Materializing Inter-Channel Relationships with Multi-Density Woodcock Tracking

Alper Sahistan[*], Stefan Zellmann[†], Haichao Miao[‡], Nate Morrical[§], Ingo Wald[§], and Valerio Pascucci[*]

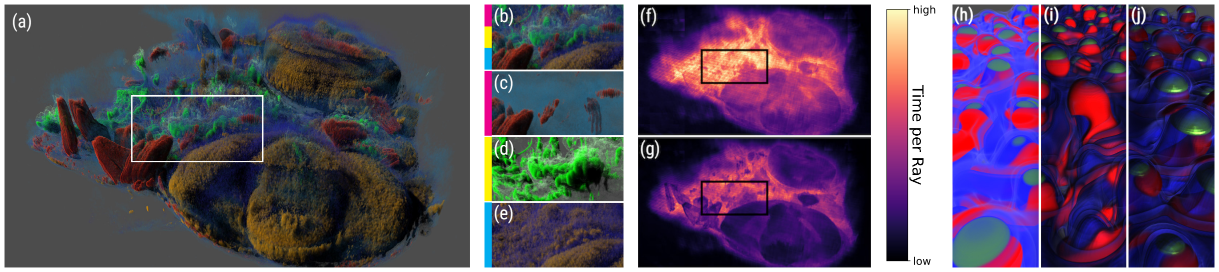


Fig. 1: Multi-channel volume renderings for two datasets. Zebrafish dataset ($640 \times 640 \times 121$) shown in (a–g): (a) converged rendering at 2560×1440 with 100 FPS using our multi-density Woodcock tracking. (b) Crop of (a) with individual channels shown in (c–e). (f) Heatmap of per-ray cost when sampling all channels independently versus (g) our amortized multi-density approach. Demonstration of gradient-based shading over the Miranda dataset ($384 \times 384 \times 256$) in (h–j): (h) emission+absorption only, (i) shaded with the same material properties, and (j) using our per-channel material shading system that assigns different diffuse and specular properties per channel.

Abstract

Volume rendering techniques for scientific visualization have recently shifted toward Monte Carlo (MC) methods for their flexibility and robustness, but their use in multi-channel visualization remains underexplored. Traditional multi-channel volume rendering often relies on arbitrary, non-physically based color blending functions that hinder interpretation. We introduce multi-density Woodcock tracking, a simple extension of Woodcock tracking that leverages an MC method to produce high-fidelity, physically grounded multi-channel renderings without arbitrary blending. By generalizing Woodcock's distance tracking, we provide a unified blending modality that also integrates blending functions from prior works. We further implement effects that enhance boundary and feature recognition. By accumulating frames in real-time, our approach delivers high-quality visualizations with perceptual benefits, demonstrated on diverse datasets.

Ray tracing, Volume Rendering, Scientific visualization, Monte Carlo Methods

1 Introduction

From simulations to microscopy images, visualization of scientific data is essential for domain scientists to draw informed conclusions quickly and accurately. Modern scientific data are typically multifaceted, with each aspect

[*]SCI Institute, University of Utah, Salt Lake City, Utah, USA.
[†]University of Cologne, Germany.
[‡]Lawrence Livermore National Laboratory, Livermore, California, USA.
[§]NVIDIA Corporation.

represented as a distinct *channel* (field) corresponding to a specific attribute (such as velocity, temperature, or pressure) of the phenomena within a volume. Understanding causality and the relationships between these attributes requires examining them not only in isolation but also in relation to one another. Multi-channel visualization offers a viable solution, providing a practical approach for observing the intricate connections among different fields in the data.

Volume rendering techniques have evolved from analytical compositing-based methods—such as slicers [1] and ray marching [2]—to Monte Carlo-based tracking solutions. Unlike compositing methods, which combine partial samples via *opacity* [3], tracking approaches interpret opacity as a physical *density*, allowing for a single collision per ray. A prominent technique in this domain is Woodcock tracking [4]. These Monte Carlo methods are generally easy to implement and provide greater interactivity and flexibility, but their application to multi-channel scientific volumes remains largely unexplored.

Multi-channel volume rendering of multiple fields comes with additional challenges in color *blending*. Unlike single-channel methods, rendering N fields involves N opacities that can sum to over 100%, complicating calculation of color contributions. Standard approaches normalize opacities and blend with respect to the maximum opacity, compositing over previously accumulated color. However, these methods can cause color discontinuities and out-of-colormap colors, and increase the computational cost ($\Theta(N)$) as the number of channels grows. More channels also complicate user interaction [5, 6], making it harder to design blending operators that ensure proper depth ordering and remain intuitive [7].

Despite their limitations, prior blending functions remain useful in certain contexts. For instance, ignoring occlusion can aid when spatial relationships are irrelevant [8], and mixed colors can reveal correlations with primary-color transfer functions [5, 6]. However, no formal method exists to unify these techniques, and doing so becomes especially counterintuitive with compositing-based rendering.

A further overlooked aspect of multi-channel rendering is the use of additional effects. Most existing multi-channel renderers [9] support only basic emission–absorption models [10], leaving effects like shadows or proxy-geometry-based shading underexplored. Whether due to rendering costs or implementation complexity, these features are rarely included—yet, considering their utilization in single-channel visualization [11], they could greatly improve the perception of structures and boundaries within the data.

In this work, following a similar idea to analog decomposition tracking [12], we extend Woodcock tracking to multi-channel scientific visualization (sci-vis), addressing the fundamental challenges of rendering multi-channel data. Our algorithm employs a Monte Carlo process to identify the closest sample by evaluating the density of each channel along the ray, thereby avoiding the limitations of previous methods that rely on accumulating numerous partial samples. Building on our earlier work [13], this work further expands on the application of additional effects, such as shadows and material shading. It broadens the evaluation by incorporating new datasets and additional experiments. As illustrated in Figure 1, our approach enables order-independent, accurate, and efficient visualization of arbitrary combinations of N channels across a volume. Our contributions are:

- Two generalizations of the Woodcock tracking traversal to multi-channel volumes:
  - an easy-to-implement serial method that traverses and collects samples for N channels one at a time,
  - and an optimized method that eliminates redundant steps by traversing all N channels in a single pass.
- A density-driven, physically grounded inter-channel blending function that overcomes the limitations of earlier color blending methods.
- A unified rendering modality within the Woodcock tracking that allows substituting inter-channel color blending functions from prior works—e.g., max intensity projection and weighted mixing.
- Low-cost implementations of additional effects—such as shadows and a per-channel material shading system— tailored for multi-channel rendering.

2 Related Work

2.1 Volume Rendering

Earlier works in volume rendering utilize slicers [14, 15, 1, 16] where a 3D texture of view-aligned slices are re-sampled to approximate the volume rendering equation. Although slicers are actively used in multi-channel volume rendering, single-channel methods largely abandoned slicers with the advent of GPGPU frameworks like CUDA and OpenCL.

Nowadays, the standard way to render volumes is ray marching-based methods [17, 18, 19, 20]. These methods produce high-quality, noise-free images by analytically approximating the volume rendering equation via accumulated contributions from multiple partial samples along a ray's path. However, when undersampling, marchers introduce bias and artifacts even with jittered sampling [21]. Additionally, incorporating effects such as volumetric shadows or gradient shading dramatically hinders rendering performance, as these require more sampling for each initial view-aligned sample [22]. To combat these costs, prior works have proposed space skipping and adaptive sampling strategies [23, 24, 25, 26]. Specifically for shadows, shadow maps offer a solution for achieving more interactive rates, but they occupy extra memory and require recomputation whenever the transfer function or lighting changes [10].

Tracking-based Monte Carlo estimators have gained traction in scientific and cinematic rendering [27]. Despite introducing variance (noise) and needing to converge over multiple samples, they offer advantages such as efficient handling of additional effects like shadows due to their lower asymptotic complexity. They also enable artifact-free adaptive sampling [28, 29]. Szirmay-Kalos et al. further improve efficiency by using a 3D digital differential analyzer (DDA) to adjust sampling rates based on local density estimates stored in a coarser grid.

Woodcock (delta) tracking has become popular in recent works [30, 31, 32, 33], where volumes are homogenized using fictitious particles based on maximal density. Morrical et al. introduce adaptive Woodcock tracking for compressed clusters of unstructured meshes [34]. Zellmann et al. explore data structures and algorithms to efficiently path trace Adaptive Mesh Refinement volumes using Woodcock tracking [35].

The use of tracking methods for rendering multiple overlapping volumes has been explored in prior work [36, 12], predominantly within cinematic rendering. Novak et al.'s residual ratio tracking [36] reduces variance by splitting the volume into homogeneous control and heterogeneous residual volumes. Kutz et al.'s spectral and decomposition tracking [12] introduces the idea of selecting the minimum distance among free-flight samples from multiple volumes, a technique shown to fit within the integral framework of Galtier et al. [37] and later expanded further [38]. In contrast, we utilize these ideas for multiple overlapping heterogeneous volumes within the sci-vis applications.

2.2 Multi-Channel Visualization

The multi-channel visualization has two orthogonal problems: user interface [39] and rendering [40]. In this paper, we primarily address the rendering aspect while allowing the use of traditional transfer functions. Regardless, the solutions to these orthogonal problems can sometimes be intertwined [41]. The work by Kim [7] offers many insights into the multi-channel visualization domain.

Given the tedious nature of setting a precise transfer function for all the channels, Pan et al.'s work [42] concentrates on design galleries where users can select images resembling their target image to construct a transfer function implicitly. Likewise, Kim et al. [43, 44] employ dimensionality reduction schemes to aid in the design of multidimensional transfer functions. There is no reason our work would not be compatible with these transfer function helper frameworks if their neural networks are trained using our renderer.

Khlebnikov et al. [45] propose a random-phase Gabor noise-driven cell-space redistribution pattern and filtering scheme for simultaneous multi-channel display. However, this technique can blur details or create counterintuitive renderings, notably for untrained users. Herzberger et al. [46] introduce the residency Octree for out-of-core mixed resolution in web-based multi-channel rendering, focusing on efficient data streaming with a ray-guided volume renderer from Crassin et al. [47].

FluoRender [48, 49, 9] is one of the most prominent tools for multi-channel visualization, using slicer-based rendering while enabling a variety of user interactions. Slicers render one slice at a time, loading only necessary transfer functions and channels, effectively serializing the rendering and reducing texture memory and VRAM usage. FluoRender users can select, highlight, or segment structures using brushing tools with morphological diffusion, aiding in exploring correlations. To avoid visual clutter from overlapping fields, Fluorender offers non-physically based compositing modes, such as post-render compositing and weight-based color mixing with depth correction. Given Fluorender's broad appeal, we incorporate some of these modes into our Woodcock tracking experiments.

Much like Fluorender, Napari [50] is a multi-channel, opacity-based visualization tool widely used in the biological sciences. It supports multiple blending and compositing modes, making it one of the most versatile publicly available tools for multi-channel visualization. However, Napari presents a trade-off between color-correct and depth-correct blending, as achieving depth accuracy requires the use of maximum intensity projection (MIP).

Opacity-based rendering often relies on physically inaccurate models and struggles with opacity mixing. Previous efforts [8, 5, 6] have addressed some issues with inter-channel color blending. We propose adopting density-based approaches, such as Woodcock tracking, to overcome to tackle these issues in a physically-motivated manner. Density-based approaches shift color blending to screen space, accumulating one sample at a time across frames. Our approach also supports integrating other blending techniques from prior works.

3 Woodcock Tracking Background

In Woodcock tracking [4], photon and particle behaviors are simulated through a Monte Carlo process. Generalizing the rendering equation [51] for volumes and simplifying for the emission/absorption model gives us the simplified Volume Rendering Equation (VRE):

$$L(x,\omega) = \int_{t=0}^{d} T(t)\sigma_a(x)L_e(x_t,\omega)dt. \tag{1}$$

In Equation 1, radiance, $L(x,\omega)$, at point x looking in the direction of ω is calculated by taking the integral of transmittance $T(t)$ times the absorption coefficient $\sigma_a(x)$ and emitted radiance $L_e(x_t,\omega)$ between x and x_t where d denotes the maximum path length in ω. In the sci-vis context, L_e is determined by the colormap of the transfer function that maps a scalar number to an RGB value, and $\sigma_a(x)$ is influenced by the user-defined alpha component of the transfer function for the same scalar value.

The transmittance for homogeneous volumes follows the Beer-Lambert law where σ_t is constant:

$$T(t) = \exp(-\sigma_t t). \tag{2}$$

To simulate photon-particle collisions, we calculate a photon's *free-flight distance* until it hits a particle, denoted as t'. This calculation involves computing the probability density function (PDF) of $T(t)$, denoted as $p(t)$, and importance sampling $p(t)$ using ξ as a random number:

$$p(t) = \sigma_t \exp(-\sigma_t t), \quad t' = \frac{-\ln(1-\xi)}{\sigma_t}. \tag{3}$$

To apply this formulation to heterogeneous volumes, we can imagine the heterogeneous volume homogenized by fictitious *null particles*. The ratio, and thus the probability, of encountering these null particles is dictated by the maximum density of the volume or subvolume. This coefficient, known as the *majorant*, is denoted as $\bar{\sigma}$ and can be substituted for σ_t in Equation 3.

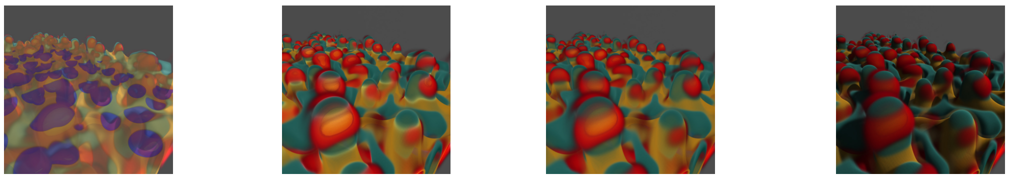
Finally, the VRE needs to be adjusted to handle null collisions and normalize coefficients, essentially turning them into probabilities. Therefore we define the probability of hitting a real particle, $P_{real}(x)$ as the ratio of density at point x to maximum density (*majorant*). Therefore, the remainder of the probabilities is the null collision, $P_{null}(x)$, probability.

$$P_{real}(x) = \frac{\sigma_a(x)}{\bar{\sigma}}, \quad P_{null}(x) = \frac{\bar{\sigma} - \sigma_a(x)}{\bar{\sigma}}. \tag{4}$$

So we end up with the final form of Equation 1:

$$L(x,\omega) = \int_{t=0}^{d} p(t)\left[P_{real}(x)L_e(x_t,\omega) + P_{null}(x)L(x_t,\omega)\right]dt. \tag{5}$$

This version of the VRE selects one of two paths when computing incoming color at x from direction ω: (i) With probability P_{real} the scalar field is sampled, assigning $L_e(x_t,\omega)$ a color via transfer function f_{cm}). (ii) With probability P_{null}, the function $L(x,\omega)$ is invoked again (recursive) from a further point x_t. The integral relies on $p(t)$, a stepping function. Since this approach follows a Monte



	Post-render compositing	Mix w/ equal weights	**Density-based (ours)**	**D.-based (ours) w/ shdws**
Depth Ordering	✗	✓	✓	✓
Phys.-B. Colors	✗	✗	✓	✓

Fig. 2: Visual comparison of various blending modes over a close-up of Miranda dataset with four channels (left-to-right): Compositing after rendering, mix function from Equation 10, our density-based blending function from Equation 6, and our blending mode with volumetric shadows. They are evaluated on their ability to achieve correct depth ordering and physically based color blending.

Carlo process, we use its importance-sampled version, the free-flight distance t', from Equation 3 to determine the stepping distance.

While the majorant $\bar{\sigma}$ is presented as a global constant, subdividing the volume into regions with local majorants $\bar{\sigma}_i$ reduces null collisions by better constraining null particles, improving performance. Instead of a global $\bar{\sigma}$, we use a grid of $\bar{\sigma}_i$s.

We use a similar notation to [27] and point the readers to [52] for further details of these derivations.

4 Method

As the VRE is defined for single channel volumes in Equation 5, we introduce our physically motivated formulation to render and blend N fields, $\theta_n \mid n \in \mathbb{N} \wedge n \leq N$, as a natural extension of Woodcock tracking in subsection 4.1. Additionally, we propose another extension to Equation 5 to formally incorporate blending functions from prior works in subsection 4.2.

We detail how to set up a Woodcock renderer that uses multiple densities in subsection 4.3. Then, in subsection 4.4 and subsection 4.5, we explore the implementation space for the method proposed in subsection 4.1.

4.1 Multi-density Woodcock Tracking

We can devise a physically based and convenient Woodcock tracking method by treating N fields as multiple densities. Doing so, we naturally extend the physically based Woodcock tracking to multiple channels.

We generalize the formulation Equation 5 to operate over multiple fields. We substitute the $P_{real}L_e(x_t,\omega)$ term with the sum of the multiplication between probabilities $P(\theta_n)$ and N radiances where θ_n is the n-th field of the volume to end up with a formulation for multi-density Woodcock tracking:

$$L(x,\omega) = \int_0^d p(t)\Big[\sum_{n=1}^{N}(P(\theta_n,x)\,L_e(x_t,\omega,\theta_n)) + P_{\text{null}}(x)\,L(x_t,\omega)\Big]dt. \tag{6}$$

The modified probabilities are as follows:

$$P(\theta_n,x) = \frac{\sigma_a(x,\theta_n)}{\bar{\sigma}_n}, \quad P_{null}(x) = 1 - \sum_{n=1}^{N}(\frac{\sigma_a(x,\theta_n)}{\bar{\sigma}_n}). \tag{7}$$

Where function σ_a yields the density for channel θ_n at point x, i.e., $\sigma_a(x,\theta_n) = \theta_n(x)$. We keep the property of probabilities adding up to 1 from Equation 4, so $\sum_{n=1}^{N} P(\theta_n) + P_{null} = 1$ for any point x.

One way to think about this formulation is that we are testing for a real collision (absorption) for N volumes. The remaining possibility is the null collision probability.

Mapping the set of N opacities to physical densities, we turn the process into a Monte Carlo process that blends the colors through accumulation in screen-space. Unlike prior blending/sampling strategies, our approach ensures physical correctness without arbitrary normalizations. Therefore, Equation 6 implicitly defines a distinct blending function.

Figure 3 illustrates our density-based blending function applied to three overlapping volumes, each mapped to a flat primary-colored transfer function.

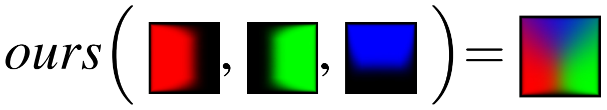


Fig. 3: Three single-channel renderings are shown from left to right: Red, green, and blue cube volumes with linearly decreasing densities on the X, -X, and -Y axes, respectively. Using our formulation from Equation 6, their multi-channel rendering is given on the right-most image.

4.2 Generalization to Blending Functions in Woodcock

Previously defined methods for blending N channels are often exploratory rather than physically based. Researchers have introduced various blending functions, such as maximum opacity selection and user-weighted mixing [5, 6], which have been effective in practice. While these blending techniques are not physically motivated, they offer valuable heuristics and have an established user base accustomed to their results. This section demonstrates how these methods can be integrated within a Woodcock tracker, leading to alternative formulations.

To allow the substitution of other blending functions, we go back to Equation 5, and we exploit the fact that these blending functions serve the purpose of reducing N colors and N opacities to one color and opacity pair. By allowing redefinition of $\sigma_a(x,\theta_n)$, and $L_e(x,\omega,\theta_n)$, we can specialize to other blending functions for a Woodcock tracker. Note that, since N fields co-exist in the same volume, the majorant, $\hat{\sigma}$, is now the sum of N maximum densities.

The maximum intensity projection (or MIP) always chooses the sample with the highest opacity at the given sample point. Therefore, the MIP can be defined as:

$$\sigma_a(x,\theta_n) = \begin{cases} \theta_n(x), & \text{if } \theta_n(x) > \theta_i(x) \mid \\ & \forall i \in \mathbb{N} \wedge i < N \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

$$L_e(x,\omega,\theta_n) = \begin{cases} f_{cm}(\theta_n(x)), & \text{if } \theta_n(x) > \theta_i(x) \mid \\ & \forall i \in \mathbb{N} \wedge i < N \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

As max deterministically picks the same sample, overlapping semitransparent regions can get filtered out by one more dense region. Despite allowing some regions to steal visibility from others, it merits intuitive usage. Figure 4 shows a visualization of max blending.

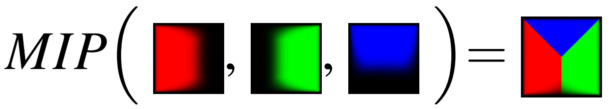


Fig. 4: Visualization of max blending function from Equation 8 using the same three fields from Figure 3.

Mixing with user-defined weights (i.e., *mix*) allows users to adjust weights for each channel to allow color mixing among them. Using weights, w_n, mix blending can be achieved by using the same $\sigma_a(x,\theta_n)$ as Equation 8 and re-defining the radiance as:

$$L_e(x,\omega,\theta_n,w_n) = \frac{\sum_{n=1}^{N} f_{cm}(\theta_n(x)) \cdot \theta_n(x) \cdot w_n}{\max(\theta_1(x),\theta_2(x)\cdots\theta_N(x))}. \tag{10}$$

Although the user interface complexity for the mix blending scales linearly with the number of channels, it can show correlations with mixed colors. Please see Figure 5 for a rendering with the mix function.

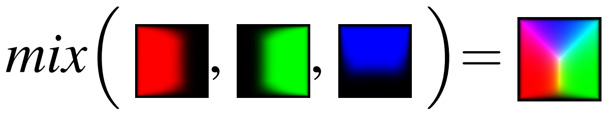


Fig. 5: Visualization of mix blending function from Equation 10 using equal weights and the same three fields from Figure 3.

Another downside of blending modes like mix and max is that they require finding a maximum among N samples as a normalization factor, which has the algorithmic complexity of $\Theta(N)$. In contrast, our density-based blending function from subsection 4.1 does not require a normalization factor (e.g., maximum opacity selection) that entails a linear cost. Instead, it probabilistically selects one sample among channels, resulting in better average algorithmic complexity.

We present definitions for some well-known functions but this approach is not constrained by them, and using the open-ended formulation we provided in this section, users can invent new blending functions. $\sigma_a(x,\theta_n)$ controls collision possibilities, and it can be parameterized to pick something other than the maximum opacity sample, or $L_e(x,\omega,\theta_n)$ can define a nonlinear combination of N channels.

There are also blending modalities completely disassociated from the rendering process that occurs in the image space, such as compositing images after rendering each channel (post-render compositing). Post-render compositing offers an occlusion-free view, but it loses the depth information. It is commonly used in tools like Fluorender.

Figure 2 illustrates some of these blending functions' benefits and visual distinctions on a real dataset.

4.3 Setup for the Woodcock Renderer

We developed a base renderer in CUDA that stores N structured grids of scalars in 3D textures, each with dimensions $k \times l \times m$. Each loaded channel utilizes their separate transfer function to determine their user-defined densities and colors. The renderer builds a coarser $k' \times l' \times m'$ *macrocell* grid. Within each macrocell, scalar ranges for each channel (a min and max pair) are stored, resulting in a memory consumption of $N \times k' \times l' \times m' \times 2$ floating-point values per channel. The construction of the macrocells can be considered a rasterization process, where the original grid of scalars is projected onto a grid with reduced resolution.

Every time a transfer function changes, we calculate that channel's majorants for each macrocell. The process involves determining the maximum opacity associated with the scalar value range of each macrocell for the given channel. Therefore, for each macrocell, we iterate between the min and the max scalar values, fetching the opacity mapped to them from the respective channel's transfer function. The majorant for that macrocell is the maximum opacity found after the loop. We run these calculations using a CUDA kernel in parallel. The process yields a majorant grid with the exact resolution as the macrocell grid. Therefore, we store $k' \times l' \times m' \times N$ floating-point numbers for the majorants.

Listing 1: Implementation of Woodcock stepping function

```
float woodcockStep(float majorant){
  return -(log(1.0f - randFloat()) / majorant);}
```

Using a 3D Digital Differential Analyzer (DDA) traversal, we employ ray tracing through the majorant grids. Woodcock steps are taken as shown in Equation 3 (calculated with Listing 1) within each macrocell based on the active majorants. Upon identifying a collision at point p, we sample the 3D textures to retrieve the relevant channels' scalar value at p. The scalar is given to the respective channel's transfer function (TF) to get the color and opacity. We interpret the linear RGB returned from TF as an emissive color. We employ rejection sampling using the null collision probability in Equation 4 to see if the sample is accepted (see Listing 2). Upon acceptance, we halt the traversal and return the color value (discarding the "w" component). We utilize an accumulation buffer to allow convergence over multiple frames.

Listing 2: Implementation of rejection scheme via null collisions

```
bool nullCollision(float majorant, float density){
  return (density < randFloat() * majorant);}
```

We identify several meaningful approaches to achieve this. They differ in implementation; however, they ultimately converge to the same image when using the same parameters.

4.4 Method #1: *N*-DDA Traversals over Majorant Grids

One obvious way to approximate Equation 6 is by simply applying Woodcock tracking for each channel as separate volumes and choosing the closest sample. This approach, although not the most computationally efficient, allows for serializing the rendering of individual channels by loading volumes and their corresponding transfer functions one at a time, thereby mitigating memory constraints. It can be utilized for an out-of-core implementation. Moreover, this provides a good baseline for comparison and can be viewed as an initial step in understanding the multi-channel Woodcock tracking.

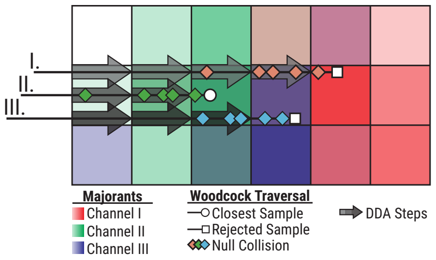


Fig. 6: The Woodcock tracking process with three DDA traversals across three majorant grids from a perspective of one ray: Starting from channel I, a DDA traversal is initiated, and between each DDA step, Woodcock steps are taken where null collisions are discarded. These processes are repeated until a sample for each channel is found. The closest sample among the three channels (channel II) is accepted, and the others (I and III) are rejected.

For this approach, we build majorant grids for each channel where the maximum density for a group of cells contained within a macrocell is stored. The majorants are recalculated every time a TF is edited.

The rendering process, illustrated in Figure 6, involves applying Woodcock tracking to all N channels using their respective majorant grids. Specifically, for a given channel C_0, we perform a DDA traversal over the majorant grid M_0. Within each cell of M_0, Woodcock steps are taken using the majorant $m_{0,i}$, where i corresponds to the 1D index of the current cell of M_0, for the ray position. The DDA traversal and steps within are repeated until we encounter a real collision. Upon hitting a particle, we record the distance (t_0) and color for C_0. This entire procedure is then repeated for $C_1, C_2, \ldots, C_N$, utilizing their respective majorant grids $M_1, M_2, \ldots, M_N$. Once we complete iterations through all channels, we select the closest sample — $\min(t_0, t_1, \ldots, t_n, \ldots, t_N)$. One obvious optimization we apply to this approach is early termination of the ray traversal for the channel C_{n+1} if t_{n+1} surpasses the closest hit distance, t_{hit}, for channels between C_0-C_n, as it would have been discarded in the final stage anyway. C-style pseudo code for this method is shown in Listing 3.

Listing 3: Multi-channel Woodcock tracking using N DDA traversals

```
void nDDAMultiChannelWoodcock(
                Ray ray, HitRecord& hit, int N){
  hit.t = FLT_MAX;
  for(int n = 0; n < N; n++){//do N DDA traversals
    DDA dda(ray);
    int cellID = dda.curCell();
    do{//DDA traversal
      float t = dda.cellEntryT();
      float maj = majorants[cellID * N + n];
      while(true){//Woodcock steps in the cell
        t += woodcockStep(maj);
        if(t > hit.t){//a closer sample exists?
          dda.stop();
          break;//No need to sample
        }
        if(!dda.InCurCell(t))//bounds check
          break;//go to the next cell
        float scalar = volumeAt(ray.org
                           + ray.dir * t, n);
        //scalar's color (r,g,b) and density (a)
        float4 sample = trFunc(scalar, n);
        //null collision check n-th channel
        if(!nullCollision(maj, sample.a)){
          //Return the color, discard the "a"
          hit.color = float3(sample);
          hit.t= t;
          dda.stop(); break;//stop this traversal
        }
      }cellID = dda.nextCell();
    }while(!dda.shouldStop()); } }
```

4.5 Method #2: One DDA Traversal over Majorant Grids

The next logical step on top of the method presented in subsection 4.4 is using a single synchronized DDA traversal over N channels to prune to-be-discarded samples earlier.

The data structures employed remain consistent with those in subsection 4.4, featuring majorant grids for each channel, which update upon modification of the transfer function.

The rendering commences with a singular DDA traversal over the majorant grid. Multiple Woodcock steps are taken within each majorant cell for each channel until either a sample is accepted or all of the rays leave the current majorant grid cell. Similarly to the method in Listing 3, we pick the closest sample to the ray origin if there are multiple hits within a majorant cell. In other words, the outer for-loop iterating over N channels in Listing 3 is placed inside the DDA traversal.

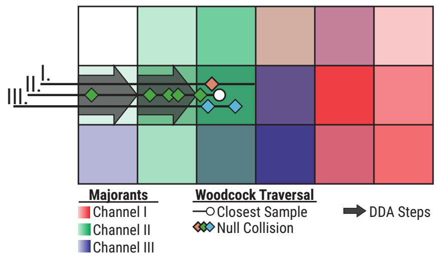


Fig. 7: The Woodcock tracking process with a single DDA traversal across three majorant grids from a perspective of one ray: A DDA traversal is initiated, and between each DDA step, three Woodcock traversals are initiated in the order of channels I, II, and III. The processes are repeated until a sample is taken (by channel II in this case). We let a final round of Woodcock traversal run for channel III as it could find a closer sample. Only the closest sample from channel II is accepted.

By synchronizing the N Woodcock tracking steps into one DDA traversal, we potentially reach the closest sample sooner and with fewer iterations (compared to subsection 4.4). Also refer to Figure 7 for an illustration of this method and Listing 4 for a streamlined kernel code.

Listing 4: Multi-channel Woodcock tracking using a synchronized DDA

```
void syncDDAMultiChannelWoodcock(
                Ray ray, HitRecord& hit, int N){
  DDA dda(ray);
  int cellID = dda.curCell();
  hit.t = FLT_MAX;
  do{//do a DDA traversal
    for (int n = 0; n < N; n++){//for channels...
      float maj = majorants[cellID * N + n];
      float t = dda.cellEntryT();
      while(true){//Woodcock steps in the cell
        t += woodcockStep(maj);
        if(t > hit.t || //a closer sample exists?
            !dda.InCurCell(t))//bounds check
          break;//go to the next channel/cell
        float scalar = volumeAt(ray.org
                            + ray.dir * t, n);
        //scalar's color (r,g,b) and density (a)
        float4 sample = trFunc(scalar, n);
        //null collision check for n-th channel
        if(!nullCollision(majs[n], sample.a)){
          //return the color, discard "a"
          hit.color = float3(sample);
          hit.t= t;
          //regardless of there is a closer
          //sample or not we need to stop DDA
          dda.stop();
          break;//go to the next channel/stop
        }
      }
    }// no channel was selected: null collision
    cellID = dda.nextCell();
  }while(!dda.shouldStop()); }
```

4.6 Implementation of Additional Effects

We can incorporate effects such as shadows and surface shading from proxy geometry into our method at a relatively low cost per ray. These effects are especially valuable because they reveal depth cues and structural details that may not be obvious from emission–absorption alone. Extending some of these effects to multiple channels is straightforward.

For volumetric shadows, the process is simple: once a real collision is found, we cast a shadow ray toward the light source. This ray is tracked through the multi-density volume until it either collides or exits. If the shadow ray collides with any channel, the original point is treated as fully shadowed, receiving only ambient light (refer to Figure 9).

For shading, we adopt a basic Blinn–Phong model [53] with normals computed via central difference gradients [10]. This is common in many single-channel visualization software [11, 54]. In multi-density Woodcock tracking, after a real collision in channel n, we sample the same channel at six neighboring positions to compute the gradient, which then serves as the normal for shading (see Figure 10).

Human perception can differentiate and identify materials [55] visually. Since our application focuses on multiple fields, we can also assign different material properties, such as diffusivity, specularity, or shininess, to each channel alongside the transfer functions. This per-channel material shading system can allow better separation of features that belong to different channels. The Figure 11 shows our per-channel material shading system over a view of the Miranda dataset.

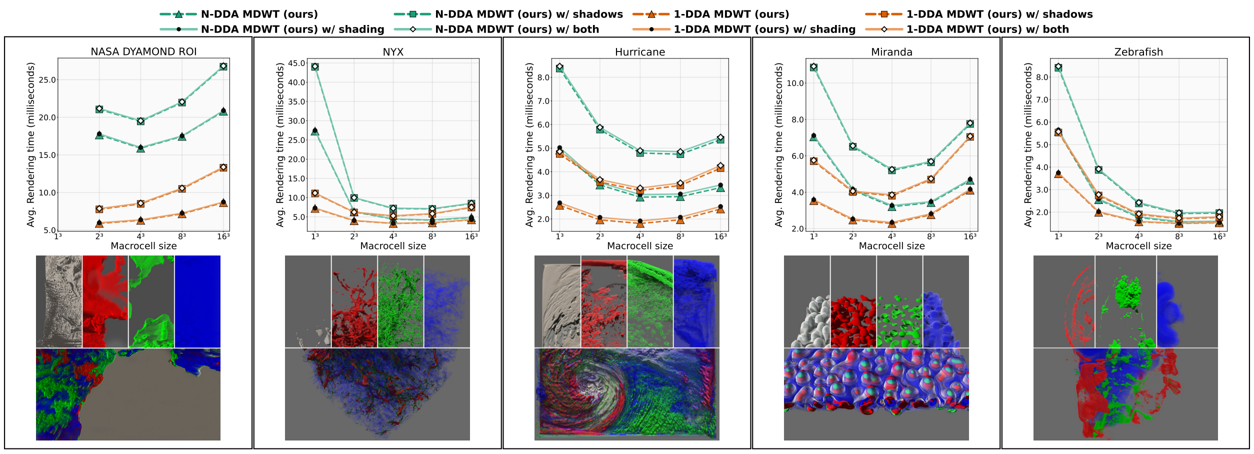


Fig. 8: The plots of average rendering times in seconds (lower is better) against increasing macrocell sizes (of cells within a macrocell) for five datasets. We compare our two methods: "N-DDA MDWT (Multi-density Woodcock Tracking)" from subsection 4.4, and "1-DDA MDWT" from subsection 4.5. Each method is evaluated under four rendering configurations: Emission+Absorption (triangles), central difference gradient shading (circles), volumetric shadowing (squares), and both effects combined (diamonds). The split images in the second row show individual channels (with shadows) on the top halves and multi-channel visualization (with both effects) in the bottom halves.

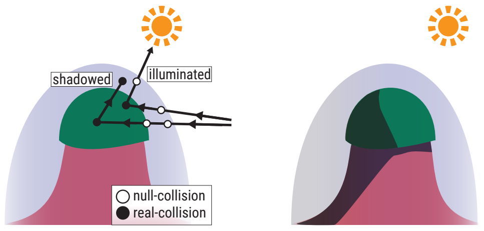


Fig. 9: Illustration of volumetric shadows in multi-density Woodcock tracking: one ray reaches the light source (illuminated), while the other is blocked by a different channel (shadowed).

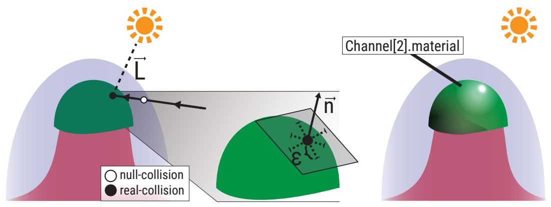


Fig. 10: Illustration of the per-channel material shading process: after a ray–volume collision, the light vector $\vec{L}$ is computed, a surface normal $\vec{n}$ is approximated via central differences using six (ε) offset samples from the hit channel, and Blinn-Phong shading is calculated from the channel's material properties with $\vec{L}$ and $\vec{n}$.

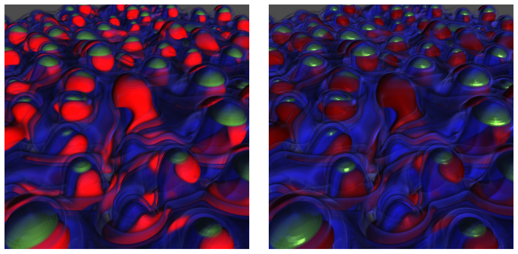


Fig. 11: Side-by-side comparison for our per-channel material shading: left, all channels are shaded with the same material properties; right, each channel utilizes distinct material properties.

5 Results and Evaluation

In this section, we evaluate the impacts of macrocell resolution (subsection 5.1), the number of channels (subsection 5.2), the trade-off between quality and performance (subsection 5.3), and the choice of blending functions, including our proposed method and prior alternatives (subsection 5.4). We also present feedback from domain experts on our method (subsection 5.5) from a series of semi-structured interviews.

We utilize a variety of datasets to drive more extensive conclusions. *NASA DYAMOND ROI* (region of interest) is the Strait of Gibraltar from the NASA DYAMOND ocean data that is in $1400 \times 1600 \times 350$, *NYX* is a cosmological simulation that is in $512 \times 512 \times 512$ resolution, *Hurricane* is a weather simulation that is flatter in the z dimension with $500 \times 500 \times 100$ resolution, *Miranda* is a hydrodynamics simulation with highly overlapping features in $256 \times 384 \times 384$ resolution and, *Zebrafish* is a well-known multi-channel microscopy dataset in $640 \times 640 \times 121$ resolution [56].

We run our experiments on an NVIDIA RTX4090 GPU with the renderer we implemented in CUDA. We take the average timing of 500 frames for each data point after rendering 50 warm-up frames, which are excluded from the average.

5.1 Impact of Macrocell Resolution

The macrocell and majorant grids' resolution directly impacts our method's performance, as these grids approximate density at query points. This approximation influences the Woodcock step size and collision probabilities. A finer grid improves stepping and reduces null collisions but requires more memory, whereas a coarser grid increases null collisions, and thus the time per ray [57]. In this section, we evaluate the performance of our two multi-density Woodcock tracking methods from section 4 across different macrocell sizes.

We measure the rendering times of each algorithm using the emission+absorption (E+A) model, E+A with gradient shading [10], E+A with shadows, and E+A with both effects.

Four channels of NASA DYAMOND ROI, NYX, Hurricane, and Miranda, and three channels of Zebrafish are used for this experiment. Figure 8 shows the render times vs. decreasing macrocell resolution. For each data point, we start with the finest possible macrocell resolution, and subsequently, we insert twice as many cells per dimension within a macrocell, i.e., $8\times$ less resolution. We record 170, 291, 504, 453, and 686 frames per second (FPS) for NASA DYAMOND, NYX, Hurricane, Miranda, and Zebrafish datasets. Our fastest timings with volumetric shadows are 118, 178, 286, 299, and 577 FPS, respectively. Lines with gradient shading closely follow their unshaded counterparts, as the effect involves only six low-cost texture lookups at the collision point and results in an average increase of 0.9% in render time. We also report the memory consumption of our application using the most performant data macrocell resolution in Table 1.

We observe that method #2 from subsection 4.5 with 1-DDA traversal achieves the highest frame rates. For both methods, the performance peaks around 64 cells per macrocell ($4 \times 4 \times 4$) for almost all datasets. We record the memory consumption of our data structures to be between 0.58-4.69% of the original data sizes at the peak performance.

We also tested a third method that uses a single majorant buffer, summing all majorants into one value and testing each sample against the cumulative majorant. The idea was to reduce the number of steps, but the results were negative. This method achieved only 59.88-84.03% of the performance of our fastest method and required about $8\times$ the macrocell size, consuming more memory. Consequently, we omit this method and its results. Notably, this approach converges to a technique described in the overlapping volumes chapter of Fong et al. Production Volume Rendering course [27].

5.2 Performance Impact of Multiple Channels

The number of channels is one of the factors that can significantly increase the cost of algorithms such as ray marchers. Therefore, we document the performance of our methods against the increasing number of channels. We use a standard ray marcher as a baseline in these experiments. To closely mirror the behavior of out-of-the-box visualization solutions, we use the common mix blending function with equal weights for the ray marcher.

Figure 12 depicts the performance of our two methods and the aforementioned ray marcher against the increasing number of channels. The ray marcher and our methods are configured to produce similar-quality images using the same transfer function and scene configurations.

We artificially amplify the channel counts of four of our datasets by duplicating the same channel data with different transfer functions, allowing for testing at higher scalability. Unlike the others, the Hurricane dataset contains 12 unique channels, so we use it as is. However, with NASA DYAMOND ROI, we only use six channels due to memory constraints.

Our most efficient method (subsection 4.5) can render 12 channels simultaneously, equivalent to rendering ap-

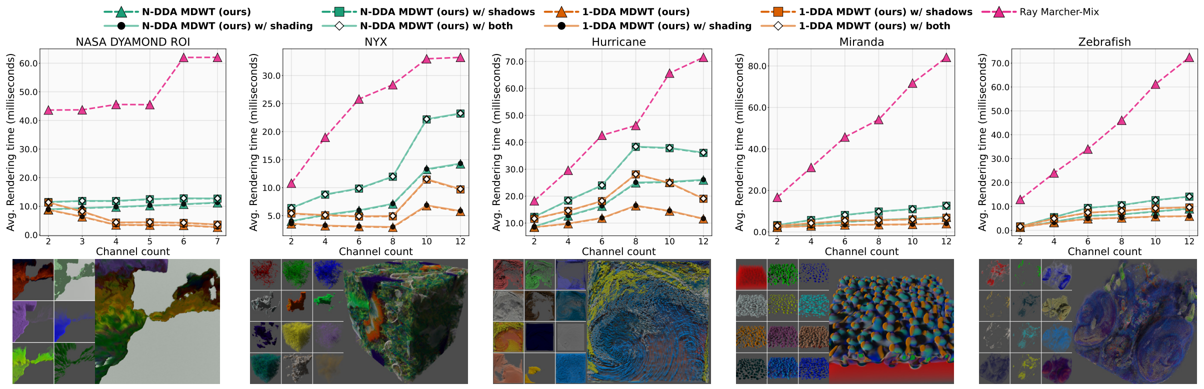


Fig. 12: The average rendering times in milliseconds (lower is better) versus the increasing number of channels across five datasets. For each dataset, the top row shows a line plot comparing "N-DDA MDWT (Multi-density Woodcock Tracking)" from Figure 6, our "1-DDA MDWT" method from subsection 4.5, and "Ray Marcher-Mix," a ray marcher using mix blending. Each of our methods is evaluated under four rendering configurations: Emission+Absorption (triangles), central difference gradient shading (circles), volumetric shadowing (squares), and both effects combined (diamonds). The bottom row presents the corresponding visualizations: individual channels on the left and all channels combined with multi-channel Woodcock tracking on the right.

proximately $\approx 4-8$ channels sequentially with volumetric shadows. Adding more channels creates occlusion, hiding previously visible layers; our multi-density Woodcock tracking algorithm exploits this fact and terminates on the first hit, yielding amortized timings. In contrast, ray marching collects partial opacities along the ray direction, leading to a linear traversal cost. The mix operator must also blend N colors, introducing another linear cost, causing ray marching to scale linearly with the number of channels.

We observe some rapid changes in the performance trend, especially noticeable in NYX between 8 and 10 channels (see Figure 12). This spike is upward, indicating a loss of performance. Although channels 8-10 are more occlusive, they do not improve performance because each volume occupies a large portion of the space without overlapping significantly. This non-overlapping space consumption causes the majorant grids to become denser and less adaptive, which in turn causes our algorithm to halt more often to check for collisions.

The NASA DYAMOND ROI exhibits nearly constant scaling with our methods, despite the increasing number of channels. This can be explained by its shallow shape, where data is large on X and Y but has comparatively less depth in the Z axis; therefore, amortization from occlusion happens even faster in this test case, as features are more condensed. We also observe Ray Marcher benefiting from early ray termination in this dataset. Nevertheless, the introduction of a more transparently adjusted channel immediately creates $\approx 17\%$ performance penalty as seen in the 6th data point.

Our N-DDA multi-density Woodcock tracking (method #1 from subsection 4.4) does not obtain as much amortization from occlusion as the 1-DDA multi-density Woodcock tracking (method #2 from subsection 4.5) does, despite using the same data structure. From our observations, this outcome can be explained by the increasing number of redundant traversals since this method does not prune some of the redundant traversals that are pruned in method #2 as explained in subsection 4.5 and seen in Figure 7.

Table 1: Memory consumption of our implementation. The columns report the consumption for the specified number of channels. "Base" shows the memory allocated for the dataset; for the macrocells and the majorant grid, the consumption for the best-performing resolution is reported.

Dataset\Memory (MB)	Base	Macrocells	Majorants	Total
NASA ROI (4 channels)	2990.00	375.98	187.99	3553.98
NYX (4 channels)	2048.00	64.00	32.00	2144.00
Hurricane (4 channels)	381.47	11.92	5.96	400.35
Miranda (4 channels)	576.00	18.00	9.00	603.00
Zebrafish (3 channels)	567.15	2.19	1.01	570.35

5.3 Quality vs. Performance

Woodcock tracking enables one sample per ray, allowing frames to accumulate and gradually reduce variance. In contrast, ray marching employs an analytic solution that takes multiple partial samples per ray, resulting in images with virtually no variance, albeit at a significantly higher cost (as shown in Figure 12).

This is indeed a trade-off between per-frame performance and noise. To evaluate the cost-efficiency of our method, we conduct "similar performance" and "similar quality" experiments using the NYX and Miranda datasets, as seen in Figure 13. First, we compare the ray marcher's performance to our 1-DDA MDWT in similar quality setups. We find 16 samples per pixel(spp) to yield visuals of similar quality to the ray marcher sampling at the Nyquist rate (to avoid aliasing artifacts). Next, we fix our method's parameters and compare its performance to the ray marcher by linearly increasing the sampling interval from the artifact-free Nyquist rate to the point where both methods perform similarly.

Our results show that the ray marcher can achieve high-quality images in less time, particularly with the NYX dataset, which reaches a similar quality image $\approx 3\times$ faster. However, similar quality is achieved at around the same frame rate in datasets with more empty space, like Miranda. In the "similar performance" benchmarks, ray marcher images rendered with sampling intervals larger than the Nyquist rate exhibit significant aliasing artifacts, obscuring features. This explains the drastic variations seen in RM-Mix images in Figure 13, where undersampling leads to severe drops in quality. Unlike our Monte Carlo-based solution, ray marching cannot improve image quality incrementally. Additionally, increasing spp in our method incurs a linear cost while reducing the ray marcher's sampling interval scales nearly exponentially. Our approach maintains higher interactivity without requiring the highest spp from the start.

5.4 Visual Comparisons for Blending Functions

We compare the visual results of various inter-channel blending operators over three datasets in Table 2, using ꟻLIP and Peak Signal to Noise Ratio (PSNR). We use four flat colormaps: red, green, blue, and white. In these experiments, we compare an image-space method, two not physically based blending functions, to our physically based density blending function. These are post-render compositing and mixing with equal weights from Fluorender[9] and maximum intensity projection (MIP) from Rice and Schulze[5].

Our results indicate the post-render compositing offers the least physically correct result and the farthest image in terms of difference. Although this mode is intended to offer another way to examine the data, it makes denser datasets such as NYX, and Hurricane harder to distinguish as both the colors and depth relationships are altered.

The mixing function has the unique property of blending colors into a mixture of two colors. However, this can be counterintuitive to untrained users, as these colors may be outside the current set of colormaps, or they may be confused with other fields if the color is present in a colormap. This effect is more pronounced in the Hurricane dataset, where reds and blues turn into magenta (not a color in the colormap set).

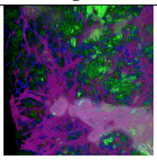
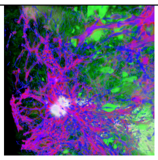
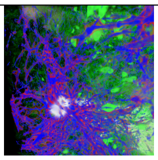
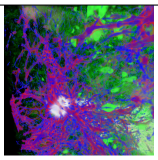
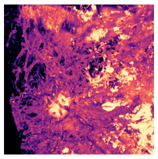
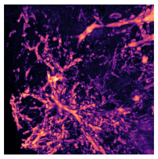
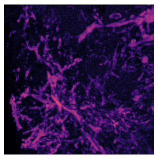
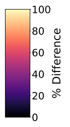
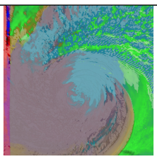
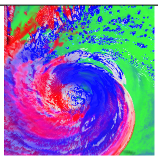
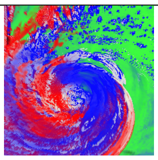
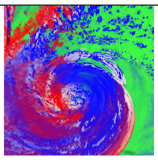
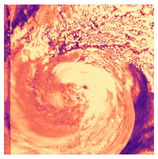
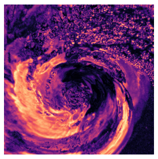
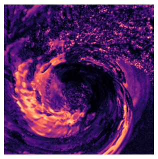
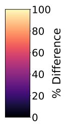
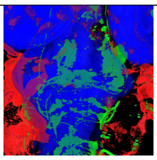
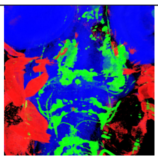
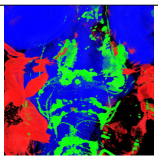
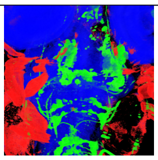
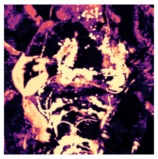
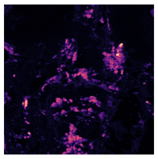
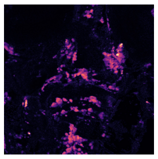
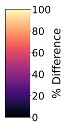
Upon reviewing the experiments in subsection 5.2 and subsection 5.4, we notice that visual clutter becomes more pronounced as the number of channels increases. Beyond five channels, the blending colors may start appearing counterintuitive. Additionally, employing blending functions that generate out-of-colormap colors like mix can exacerbate this problem, potentially leading to confusion between a field and overlap of the other two fields.

The results closest to our blending function are from MIP, as it always selects the same maximum opacity sample as ours. However, this approach can lead to the loss of semitransparent regions due to the sharp cut-off of `max()`.

When tested on the Zebrafish dataset, the mix and max functions show minimal differences from our approach. In contrast to simulation data, where features cluster in specific regions, the microscopy data exhibit less overlap between channels. This reduced overlap leads to less color blending, resulting in similar outcomes across blending modes.

The density-based blending function offers an intuitive way to blend colors, as it can blend colors through more neutral tones while preserving the information of more transparent regions, avoiding sharp cut-offs. It remains physically based as it stems from the direct extension of Woodcock tracking sampling to multiple channels.

Table 2: Visual comparisons of inter-channel blending functions over test datasets: Direct volume renderings using Emission+Absorption model with identical parameters for various blending functions, accompanied by heat map images depicting the difference to our physically motivated density-based blending function using ꟻLIP metric[58]. The overall mean difference and PSNR to density-based blending are also reported under the heat maps.

Dataset \ Blending		Post-Render Composite	Equal Mix	MIP	**Density-Based (ours)**
NYX	Direct Volume Rendering				
	Difference to ours				
	Mean ꟻLIP Diff., PSNR	47%, 16.56	21%, 23.03	13%, 22.85	
Hurricane	Direct Volume Rendering				
	Difference to ours				
	Mean ꟻLIP Diff., PSNR	77%, 11.61	33%, 19.82	24%, 23.02	
Zebrafish	Direct Volume Rendering				
	Difference to ours				
	Mean ꟻLIP Diff., PSNR	34%, 16.89	6%, 34.62	6%, 34.06	

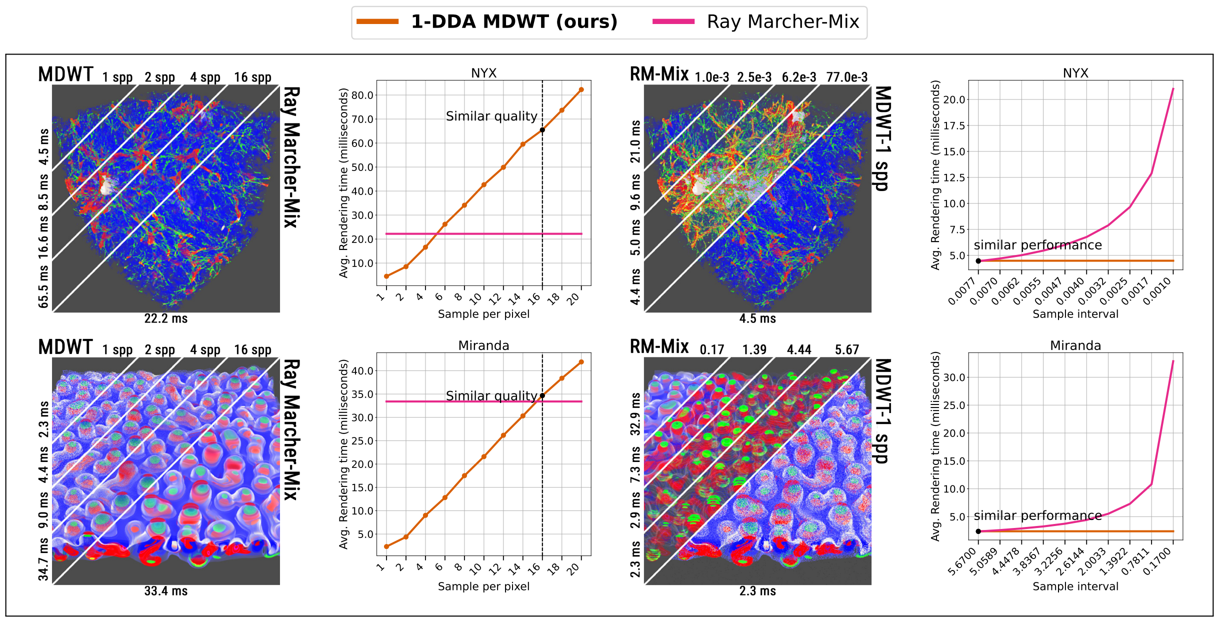


Fig. 13: Similar quality (left) and performance (right) benchmarks for two datasets. The similar quality images display slices rendered with our Multi-Density Woodcock Tracking (MDWT) at increasing samples per pixel (spp), while the bottom-right half shows the ray marcher using mix blending with half of the minimum cell size as the sampling interval (Nyquist rate). The similar performance images show slices rendered with the ray marcher and mix blending (RM-Mix) for increasing sampling intervals, with the bottom-right half showing the same scene rendered by MDWT at 1 spp. The rendering time for each slice is indicated on the left side of each image, and each image is accompanied by a line plot of the average rendering times in milliseconds (lower is better).

5.5 Domain-Expert Evaluation

To evaluate the practical relevance of our approach, we conducted semi-structured interviews [59] with four domain experts, each from a different scientific field. All participants provided informed consent before the interviews. One expert specialized in chemistry with a focus on quantum optics, another in pharmaceutical sciences with a focus on bio-molecular interactions, a third in physics with a focus on fluid dynamics, and the last in computer science with a focus on data compression. During the interviews, we used datasets familiar to the experts, along with their standard visualization tools — either VisIt [11] or Napari [50]. These tools were first set up in meaningful visualizations, after which we recreated the same scenes in our renderer for side-by-side comparison. This setup allowed us to assess how conventional solutions handle multi-channel data and to discuss the potential benefits and limitations of our method.

The datasets shown to the chemistry and pharmaceutical experts are presented in Figure 14 and Figure 15, respectively. For the physics and computer science experts, we used the Miranda dataset from our earlier evaluations. In each session, we compared results from the experts' existing workflows with those produced by our renderer. We then solicited feedback on usability and clarity. We asked participants to reflect on whether they would adopt our method if it were properly deployed into their workflows.

Across the interviews, several themes emerged, highlighting key advantages of our method: (i) accurate rendering of both color and depth (a limitation in Napari), (ii) clear visualization of feature interactions across channels, (iii) correct channel ordering without occlusion artifacts (a limitation in VisIt), and (iv) improved 3D perception.

The chemistry and pharmaceutical experts shared similar workflows, relying on Napari to visualize microscopy data. Both noted that Napari's use of MIP and additive rendering compromises depth and color accuracy, resulting in“flat” results that also cannot render shadows and other depth cues. In contrast, our renderer provided shading and depth cues that clarified interactions

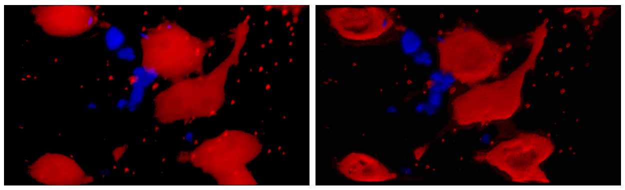


Fig. 14: E. coli perfusion visualizations as used in our expert study: left is visualized with Napari, and right is rendered with our MDWT.

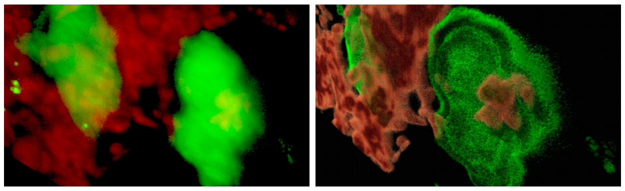


Fig. 15: Visualization of interacting Macrophage and E. coli cells as used in our expert study: left is visualized with Napari, and right is rendered with our MDWT.

between structures. The pharmaceutical expert even identified elements initially assumed to be errors, which later proved to be genuine shapes formed during data acquisition—details that were difficult to discern with Napari. While both experts acknowledged that transfer functions remain challenging to use (a challenge to be addressed by the visualization community [42, 39]), they found that our method reduced the need for extensive parameter tuning to reveal feature boundaries. Finally, both emphasized the importance of examining multiple timesteps in their workflows and suggested extending our method to time-varying data.

The physics and computer science experts, both VisIt users, highlighted additional contrasts between their preferred tool and our method. They agreed that inconsistent multi-channel ordering is a known limitation of VisIt, which does obscure important relationships between channels (and sometimes the entire channel). In contrast, our method accurately overlays channels regardless of the loading order. The physics expert stated that our method's ability to reveal inter-channel correlations could be helpful in their analysis. Both experts utilized this advantage in the example of the Miranda dataset, which allowed them to observe that regions of high velocity along the z-axis overlapped with areas of higher density—revealing the fluid mixing process captured in the data. The physics expert noted that they would utilize this feature if it were integrated into a tool like VisIt, emphasizing its potential practical value.

6 Discussion and Conclusion

In this paper, we presented an efficient method for rendering multiple volume channels by extending the Woodcock tracking algorithm to handle multiple densities. We explored two approaches: an intuitive method (subsection 4.4) using *N* serial traversals to find the closest sample and a more performant method (subsection 4.5) that combines *N* traversals into one for fewer steps. Additionally, we introduced a Monte Carlo estimator for a physically motivated inter-channel blending function. We generalized the framework to support other blending functions, including user-weighted blending and maximum intensity projection.

Our most efficient method achieved real-time frame rates for multi-channel rendering by leveraging occlusion to terminate traversal upon sampling. Unlike ray marching, which struggles with interactivity when casting shadow rays, our approach samples and casts shadow rays for a single sample without performance penalties. While initial frames exhibited variance, the results quickly converged to high-quality visuals within 20-30 ms. Despite producing initial noise-free images, taking fewer samples to increase the performance is not an option for the ray marcher, which results in aliasing artifacts.

We proposed a density-based blending approach that avoided the pitfalls of opacity and compositing methods, producing distinct results without out-of-transfer-function colors. It required no complex interfaces, handled occlusion with many opaque channels, and unified prior blending functions, allowing users to choose blending based on their needs, such as weighted mixing for correlation discovery or MIP for emphasizing dominant features.

A vital discussion is the impracticality of residual ratio tracking methods [36, 12] in the sci-vis context due to how color information is derived. Unlike cinematic rendering, where RGB-classified data can be treated as separate volumes with dedicated majorants, sci-vis applies a transfer function to a single volume. Storing per-color-channel majorants is inefficient using uniform grids— which often outperform hierarchical structures in traversal and update performance [35] — as this would incur significant memory overhead if replicated for each transfer function channel. Moreover, integrating minorant-based optimizations is unsuitable as the analytically solvable control component of extinction (alpha) and emission spectra (the RGB) can differ for the same transfer function.

This work suggests several potential avenues for future research. In subsection 4.2, we introduced a framework for defining new blending functions, which could be expanded through another user study and investigating novel functions. Attribute-aware radial basis functions [33] offer a promising multi-channel blending strategy. The potential of the *N*-DDA method (subsection 4.4) to reduce memory contention in out-of-core use cases could also be explored. Our expert study suggests that incorporating our method into existing visualization solutions that come with complementary features, such as 2D transfer function editors and temporal navigation features, would be a beneficial future step. One could also investigate intuitive user interfaces for multi-channel rendering (e.g., multi-field transfer function design). Additionally, extending the approach to unstructured meshes using techniques like those from Morrical et al. [34], as well as research into ensemble and uncertainty visualization, would be valuable.

Ultimately, our multi-density Woodcock tracking method provides a performant and versatile solution for multi-channel volume visualization. Leveraging contemporary rendering techniques, our approach produces high-fidelity images in real-time while offering greater interactivity than previous multi-channel sci-vis methods. Its physically driven blending function preserves colormaps and depth ordering without complex interfaces, enabling intuitive, user-focused visualization. These advantages translate into perceptual benefits, as demonstrated in applications such as fluid dynamics and cellular microscopy.

Acknowledgments

This work was supported by NSF[1] (Awards OAC 2138811, OISE 2330582), ARPA-H[2] (Grant D24AC00338-00), the Intel oneAPI Center of Excellence at the University of Utah, NASA AMES (cooperative agreement 80NSSC23M0013), JPL (Subcontract 1685389), and the DFG[3] (Grant 456842964). We thank Lukic et al., Wang et al., Peter Lindstrom, and Yong Wan for providing the NYX, Hurricane, Miranda, and Zebrafish datasets, respectively, and NVIDIA Corporation for providing the hardware. We also thank Andy Cook, Audrey Eshun, Kyo Lee, Peter Lindstrom, and Yuliang Zhang for donating their time and expertise.

The source code for this work is publicly available at `https://github.com/alpers-git/MDWTracker`

[1]National Science Foundation

[2]Advanced Research Projects Agency for Health

[3]Deutsche Forschungsgemeinschaft (German Research Foundation)

References

[1] K. Engel, M. Kraus, and T. Ertl, “High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading,” in *Proceedings of the ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, ser. HWWS ’01, 2001.

[2] H. K. Tuy and L. T. Tuy, “Direct 2-d display of 3-d objects,” *IEEE Computer Graphics and Applications*, vol. 4, no. 10, pp. 29–34, 1984.

[3] M. Ikits, J. Kniss, A. Lefohn, and C. Hansen, “Volume Rendering Techniques,” in *GPU Gems*, 2004, available at https://developer.nvidia.com/sites/all/modules/custom/gpugems/books/GPUGems/gpugems_ch39.html, Accessed: 24 June 2022.

[4] E. Woodcock, T. Murphy, H. P., and L. T.C., “Techniques used in the GEM Code for Monte Carlo Neutronics Calculation in Reactors and Other Systems of Complex Geometry,” Argonne National Laboratory, Tech. Rep., 1965.

[5] A. Rice and J. P. Schulze, “Real-Time Volume Rendering of Four Channel Data Sets,” in *Visualization Conference, IEEE*, 2004.

[6] C. K. Liang, “Bridging the Resolution Gap: Superimposition of Multiple Multi-Channel Volumes,” Master’s thesis, University of California, San Diego, 2008.

[7] H. S. Kim, “Visual Exploration in Volume Rendering for Multi-Channel Data,” Ph.D. dissertation, University of California, San Diego, 2011.

[8] W. Cai and G. Sakas, "Data Intermixing and Multi-volume Rendering," *Computer Graphics Forum*, vol. 18, no. 3, 1999.

[9] Y. Wan, H. Otsuna, H. A. Holman, B. Bagley, M. Ito, A. K. Lewis, M. Colasanto, G. Kardon, K. Ito, and C. Hansen, "Fluorender: Joint Freehand Segmentation and Visualization for Many-Channel Fluorescence Data Analysis," *BMC Bioinformatics*, vol. 18, no. 1, 2017.

[10] M. Ikits, J. Kniss, A. Lefohn, and C. Hansen, *Chapter 39. Volume Rendering Techniques*. Addison-Wesley, 2004, p. 352–365.

[11] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G. H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E. W. Bethel, D. Camp, O. Rübel, M. Durant, J. M. Favre, and P. Navrátil, "VisIt: An End-User Tool For Visualizing and Analyzing Very Large Data," 2012.

[12] P. Kutz, R. Habel, Y. K. Li, and J. Novák, "Spectral and decomposition tracking for rendering heterogeneous volumes," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2017)*, vol. 36, no. 4, 2017.

[13] A. Sahistan, S. Zellmann, N. Morrical, V. Pascucci, and I. Wald, "Multi-Density Woodcock Tracking: Efficient High-Quality Rendering for Multi-Channel Volumes," in *Eurographics Symposium on Parallel Graphics and Visualization*, 2025.

[14] B. Cabral, N. Cam, and J. Foran, "Accelerated Volume Rendering and Tomographic Reconstruction Using Texture Mapping Hardware," in *Proceedings of the 1994 Symposium on Volume Visualization*, ser. VVS '94, 1994.

[15] T. Cullip and U. Neumann, "Accelerating Volume Reconstruction with 3D Texture Hardware," University of North Carolina at Chapel Hill, Tech. Rep., 1993.

[16] T.-H. Lee, Y. J. Kim, and J. Chang, "High Quality Volume Rendering for Large Medical Datasets Using GPUs," in *Systems Modeling and Simulation: Theory and Applications*, 2005.

[17] K. Perlin and E. M. Hoffert, "Hypertexture," *SIGGRAPH Comput. Graph.*, vol. 23, no. 3, 1989.

[18] J. Krüger and R. Westermann, "Acceleration Techniques for GPU-based Volume Rendering," in *Proceedings of IEEE Visualization*, ser. IEEE Visualization Conference, 2003.

[19] S. Röttger, S. Guthe, D. Weiskopf, T. Ertl, and W. Straßer, "Smart Hardware-Accelerated Volume Rendering," in *Proceedings of the Symposium on Data Visualisation 2003*, ser. VISSYM '03, vol. 3, 2003.

[20] M. Hadwiger, C. Sigg, H. Scharsach, K. Bühler, and M. Gross, "Real-Time Ray-Casting and Advanced Shading of Discrete Isosurfaces," *Computer Graphics Forum*, vol. 24, 2005.

[21] M. Raab, D. Seibert, and A. Keller, "Unbiased Global Illumination with Participating Media," in *Monte Carlo and Quasi-Monte Carlo Methods 2006*, 2008.

[22] A. Sahistan, S. Demirci, N. Morrical, S. Zellmann, A. Aman, I. Wald, and U. Güdükbay, "Ray-traced Shell Traversal of Tetrahedral Meshes for Direct Volume Visualization," in *Proceedings of the IEEE Visualization Conference-Short Papers*, ser. VIS '21, 2021.

[23] P. Ljung, C. Lundström, and A. Ynnerman, "Multiresolution Interblock Interpolation in Direct Volume Rendering," in *EUROVIS - Eurographics/IEEE VGTC Symposium on Visualization*, 2006.

[24] N. Morrical, W. Usher, I. Wald, and V. Pascucci, "Efficient Space Skipping and Adaptive Sampling of Unstructured Volumes Using Hardware Accelerated Ray Tracing," in *2019 IEEE Visualization Conference*, 2019.

[25] P. Moran and D. Ellsworth, "Visualization of AMR Data With Multi-Level Dual-Mesh Interpolation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, 2011.

[26] I. Wald, S. Zellmann, W. Usher, N. Morrical, U. Lang, and V. Pascucci, "Ray Tracing Structured AMR Data Using ExaBricks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, 2021.

[27] J. Fong, M. Wrenninge, C. Kulla, and R. Habel, "Production Volume Rendering: SIGGRAPH 2017 Course," in *ACM SIGGRAPH 2017 Courses*, ser. SIGGRAPH '17, 2017.

[28] Y. Yue, K. Iwasaki, B.-Y. Chen, Y. Dobashi, and T. Nishita, "Unbiased, Adaptive Stochastic Sampling for Rendering Inhomogeneous Participating Media," *ACM Transactions on Graphics*, vol. 29, no. 6, 2010.

[29] L. Szirmay-Kalos, B. Tóth, and M. Magdics, "Free Path Sampling in High Resolution Inhomogeneous Participating Media," *Computer Graphics Forum*, vol. 30, no. 1, 2011.

[30] T. Günther, A. Kuhn, and H. Theisel, "MCFTLE: Monte Carlo Rendering of Finite-Time Lyapunov Exponent Fields," *Computer Graphics Forum*, vol. 35, 2016.

[31] J. Martschinke, S. Hartnagel, B. Keinert, K. Engel, and M. Stamminger, "Adaptive Temporal Sampling for Volumetric Path Tracing of Medical Data," *Computer Graphics Forum*, 2019.

[32] N. Hofmann, J. Martschinke, K. Engel, and M. Stamminger, "Neural Denoising for Path Tracing of Medical Volumetric Data," *ACM Transactions on Graphics (Proceedings of SIGGRAPH '20)*, vol. 3, no. 2, 2020.

[33] N. Morrical, S. Zellmann, A. Sahistan, P. Shriwise, and V. Pascucci, "Attribute-Aware RBFs: Interactive Visualization of Time Series Particle Volumes Using RT Core Range Queries," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, 2023.

[34] N. Morrical, A. Sahistan, U. Güdükbay, I. Wald, and V. Pascucci, "Quick Clusters: A GPU-Parallel Partitioning for Efficient Path Tracing of Unstructured Volumetric Grids," *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, no. 01, 2023.

[35] S. Zellmann, Q. Wu, A. Sahistan, K.-L. Ma, and I. Wald, "Beyond ExaBricks: GPU Volume Path Tracing of AMR Data," *Computer Graphics Forum*, vol. 43, no. 3, 2024.

[36] J. Novák, A. Selle, and W. Jarosz, "Residual ratio tracking for estimating attenuation in participating media," *ACM Trans. Graph.*, vol. 33, no. 6, 2014.

[37] M. Galtier, S. Blanco, C. Caliot, C. Coustet, J. Dauchet, M. El Hafi, V. Eymet, R. Fournier, J. Gautrais, A. Khuong, B. Piaud, and G. Terrée, "Integral formulation of null-collision monte carlo algorithms," *Journal of Quantitative Spectroscopy and Radiative Transfer*, vol. 125, pp. 57–68, 2013.

[38] I. Georgiev, Z. Misso, T. Hachisuka, D. Nowrouzezahrai, J. Křivánek, and W. Jarosz, "Integral formulations of volumetric transmittance," *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, vol. 38, no. 6, 2019.

[39] R. Maciejewski, Y. Jang, I. Woo, H. Jänicke, K. P. Gaither, and D. S. Ebert, "Abstracting Attribute Space for Transfer Function Exploration and Design," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 1, 2013.

[40] J. Caban and P. Rheingans, "Texture-Based Transfer Functions for Direct Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, 2008.

[41] A. Kumar, X. Zhang, H. L. Xin, H. Yan, X. Huang, W. Xu, and K. Mueller, "RadVolViz: An Information Display-Inspired Transfer Function Editor for Multivariate Volume Visualization," *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[42] B. Pan, J. Lu, H. Li, W. Chen, Y. Wang, M. Zhu, C. Yu, and W. Chen, "Differentiable Design Galleries: A Differentiable Approach to Explore the Design Space of Transfer Functions," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, no. 1, 2024.

[43] H. S. Kim, J. P. Schulze, A. C. Cone, G. E. Sosinsky, and M. E. Martone, "Dimensionality Reduction on Multi-Dimensional Transfer Functions for Multi-Channel Volume Data Sets," *Information Visualization*, vol. 9, no. 3, 2010.

[44] ——, "Multichannel Transfer Function with Dimensionality Reduction," *SPIE Proceedings*, 2010.

[45] R. Khlebnikov, B. Kainz, M. Steinberger, and D. Schmalstieg, "Noise-Based Volume Rendering for the Visualization of Multivariate Volumetric Data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, 2013.

[46] L. Herzberger, M. Hadwiger, R. Krüger, P. Sorger, H. Pfister, E. Groeller, and J. Beyer, "Residency Octree: A Hybrid Approach for Scalable Web-Based Multi-Volume Rendering," *IEEE Transactions on Visualization and Computer Graphics*, vol. 30, 2023.

[47] C. Crassin, F. Neyret, S. Lefebvre, and E. Eisemann, "Gigavoxels: ray-guided streaming for efficient and detailed voxel rendering," in *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*, ser. I3D '09. Association for Computing Machinery, 2009.

[48] Y. Wan, H. Otsuna, C.-B. Chien, and C. Hansen, "An Interactive Visualization Tool for Multi-Channel Confocal Microscopy Data in Neurobiology Research," *IEEE Transactions on Visualization and Computer Graphics*, vol. 15, no. 6, 2009.

[49] ——, "FluoRender: An Application of 2D Image Space Methods for 3D and 4D Confocal Microscopy Data Visualization in Neurobiology Research," in *2012 IEEE Pacific Visualization Symposium*, 2012.

[50] G. Bokota, N. Sofroniew, T. Lambert, J. Nunez-Iglesias, M. Bussonnier, and napari contributors, "napari: a multi-dimensional image viewer for python," 2019, with contributions from 183 developers. [Online]. Available: https://doi.org/10.5281/zenodo.3555620

[51] J. T. Kajiya and B. P. Von Herzen, "Ray Tracing Volume Densities," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, 1984.

[52] M. Pharr, W. Jakob, and G. Humphreys, "11 Volume Scattering," in *Physically based rendering: From theory to implementation*, 4th ed. The MIT Press, 2023, pp. 697–736.

[53] J. F. Blinn, "Models of light reflection for computer synthesized pictures," *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, p. 192–198, Jul 1977.

[54] J. Ahrens, B. Geveci, and C. Law, "ParaView: An end-user tool for large-data visualization," in *Visualization Handbook*, C. D. Hansen and C. R. Johnson, Eds., 2005.

[55] B. Xiao and C. Liao, "Material perception connects vision, cognition and action," *Nature Reviews Psychology*, 2025.

[56] K. Zhao, S. Di, X. Lian, S. Li, D. Tao, J. Bessac, Z. Chen, and F. Cappello, "Sdrbench: Scientific data reduction benchmark for lossy compressors," in *2020 IEEE International Conference on Big Data*. IEEE, 2020. [Online]. Available: https://sdrbench.github.io

[57] Z. Misso, Y. K. Li, B. Burley, D. Teece, and W. Jarosz, "Progressive null-tracking for volumetric rendering," in *ACM SIGGRAPH Conference Papers*, 2023.

[58] P. Andersson, J. Nilsson, and T. Akenine-Möller, "Visualizing and Communicating Errors in Rendered Images," in *Ray Tracing Gems II*, A. Marrs, P. Shirley, and I. Wald, Eds., 2021, ch. 19, pp. 301–320.

[59] R. Ruslin, S. Mashuri, M. Sarib, F. Alhabsyi, and H. Syam, "Semi-structured interview: A methodological reflection on the development of a qualitative research instrument in educational studies ruslin," vol. Vol. 12, pp. 22–29, 02 2022.