engine, iii) the VR 3D user interface, and iv) the rendering synchronization. These components will address the various challenges presented earlier, so we will dedicate the rest of this section to introducing these modules in more detail.

3.1 VPT rendering engine

The Monte Carlo VPT (MC-VPT) rendering engine is at the core of our system, mostly responsible for generating real-time images from volumetric datasets, which are then composed and presented to the HMD user. Physically-based global illumination with VPT offers an elegant and versatile solution, covering a wide variety of lighting effects, including indirect global illumination and shadowing [25]. This meets our specifications for producing high-quality visualizations of 3D medical scans. Our real-time framework is built upon progressive MC-VPT for DVR [16], a common approach that generates quick previews during interaction and refines the image once the interaction stops.

However, VPT previews tend to produce noisy approximations of the volume rendering integral (VRI) due to its stochastic nature, especially given the limited number of rendering samples achieved under real-time constraints. This issue becomes more pronounced when targeting high resolutions, presenting a greater challenge in the context of stereoscopic rendering. In particular, VPT noise degenerates into temporal flicker when a user interacts with the parameters of a DVR scene, e.g., modifying lighting or transfer functions, since VPT regenerates a new image restarting the MC integration process. Within the context of VR, user interaction is continuous, thus making progressive rendering unsuitable for head-tracked environments.

For all these reasons, we deemed it essential to incorporate a VPT denoiser in our system to filter out as much MC noise as possible without compromising image sharpness. Intuitively, there exists a promising opportunity for exploiting spatio-temporal coherence in stereoscopic rendering, thus a specialized denoiser taking advantage of the stereoscopic coherence seems appropriate. Given the stringent requirements for real-time execution, we opted to build upon a linear regression approach given its suitability for interactive and real-time rendering systems [10, 23]. An additional advantage of linear regression approaches is their independence from resource-intensive pre-training or pre-processing steps, making them especially desirable in the context of medical data visualization.

The ultimate goal of our stereoscopic denoising is to address MC variance noise in VPT results. We aim to achieve this by implementing a real-time denoising framework capable of leveraging the spatial and temporal coherence among pixel colors and stereoscopic views, ultimately delivering enhanced numerical and visual outcomes. Since this embraces one of the key contributions of our system we will cover it in further detail in Sec. 4.

3.2 VR interaction engine

This is the central module that orchestrates the VR immersive system. It is responsible for the frame synchronization among the different system components. It implements the main interaction loop and interfaces with the VR hardware. This interaction with the VR hardware is made through the OpenXR standard API, so the system is abstracted from the hardware being used. The results presented in this work have been implemented in a prototype using HTC Vive Pro, but the system has been designed and successfully tested to work on other HMD devices (e.g., Oculus Rift, Oculus Rift S, and HTC Vive XR Elite). The main interaction loop follows this sequence:

- Receive input events from the user and VR hardware.
- Wait for the start of the next frame/iteration and retrieve the presentation time for the upcoming frame.
- Predict hands and head poses for the presentation time of the next frame.
- Dispatch a render task request to the VPT rendering engine through the render synchronization module (see Sec. 3.4).
- Acquire OpenGL textures for displaying images to be composited on the HMD.
- Receive VPT rendering results and transfer them to OpenGL textures using the CUDA-OpenGL interoperability API.
- Initiate the rendering of the user interface through the VR 3D User Interface module (see Sec. 5). This module superimposes the UI elements over the VPT rendering output.

- Transmit the final images, stored in the previously acquired OpenGL textures, to the OpenXR compositor for presentation to the user via the HMD.

3.3 VR user interface

The VR user interface (VR-UI) module is responsible for the rendering and interaction through the tools offered to the user to manipulate the visualization of the volumetric datasets. Given the inherent computational intensity of the VPT rendering engine, we opted not to burden it unnecessarily with interface rendering using the MC path tracer. Instead, we pursued a more efficient approach by adopting a hybrid strategy, harnessing the GPU's highly optimized capabilities for surface raster rendering. As OpenXR needs interfacing with a graphics API to send the images to be composited in the HMD, we already had a rendering context well suited for the task. We selected OpenGL as the API to interface with OpenXR and render the interface as an overlay on the VPT-rendered image.

To manage the contents of the VR-UI in an efficient way, we chose OpenSceneGraph (OSG) as a high-level abstraction layer over OpenGL. All the VR-UI is composed in a scene graph structure, updated with the user inputs received from the VR interaction engine module (mainly head and controllers' poses as well as their associated input events). The rendering engine for VR-UI configures a scene graph with a stereoscopic setup (one camera per eye). The cameras render to FBOs where the OpenGL textures (coming from OpenXR swapchain images) are attached as render targets.

The tools and 3D widgets available in the VR-UI are implemented as pluggable components. The VR-UI module receives user inputs, updates internal scene graph nodes, and sends commands to the VPT rendering engine to alter the volume visualization. A meaningful selection of such tools and 3D widgets is later discussed in Sec. 5.

3.4 Render synchronization

The render synchronization module acts as a crucial intermediary connecting the rendering engine and the interaction engine. It serves as a synchronization point between the threads of both modules, adhering to the timing directives established by the interaction engine. Additionally, this module is responsible for encapsulating CUDA-OpenGL interoperability operations.

This module offers flexibility by supporting different modes of operation. It can work in synchronous mode, offering the advantage of experiencing minimal latency. Alternatively, it can operate in a pipelined strategy, where tasks are parallelized across both threads, at the cost of slightly increasing the latency in exchange for a higher frame rate.

The system latency is influenced by the HMD hardware and managed through the OpenXR API. The interaction engine timing aligns with OpenXR frame times. Ideally, when meeting the frame time budget in synchronous mode, the actual system latency closely matches the frame duration. However, the perceived user latency should be even lower (ideally zero), as the OpenXR runtime predicts the pose of the user's hands and head at the future frame presentation time. When the frame time isn't met, the OpenXR runtime activates motion reprojection to ensure the user experiences smooth movement without frame drops. In the pipelined mode (not used for this work), the system latency increases by one frame time. Thus, the perceived latency in the pipelined mode could be reduced by introducing an additional prediction (outside of OpenXR) for the camera and hands poses, e.g. using a Kalman filter.

4 Stereoscopic Rendering

The classical approaches for stereoscopic volume rendering [1, 8] require: (i) to select an appropriate viewpoint position (e.g., left-eye, right-eye, or the middle point between the left-eye and the right-eye; (ii) apply the chosen volume rendering scheme for the selected viewpoint; (iii) re-project the results or intermediate results

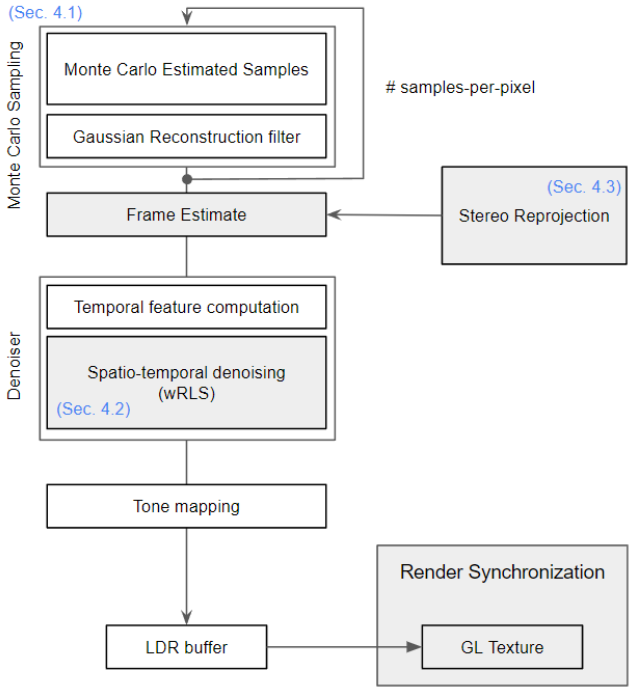

Figure 3: Stereoscopic VPT rendering framework. Our stereo reprojection (SR) reprojects the frame estimate among both eyes virtually improving the sample count of the MC estimates feed into a weighted Recursive Least Squares (wRLS) denoiser operating in real-time to produce the final output for each eye of our rendering framework.

of the previous step onto the right-image or left-image or both, as it may correspond. Thus, reprojection has been one of the typical approaches used to save computation and amortizing the cost of one view into the other.

However, in VPT it is really challenging to successfully reconstruct a whole frame from its stereo counterpart without introducing noticeable bias. Thus our selected approach is to compute a frame estimate with at least one sample-per-pixel for each separate eye and leverage per-pixel sample reprojection to increase the virtual sample count (understood as the equivalent number of samples being used to estimate one pixel of the image). Our aim is to find a reasonable trade-off to harness the available computational resources while mitigating potential sources of cybersickness symptoms (excessive MC variance) or introducing undesirable bias.

Therefore, our stereoscopic rendering can be summarized in three main stages: i) estimating of the contribution of one light path, as an approximation of the volume rendering integral for each eye; ii) performing the stereoscopic reprojection among both eyes; and iii) running a real-time spatio-temporal denoising. For our approach, we will leverage a VPT state-of-the-art denoising based on weighted recursive least squares (wRLS) [10] as it suits well for VPT and the strict real-time requirements of our rendering framework.

4.1 Stereoscopic VRI

The volume rendering integral (VRI) can be approximated by computing only a small number of light transport trajectories (*light paths*) in a single frame. So, the j-th pixel color in the rendered image I can be represented as the following integral:

$$I_j = \int_{\mathscr{P}} f_j(\bar{\mathbf{x}})\, \mathrm{d}\bar{\mathbf{x}}, \tag{1}$$

where $\mathscr{P}$ is the space of all possible light paths in the scene, and $f_j(\bar{\mathbf{x}})$ is the contribution of light path $\bar{\mathbf{x}}$. A common approach for constructing a light transport path $\bar{\mathbf{x}} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_k) \in \mathscr{P}$ is to start from the camera at point $\mathbf{x}_0$ and extend the path incrementally segment by segment. When rendering stereoscopic images, each eye will produce a different light path.

In the context of stereoscopic rendering, we simply extend the concept of the volume rendering integral (VRI) to account for the generation of separate views for each eye. Each eye produces a different light path, resulting in two distinct values, denoted as I_j^{left} and I_j^{right}, respectively. These values represent the perceived colors at the j-th pixel location for the left and right eyes. In the context of VR, there is generally no chance to compute more than one light path per eye in one frame. Thus, a frame estimate is usually generated based on just one sample per pixel.

4.2 Spatio-temporal denoising

To mitigate the MC variance noise we model the ground truth color I_j at the j-th pixel using the following linear regression (corresponding I to either I^{left} or I^{right} for simplified notation):

$$I_j = \mathbf{p}_j \beta_j^T + \xi_j, \tag{2}$$

where $\mathbf{p}_j$ and β_j represent the input predictor vector and its coefficients, respectively. ξ_j represents the prediction error of the linear regression model $\mathbf{p}_j \beta_j^T$. For brevity's sake, we shall treat the value I_j as a scalar unless otherwise mentioned, since our denoising is applied to each color channel independently. Also note that the linear model represents an approximation of the integral (Equation 1) over all light path contributions $f(\bar{\mathbf{x}})$ for the j-th pixel.

Ideally, for a given pixel j at frame t, we would compute the real error by using the values of the ground truth image I_j, as $e_j(t) = I_j(t) - \hat{I}_j(t)$. Because the ground truth of the MC integral is not available in practice, we need to estimate this error using the noisy MC estimate $\tilde{I}_j(t)$ as the following:

$$\hat{e}_j(t) = \tilde{I}_j(t) - \hat{I}_j(t) = \tilde{I}_j(t) - \mathbf{p}_j(t) \beta_j^T (t-1), \tag{3}$$

where $\mathbf{p}_j(t)$ is the predictor vector concatenating the auxiliary features $\mathbf{z}_j(t)$. Similarly, for stereoscopic rendering, we can refer to the error of each separate image. Thus, after computing each separate image error $\hat{e}_j(t)$, the model coefficients β_j are incrementally updated at time t [10]:

$$\beta_j(t) = \beta_j(t-1) + \mathbf{q}_j^w(t) \hat{e}_j(t) \tag{4}$$

$$\mathbf{q}_j^w(t) = \frac{\mathbf{P}_j(t-1)\mathbf{p}_j^T(t)}{\frac{\lambda}{w_j(t)} + \mathbf{p}_j(t)\mathbf{P}_j(t-1)\mathbf{p}_j^T(t)}, \tag{5}$$

being $\mathbf{P}_j$ the inverse covariance matrix of the predictor vectors, λ the forgetting factor, and $w_j(t)$ a weight inversely proportional to temporal sample variance, as described in wRLS [10].

In our framework, we run in parallel, for both eyes, the aforementioned linear model regression updates simultaneously. As temporal denoising auxiliary features ($\mathbf{p}_j$) we maintain a stereoscopic temporal accumulation history buffer, independent for each eye.

4.3 Stereoscopic reprojection

Before running the denoising filter, our rendering framework performs our key contribution, a stereoscopic reprojection in order to virtually increase the effective number of samples per pixel in the frame estimates (as shown in Fig. 3).

While the conventional stereoscopic reprojection would involve reprojecting final pixel colors, our method reprojects intermediate MC sample estimates among both eyes, just before running the wRLS denoising filter. This way, our framework can exploit the spatial coherence more robustly, since the rendering threads are synchronized among the left and right eye in real-time. Moreover, as linear regression can predict single-frame changes in the VPT, e.g., changes in the dynamic lighting, or transfer function editing, our stereoscopic framework can react immediately to abrupt changes in shading.

We estimate per-pixel reprojection vectors υ_j (i.e., similar to optical flow, but for stereoscopic reprojection) using the view-projection matrix and per-pixel world coordinates of both eyes. Our VPT rendering engine internally stores buffers with nearest and average world position coordinates and voxel normals for every pixel in the image for both current (t) and previous $(t-1)$ frames.

Once a reprojection vector υ_j for the j-th pixel is calculated, we can define a reprojection operation π that obtains the corresponding pixel coordinates q in the adjacent frame (from the other eye) as $q \leftarrow \pi(\upsilon_j)$. While previous approaches (like the original wRLS [10]) already use q to perform reprojection (temporal reprojection in their case), we further improve such initial reprojection coordinates estimating new ones after applying dual motion vectors for occlusions [38]. Compared with traditional reprojection vectors, dual motion vectors (or dual reprojection vectors) allowed us to more accurately estimate reprojection coordinates, improving our final reprojection result. Implementing dual motion vectors requires a few additional computations to prevent incorrect reprojections (e.g. caused by occlusions or disocclusions), but represents a negligible execution cost when compared to the overall timing of the denoising process.

To make our stereoscopic reprojection more robust, and mitigate incorrect reprojections, we perform further tests to confirm whether the reprojected information from the neighbor-eye corresponds to a relatively similar world position and shares a similar normal orientation to the samples in the target destination of the reprojection. Consequently, we establish conservative thresholds to discard distinct world positions or dissimilar normal orientations. For accepted reprojections, source and target estimates are blended with equal weight. If reprojection was previously discarded, we just use the initially computed color without blending.

5 DVR INTERACTIVE TOOLS AND 3D USER INTERFACE

One of the big advantages of immersive VR is that users can naturally change the point of view by freely moving in 3D space thanks to the head positional and orientation tracking provided by the HMD hardware. Also, manipulating and rotating the 3D volume under study can be done with great precision thanks to the hand controllers typically accompanying the HMD devices, hand-tracking solutions or a combination of both. In this system we focused initially on testing hand-tracker controllers given their great tracking accuracy, however we do not discard hand-tracking as a perfectly viable alternative in the near future.

In the following, we will describe some of the tools we implemented in our system to favor the interaction in VR with photorealistic 3D volume visualizations. We highly recommend consulting our supplementary video to see the actual visualization system in action and how these tools could be used as part of our system.

5.1 Dynamic global illumination

Since our stereoscopic rendering is based on a Monte Carlo volumetric path tracer, dynamic global illumination is one of the main advantages we have in our system. We can simulate multiple types of light sources (casting both hard or soft shadows), and translucency and scattering effects. In Fig.4, we show an example of the versatile lighting configurations our system can reproduce under user interactions.

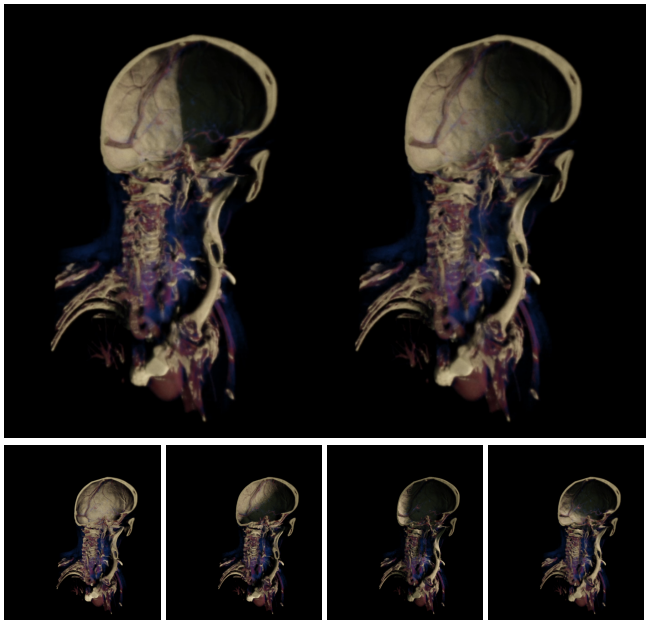

Figure 4: Dynamic lighting and shadowing. (Top) Example of adjusting a light source to produce hard or soft shadows. (Bottom) Example of adjusting lighting orientation with respect to the volume data. These lighting properties can be interactively changed using the VR interface.

Light sources in our system can be manipulated by just selecting them with one controller, grabbing and moving them as desired to reposition the lighting direction. Light sources are locked to aim towards the center of the volume, so the user simply changes the angle of the light by placing it in 3D space. Our system also supports adjusting both the size and color of light sources.

5.2 Transfer-function editing

The editing of the transfer function can be done in different ways. The highest level of control can be reached by directly editing the curve, usually by placing and moving control points and defining the ways to interpolate between these points. Control points have associated parameters defining the appearance of the volume render, such as opacity, diffuse color, roughness, specular color, or emission properties.

Editing the transfer function with such a level of control may be a complex operation even for medical imaging professionals. The typical use case is a combination of selecting presets, i.e. transfer functions predefined for a specific kind of volume and intended to show certain features (skin tissue, bone, muscle...), and then modify those presets with simpler and more direct actions like the intensity windowing (IW) [32].

In VR, editing the curve by setting control points could be a tedious operation, so we implemented IW, that fits perfectly with VR interaction. The user can select a preset, visualize the histogram of the volume density values with the transfer function overlaid on it and then modify the curve by performing ulterior scaling and/or offsetting by just moving the hand controller vertically or horizontally in 3D space while holding the trigger (Fig. 5). We show several sequences in our supplementary video where users manipulate the transfer function in real time.

5.3 Volume clipping operations

One essential operation when exploring medical datasets is the ability to perform cuts within the volume to identify the region of interest

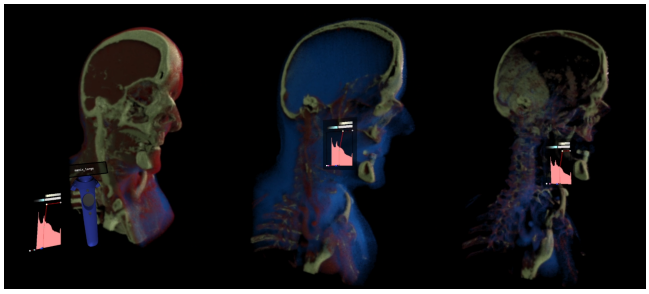

Figure 5: Interactive transfer-function editing in VR.

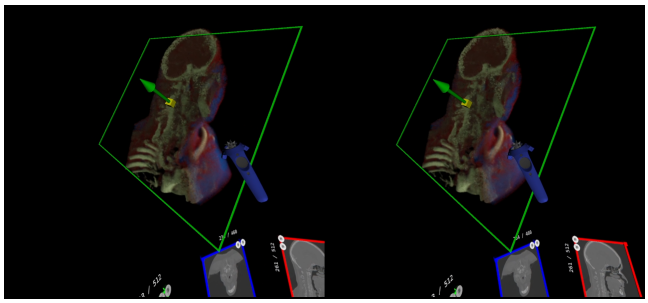

Figure 6: Arbitrary clipping plane using a 3D oriented plane gizmo.

(ROI). These cuts can be either axis-aligned or arbitrarily oriented, and our system accommodates both types of clipping planes.

For arbitrary-oriented clipping planes, we designed a 3D gizmo that takes the form of a transparent rectangle with a solid frame (Fig. 6). At its center, there's a guiding arrow indicating the direction of the cut. The user can easily interact with this gizmo by grabbing it, either directly or from a distance, and effortlessly positioning it with a simple hand gesture. This intuitive approach has proven to be highly effective and precise, surpassing the efficiency of traditional 2D mouse-based interfaces when defining arbitrary clipping planes.

In addition to arbitrary clipping planes, our system also supports axis-aligned clipping planes that can be enabled or inverted along all three spatial axes in world space. However, the interaction method for axis-aligned clipping planes differs slightly. These planes are associated with 2D viewers that allow users to conveniently manipulate and configure their positions, providing a comprehensive and versatile solution for working with various types of clipping planes within the VR environment.

5.4 Immersive 2D viewer

Radiologists typically refer to 2D images in their routine work. Thus 3D visualizations are great additions to their decision-making process. However, it is mandatory for them to be able to operate and access simultaneously the different anatomical plane views of a medical scan (i.e. axial, sagittal, and coronal) and interact with the 3D volume based on that information.

We implemented a floating window VR widget to display the 2D slices of the raw volumetric dataset. The 2D viewers are synchronized with the axis aligned clipping planes that can be applied to the 3D volume, and allow users to easily enable/disable and relocate them, as illustrated in Fig. 7.

6 Results and Evaluation

6.1 Hardware and medical datasets

We focus our evaluation on two main aspects. First, by studying the performance and quality enhancement of using our stereoscopic

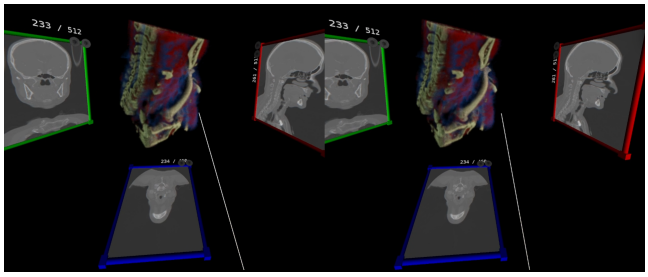

Figure 7: Immersive 2D slice viewers synchronized and simultaneously cutting the volumetric reconstruction.

denoising technique for HMD volume rendering. Second, we showcase and demonstrate the interaction and rendering capabilities of our immersive 3D visualization system by interactively exploring medical datasets in real-time.

Our benchmark system is powered by a Ryzen 9 5950X CPU, 128 GB DDR4 memory, and an Nvidia RTX 3090 Ti GPU driving our immersive 3D visualization system. For the VR hardware, we use the HTC Vive Pro Eye HMD, featuring a combined display resolution of 2560×1440 pixels. We aimed to reach the refresh rate of 90 Hz, currently found in most high-end HMD systems, to run our interactive 3D visualization on moderately-sized CT scans.

As medical datasets for our evaluation, we chose MANIX, a CT scan of a human head with a resolution of $512 \times 512 \times 460$ voxels, HAND, a CT scan of a human hand with a resolution of $512 \times 512 \times 496$ voxels, and MACOESSIX, a lower limbs CT scan with $512 \times 461 \times 512$ voxels. To guarantee that each rendering technique under evaluation uses exactly the same input data, we captured the actions of a real user interacting with our VR system. This way, we can fairly compare all techniques. We obtained as result three pre-recorded sequences, one per dataset. All the transfer functions used show translucent or semitransparent structures, to stress on purpose the reprojection on volumetric data, as opposed to reprojection on surface models. All images were rendered at a per-eye resolution of 1424×1584 pixels. The reference images were rendered offline at 2048 spp taking each > 12 sec. to render.

6.2 Benchmarking denoising of stereoscopic VPT

We perform benchmark comparisons both in terms of image quality metrics and runtime execution time. As quality metrics we report root mean square error (RMSE) and a perceptual dissimilarity metric [36] (DSSIM) for all frames in the aforementioned pre-recorded sequences, one per dataset: HAND, MANIX, and MACOESSIX. In Fig. 8 we show a visual comparison using frames randomly chosen from these sequences[1], while the error plots over the whole sequences are shown in Fig. 9. Next, we discuss the results of the following techniques: 1 spp and 4 spp with MC-VPT; 1 and 4 spp with the adaptive temporal reprojection (ATR) proposed by Martschinke et al. [21] [2]; 1 spp denoised with weighted recursive least squares (wRLS) [10] running independently for each eye (EID); and our extended version of wRLS denoising using stereoscopic reprojection, named as SR (Ours).

Discussion. The obtained results vary depending on the dataset under evaluation, the specific transfer function, and the type of background and lighting conditions. From the three datasets, MANIX and MACOESSIX demonstrated to be more challenging than HAND, probably due to the presence of thinner 3D structures that prevented a more frequent use of the reprojected information. Specifically for these cases, SR (Ours) seems to perform consistently better with both metrics (RMSE and DSSIM) than any other real-time technique. To disambiguate when the two best methods EID and SR perform similar in terms of quality metrics, we recorded a supplementary video with a temporal flicker comparison. In that comparison, SR remains temporally more stable than EID, which makes sense given the higher virtual sample count used by SR. The ATR technique showed remarkable temporal stability, mostly for surface-like regions of the volume, but it fails to reduce flicker for pixels that suffer a hard reset of their history buffer, as it can be noticed in the temporal flicker comparison in our supplemental (better seen for MANIX and MACOESSIX). Also, that stability comes at the trade-off of a per-frame lower quality, as evidenced in all our temporal plots (Fig. 9). One drawback noticed for ATR is that it has some latency when editing the lighting conditions, as their heuristic for reprojection is based exclusively on per-pixel depth and ray directions, that do not capture changes in shading. For all these hard cases in ATR, both EID and SR offer a better reconstruction and a temporally more stable solution. This behavior is quite desirable in VR/XR, as temporal flicker is known to be quite disturbing when displayed on HMD devices. Thus, we conclude that SR offers nowadays a more sustainable performance for all type of scenarios.

[1] highlighted using a vertical line in Fig. 9

[2] the 1 spp version runs in real-time, the 4 spp is an upper bound of their best possible quality when running their adaptive sampling with max. 4 spp.

Dataset	VPT 1spp	VPT 2spp	VPT 4spp	ATR 1spp	EID 1spp	SR 1spp	2K spp
HAND	10.69 ms	22.92 ms	39.51 ms	21.14 ms	18.91 ms	18.88 ms	21.5 s
MANIX	8.22 ms	15.36 ms	29.57 ms	16.90 ms	16.06 ms	16.16 ms	13.8 s
MACOESSIX	10.02 ms	19.05 ms	36.73 ms	19.49 ms	17.99 ms	18.11 ms	18.6 s

Table 1: Average execution times.

6.3 Interactive DVR operations

We demonstrate the capabilities of our immersive 3D medical visualization system under different types of user interactions, akin DVR operations and their functionality. We run multiple live sessions using the system and exploring different medical datasets. We strongly recommend watching our supplementary video where we demonstrate the suitability of the proposed technique for its use in immersive VR experiences for medical visualization, enabling photorealistic rendering at interactive rates on a commodity PC platform.

6.4 Performance evaluation and discussion

Table 1 shows the average running times of the VPT rendering system when used with an HMD setup for VR. The first obvious conclusion is that simply increasing the number of MC samples is not a practical alternative, as increasing the number of MC samples results in a linear growth in rendering time, rapidly exceeding the available budget in VR. Moreover, the attained image quality of MC-VPT is greatly surpassed by using adaptive temporal reprojection (ATR) or denoising approaches (EID and SR). This justifies our choice of limiting our sampling budget to 1 spp and aims for an efficient denoising in real-time.

When comparing ATR, the previous wRLS (EID) and the enhanced stereoscopic reprojection (SR), running times are still very similar, as most of the time is consumed by MC-VPT rendering (typically $> 50-60\%$). The cost of the denoising in both cases (EID and SR) ranges from 7.84 to 8.22 ms, which, added to the VPT render time with 1spp, lies close to the render time of 2 spp. Renders with 4 spp nearly double the time of MC-VPT with 1 spp plus SR denoising, and still remain behind ATR or the other denoising approaches. Thus, given the the discussed trade-offs of each technique regarding their temporal flicker, and the relatively small performance overhead, we conclude that adopting SR is the recommended solution for our interactive 3D medical visualization system in VR.

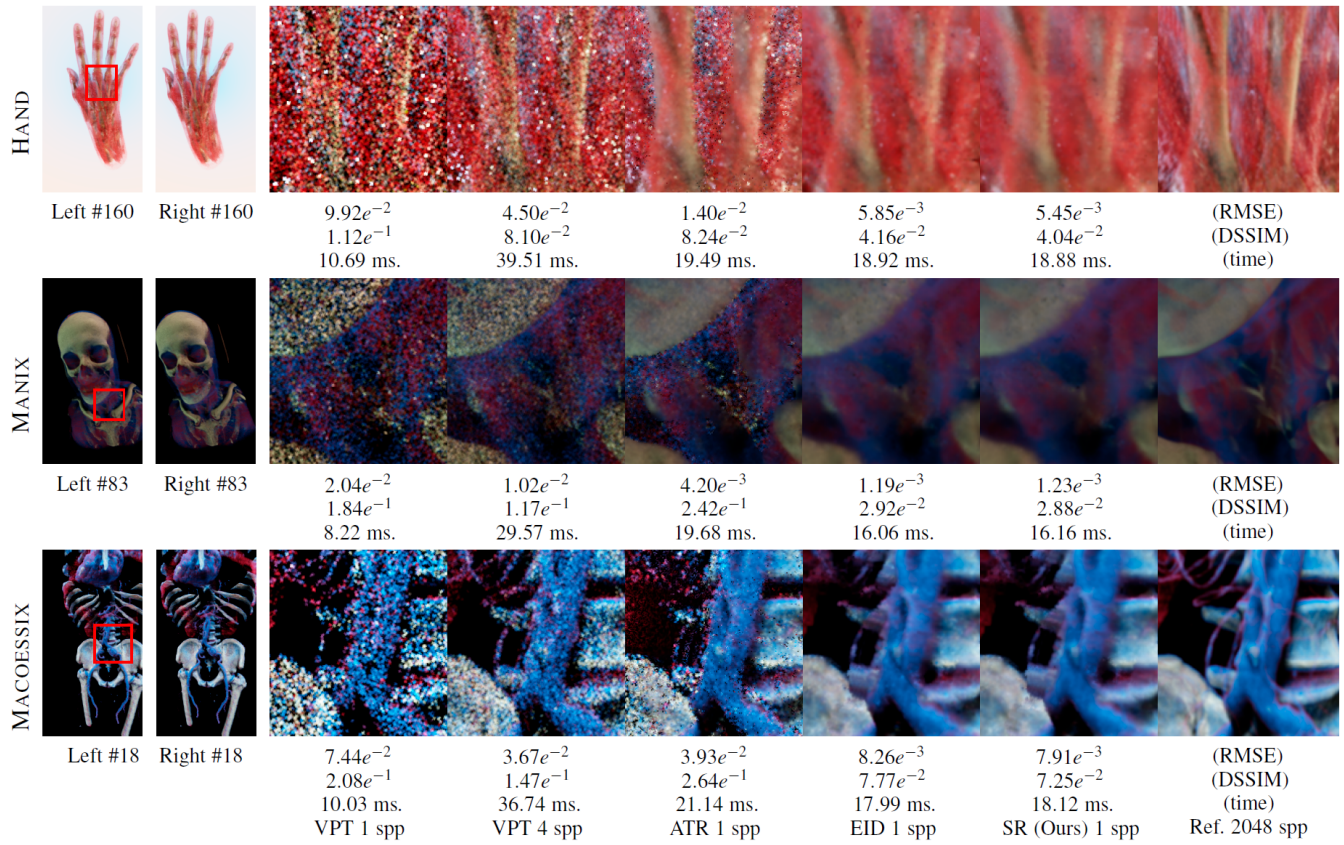

Figure 8: Benchmark comparison in terms of image quality (RMSE and DSSIM) and runtime performance (ms.). We compare our stereoscopic denoising approach (SR) to Monte Carlo VPT with 1 and 4 spp per-eye, the adaptive temporal reprojection (ATR) scheme by Martschinke et al. [21], and also with running a per-eye independent denoising based on wRLS [10] (EID). Our reference image for comparisons took more than 12 sec. per frame to compute with 2048 spp.

7 Conclusions and Future Work

This paper presented an immersive 3D medical visualization system enabling VPT for stereoscopic virtual reality. Our approach combined stereoscopic rendering with state-of-the-art image reprojection and denoising techniques to meet the requirements of VR experiences. We extended previous real-time linear regression denoising (wRLS) for stereo reprojection by leveraging on more precise dual motion vectors. Our denoising mitigates the disturbing flicker introduced by Monte Carlo noise, and its undesired side effects that harm the overall VR experience.

We also showcased a practical virtual reality implementation of our visualization system in the medical domain, demonstrating feasible interactive rendering and appearance manipulation for 3D medical data operating in real-time. Our approach addresses common interactive operations for DVR within our VR immersive system, achieving unprecedented quality for interactive cinematic rendering within PC commodity VR settings.

Our system does not come without limitations. First, the limited performance of the combined rendering and denoising, which still leverages on OpenXR motion reprojection to achieve the required 90 Hz in VR; Second, the residual flickering artifacts introduced by sparse linear model reset operations caused by having a 1 spp input signal to the linear models. We believe that further investigation of joint adaptive temporal reprojection (ATR) and linear regression denoising (wRLS) is worth exploring, since an hybrid approach could potentially exploit the best of each approach to tackle the remaining limitations. As future work, we would like to further optimize our system based on foveated rendering and perceptual observations of the human visual system, i.e., preattentiveness [15, 22]. Finally, we would be very interested in developing user studies to verify the adequacy of the proposed rendering and denoising techniques, as well as assessing better designs for our VR user interface gathering more feedback together with medical experts.

Acknowledgments

The anonymized medical datasets employed in this study were obtained courtesy of the OsiriX Foundation (specifically, MANIX and MACOESSIX), and were freely available through Embodi3D LLC (HAND). We thank Prof. Jérôme Schmid for the helpful and insightful discussions about the usability of the system by healthcare professionals. We also thank Prof. Bochang Moon for our helpful discussions on image-based denoising using weighted recursive least squares. Both authors would like to express their gratitude with Tom Boudard, Enrique García, Cristina Pelayo and Miguel Osorio who all helped us improve the VR GUI and tools of the system. We also thank Manuel Silva for his last-minute help in preparing the supplementary website, and Bea Blanco for her administrative assistance. We also thank Giovanni Fiorilli for his support in building our HTC Developer Partnership. Jose A. Iglesias-Guitian was supported by a 2021 Leonardo Grant for Researchers and Cultural Creators, BBVA Foundation. He also acknowledges the UDC-Inditex InTalent programme, the Spanish Ministry of Science and Innovation (AEI/RYC2018-025385-I) and Xunta de Galicia (ED431F 2021/11).

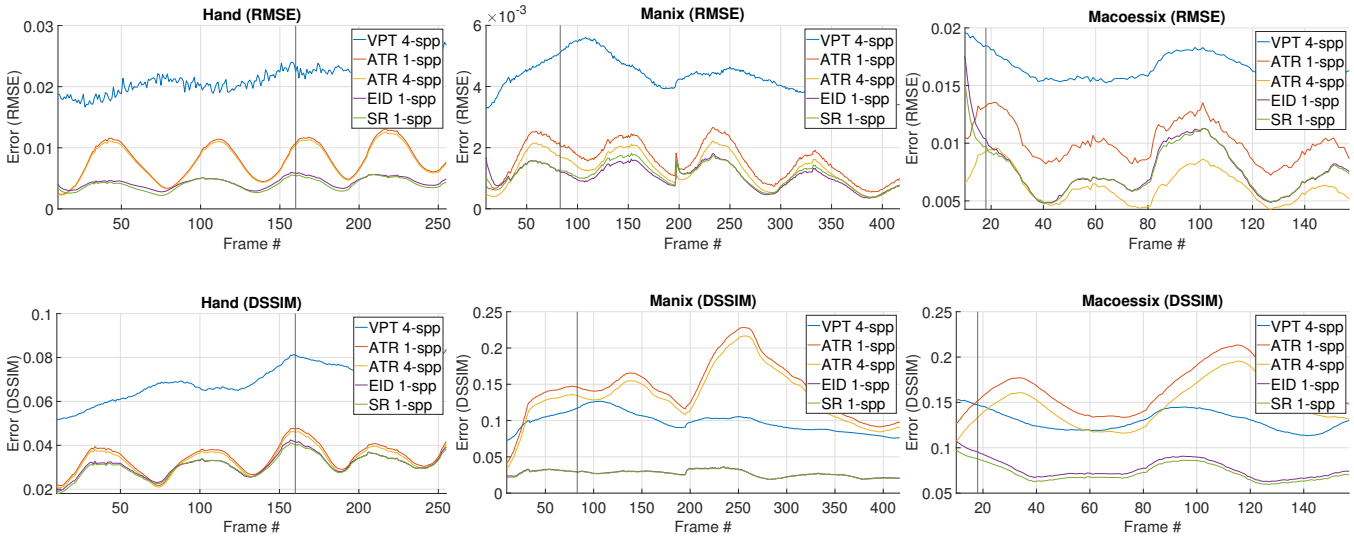

Figure 9: Temporal plots showing RMSE and DSSIM metrics over the whole sequences used to evaluate our results in this paper. We compare VPT 4 spp, ATR with 1 and 4 spp, per-eye independent denoising using wRLS (EID) and our stereoscopic denoising approach (SR).

REFERENCES

[1] S. J. Adelson and C. D. Hansen. Fast stereoscopic images with ray-traced volume rendering. In *Proceedings of the 1994 symposium on Volume visualization*, pp. 3–9, 1994.

[2] M. Agus, E. Gobbetti, J. A. Iglesias Guitián, F. Marton, and G. Pintore. Gpu accelerated direct volume rendering on an interactive light field display. In *Computer Graphics Forum*, vol. 27, pp. 231–240. Wiley Online Library, 2008.

[3] C. Cruz-Neira, J. Leigh, M. Papka, C. Barnes, S. M. Cohen, S. Das, R. Engelmann, R. Hudson, T. Roy, L. Siegel, et al. Scientists in wonderland: A report on visualization applications in the cave virtual reality environment. In *Proceedings of 1993 IEEE research properties in virtual reality symposium*, pp. 59–66. IEEE, 1993.

[4] J. Egger, M. Gall, J. Wallner, P. Boechat, A. Hann, X. Li, X. Chen, and D. Schmalstieg. Htc vive mevislab integration via openvr for medical applications. *PloS one*, 12(3):e0173972, 2017.

[5] M. Elshafei, J. Binder, J. Baecker, M. Brunner, M. Uder, G. F. Weber, R. Grützmann, and C. Krautz. Comparison of Cinematic Rendering and Computed Tomography for Speed and Comprehension of Surgical Anatomy. *JAMA Surgery*, 154(8):738–744, 08 2019. doi: 10.1001/jamasurg.2019.1168

[6] K. Engel, M. Hadwiger, J. Kniss, C. Rezk-Salama, and D. Weiskopf. *Real-time volume graphics*. AK Peters/CRC Press, 2006.

[7] F. A. Fellner, K. Engel, and C. Kremer. Virtual anatomy: The dissecting theatre of the future—implementation of cinematic rendering in a large 8 k high-resolution projection environment. *Journal of Biomedical Science and Engineering*, 10(8):367–375, 2017.

[8] T. He and A. Kaufman. Fast stereo volume rendering. In *Proceedings of Seventh Annual IEEE Visualization'96*, pp. 49–56. IEEE, 1996.

[9] N. Hofmann, J. Martschinke, K. Engel, and M. Stamminger. Neural denoising for path tracing of medical volumetric data. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 3(2):1–18, 2020.

[10] J. Iglesias Guitián, P. Mane, and B. Moon. Real-time denoising of volumetric path tracing for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 28(7):2734–2747, 2022. doi: 10.1109/TVCG.2020.3037680

[11] J. A. Iglesias Guitián, E. Gobbetti, and F. Marton. View-dependent exploration of massive volumetric models on large-scale light field displays. *The Visual Computer*, 26:1037–1047, 2010.

[12] S. Jadhav and A. E. Kaufman. Md-cave: An immersive visualization workbench for radiologists. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[13] D. Jönsson, E. Sundén, A. Ynnerman, and T. Ropinski. A survey of volumetric illumination techniques for interactive volume rendering. In *Computer Graphics Forum*, vol. 33, pp. 27–51. Wiley Online Library, 2014.

[14] R. Khlebnikov, P. Voglreiter, M. Steinberger, B. Kainz, and D. Schmalstieg. Parallel irradiance caching for interactive monte-carlo direct volume rendering. In *Computer Graphics Forum*, vol. 33, pp. 61–70. Wiley Online Library, 2014.

[15] A. Krekhov, S. Cmentowski, A. Waschk, and J. Krüger. Deadeye visualization revisited: Investigation of preattentiveness and applicability in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):547–557, 2019.

[16] T. Kroes, F. H. Post, and C. P. Botha. Exposure render: An interactive photo-realistic volume rendering framework. *PloS one*, 7, 2012.

[17] J. J. LaViola Jr. A discussion of cybersickness in virtual environments. *ACM Sigchi Bulletin*, 32(1):47–56, 2000.

[18] F. Lindemann and T. Ropinski. About the influence of illumination models on image comprehension in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1922–1931, 2011.

[19] N. Liu, D. Zhu, Z. Wang, Y. Wei, and M. Shi. Progressive light volume for interactive volumetric illumination. *Computer Animation and Virtual Worlds*, 27(3-4):394–404, 2016.

[20] L. C. Loschky, S. Szaffarczyk, C. Beugnet, M. E. Young, and M. Boucart. The contributions of central and peripheral vision to scene-gist recognition with a 180 visual field. *Journal of Vision*, 19(5):15–15, 2019.

[21] J. Martschinke, S. Hartnagel, B. Keinert, K. Engel, and M. Stamminger. Adaptive temporal sampling for volumetric path tracing of medical data. In *Computer Graphics Forum*, vol. 38, pp. 67–76. Wiley Online Library, 2019.

[22] X. Meng, R. Du, and A. Varshney. Eye-dominance-guided foveated rendering. *IEEE Transactions on Visualization and Computer Graphics*, 26(5):1972–1980, 2020.

[23] B. Moon, J. A. Iglesias-Guitian, S.-E. Yoon, and K. Mitchell. Adaptive rendering with linear predictions. *ACM Trans. on Graphics (TOG)*, 34(4):121, 2015.

[24] A. Munawar, Z. Li, N. Nagururu, D. Trakimas, P. Kazanzides, R. H. Taylor, and F. X. Creighton. Fully immersive virtual reality for skull-base surgery: surgical training and beyond. *International Journal of Computer Assisted Radiology and Surgery*, pp. 1–9, 2023.

[25] J. Novák, I. Georgiev, J. Hanika, and W. Jarosz. Monte carlo methods

for volumetric light transport simulation. In *Computer graphics forum*, vol. 37, pp. 551–576. Wiley Online Library, 2018.

[26] M. L. Pachowsky, H. Morf, D. Simon, V. Schönau, L. Valor-Mendez, J. Knitza, F. Fagni, K. Engel, M. Uder, A. Hueber, et al. Cinematic rendering in rheumatic diseases—photorealistic depiction of pathologies improves disease understanding for patients. *Frontiers in Medicine*, 9:946106, 2022.

[27] B. Preim and P. Saalfeld. A survey of virtual human anatomy education systems. *Computers & Graphics*, 71:132–153, 2018.

[28] T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. Hinrichs. Interactive volume rendering with dynamic ambient occlusion and color bleeding. In *Computer Graphics Forum*, vol. 27, pp. 567–576. Wiley Online Library, 2008.

[29] S. P. Rowe, J. Fritz, and E. K. Fishman. Ct evaluation of musculoskeletal trauma: initial experience with cinematic rendering. *Emergency radiology*, 25:93–101, 2018.

[30] C. R. Salama. Gpu-based monte-carlo volume raycasting. In *15th Pacific Conference on Computer Graphics and Applications (PG'07)*, pp. 411–414. IEEE, 2007.

[31] I. Scholl, A. Bartella, C. Moluluo, B. Ertural, F. Laing, and S. Suder. Medicvr: Acceleration and enhancement techniques for direct volume rendering in virtual reality. In *Bildverarbeitung für die Medizin 2019: Algorithmen–Systeme–Anwendungen. Proceedings des Workshops vom 17. bis 19. März 2019 in Lübeck*, pp. 152–157. Springer, 2019.

[32] J. Z. Turlington and W. E. Higgins. New techniques for efficient sliding thin-slab volume visualization. *IEEE Transactions on Medical Imaging*, 20(8):823–835, 2001.

[33] W. Usher, J. Amstutz, C. Brownlee, A. Knoll, and I. Wald. Progressive cpu volume rendering with sample accumulation. In *EGPGV@ EuroVis*, pp. 21–30, 2017.

[34] 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*, 24(1):994–1003, 2017.

[35] P. von Radziewsky, T. Kroes, M. Eisemann, and E. Eisemann. Efficient stochastic rendering of static and animated volumes using visibility sweeps. *IEEE Transactions on Visualization and Computer Graphics*, 23(9):2069–2081, Sep. 2017. doi: 10.1109/TVCG.2016.2606498

[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[37] A. Waschk and J. Krüger. Favr-accelerating direct volume rendering for virtual reality systems. In *2020 IEEE Visualization Conference (VIS)*, pp. 106–110. IEEE, 2020.

[38] Z. Zeng, S. Liu, J. Yang, L. Wang, and L.-Q. Yan. Temporally reliable motion vectors for real-time ray tracing. In *Computer Graphics Forum*, vol. 40, pp. 79–90. Wiley Online Library, 2021.

[39] Y. Zhang, Z. Dong, and K.-L. Ma. Real-time volume rendering in dynamic lighting environments using precomputed photon mapping. *IEEE Transactions on Visualization and Computer Graphics*, 19(8):1317–1330, 2013.