



Recolorable Posterization of Volumetric Radiance Fields Using Visibility-Weighted Palette Extraction

Kenji Tojo  and Nobuyuki Umetani 

The University of Tokyo

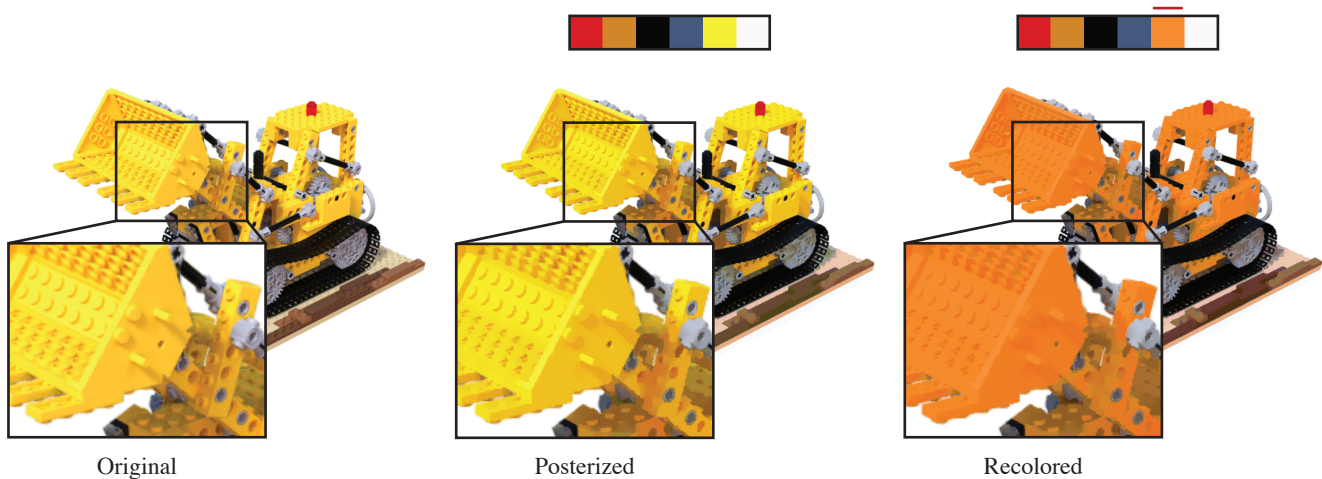


Figure 1: We stylize a volumetric radiance field (left) with a posterized look (middle). After the palette-based decomposition of the radiance field, we convert the decomposed layers into the ones that are uniformly colored with one of the palette colors. We also introduce intuitive recoloring via palette color modification into the stylized rendering of the volumetric radiance field (right).

Abstract

Volumetric radiance fields have recently gained significant attention as promising representations of photorealistic scene reconstruction. However, the non-photorealistic rendering of such a representation has barely been explored. In this study, we investigate the artistic posterization of the volumetric radiance fields. We extend the recent palette-based image-editing framework, which naturally introduces intuitive color manipulation of the posterized results, into the radiance field. Our major challenge is applying stylization effects coherently across different views. Based on the observation that computing a palette frame-by-frame can produce flickering, we propose pre-computing a single palette from the volumetric radiance field covering its entire visible color. We present a method based on volumetric visibility to sample visible colors from the radiance field while avoiding occluded and noisy regions. We demonstrate our workflow by applying it to pre-trained volumetric radiance fields with various stylization effects. We also show that our approach can produce more coherent and robust stylization effects than baseline methods that compute a palette on each rendered view.

CCS Concepts

• *Computing methodologies* → *Non-photorealistic rendering; Image processing;*

1. Introduction

Volumetric radiance fields have recently gained significant popularity as promising scene representations for neural-network applica-

tions. For example, neural radiance fields (NeRFs) [MST*20] and their variants can effectively reproduce complex view-dependent effects (e.g., occlusions and highlights) and allow for highly pho-

photorealistic free-view synthesis [MST*20, PSB*21, PSH*21]. More recently, the volumetric radiance field has been applied to more challenging tasks other than view interpolation, such as generative modeling of human faces [CMK*20] and surface appearance interpolation [BGP*21]. While previous approaches have focused on reproducing the photorealistic appearance of objects, the visualization of the radiance field with non-photorealistic styles remains unexplored. Computationally simulating the non-photorealistic styles and illustration techniques, such as perceptual abstraction and feature line drawing, helps digital artists pursue their expression and facilitates visual communication for illustrative purposes [DS02, ST90, GGSC98].

Posterization, also known as cartoon rendering, is a popular non-photorealistic rendering (NPR) technique that transforms a continuous image into regions of constant color, as is often seen in paintings, comic books, and cel animations. To achieve such effects, the classic approach for 3D models [LMHB00] relies on the lighting direction and surface geometry. However, such information is not readily available for the volumetric radiance fields, typically trained via visual supervision. Other previous approaches addressed the stylization of photographs [DS02] or videos [WOG06]. More recently, Chao et al. [CSG21] have presented palette-based posterization of images, in which a palette of the input image is used for intuitive color manipulation of the result. Although such a palette is useful for generating user-intended effects, automatic extraction of the palette is not straightforward for the volumetric radiance fields, which represent multiple views using a radiance (i.e., view-dependent color) and a scalar density field. On the other hand, extracting the palette for each rendered view results in a stylization effect that is incoherent across different views.

In this study, we investigate posterized rendering of volumetric radiance fields. We extend the palette-based approach to the volumetric radiance fields. To achieve temporally coherent stylization results, we precompute a single palette that covers the entire visible color of the field. Our algorithm extracts the palette by generating discrete color samples from the continuous radiance field and analyzing the RGB space geometry using the method based on convex hull simplification [TLG16]. However, due to several key challenges, we cannot directly apply the existing method based on a convex hull. According to the volume transmittance, a considerable portion of the radiance field is occluded and thus not visible from outside. Such invisible regions can contain significant noise even after the training converges. Moreover, the volume-rendering formulation allows the radiances emitted at different positions and directions to have different degrees of contribution to the rendered image. Hence, directly using the exhaustive radiance samples results in a low-quality palette severely affected by the noise that obscures the visible colors.

To address these challenges, we leverage the volume-rendering equation and compute the volumetric visibility of each radiance sample to adjust its effects on the palette extraction. We present a novel filtering method based on the volumetric visibility for generating fewer representative points from the raw radiance samples. Specifically, by introducing visibility weights, we extend the outlier filtering method based on k-means clustering [CSG21] to radiance samples. Since the radiance field is a function that takes five-

dimensional inputs (three for spatial location and two for view direction), the radiance sampling typically yields a prohibitively large number of samples for the k-means algorithm. We overcome this problem by performing histogram-based sample reduction, which conserves the visibility weight.

At runtime, the extracted palette is used to posterize the rendered frames. As the previous work [CSG21] focused on image posterization with high-quality region boundaries, their method typically requires minutes for a single image, which is too costly to be applied directly to animation frame-by-frame. We present a simpler method based on the RGB space geometry suitable for temporally coherent posterization of real-time free-viewpoint rendering.

We implement our palette extraction using the *PlenOctree* data structure [YLT*21] to efficiently sample from the radiance field and compute the volumetric visibility with a sparse hierarchical representation. We also use the *PlenOctree* to render the results interactively and demonstrate the effectiveness of our posterization algorithm by showing various visual effects and color adjustment results. We quantitatively and qualitatively evaluate our posterization workflow and show that our method can produce more temporally coherent stylization effects than baseline methods. We also compare our visibility-weighted palette extraction to other state-of-the-art approaches based on a convex hull and show that our method can produce a palette that better explains the visible colors in the entire radiance field, leading to higher-quality posterization results than those generated by other methods.

In summary, our contributions are as follows:

- We propose the approach to the NPR of static volumetric radiance fields. We develop palette-based posterization and recoloring system tailored to the free-view synthesis using the volumetric radiance fields.
- We present a method for palette extraction from the pre-trained volumetric radiance field. We weight the radiance samples according to the volumetric visibility for improved robustness to noise in occluded regions.
- We develop a real-time temporally coherent posterization system and demonstrate the visual effects achieved using our method. We also show that our weighted palette extraction method leads to better posterization results.

2. Related Work

2.1. Neural Radiance Fields

Neural networks have expanded the vocabulary of scene representations by replacing traditional graphic primitives [TZN19, PFS*19] or introducing a new interpretation of arbitrary 3D scenes [MST*20]. Mildenhall et al. [MST*20] viewed the scene as fields of volume density and view-dependent emitted radiance converted into a pixel color using volume rendering along a camera ray. Exploiting the differentiability of the volume rendering, they fitted a neural network to the field and achieved the state-of-the-art novel view synthesis.

This neural radiance field (NeRF) has caused an explosion of follow-up research [DL21]. Most related to ours, researchers have tackled a variety of appearance edits, such as relighting [SDZ*21],

material editing [ZSD*21], texture editing [XXH*21], and style transfer [CTT*22] to name a few. These approaches train the radiance and density (and often some other) neural networks through a custom procedure, which typically takes a long time and is subject to hyperparameter settings. In contrast, we consider the volumetric radiance field to be static data. Our method does not require neural network training and can use any pre-trained NeRF with the density and radiance fields.

One notable limitation of the original NeRF was the slow rendering time owing to the hundreds of neural network evaluations required for rendering a single pixel. This limitation has been addressed by more recent studies using either traditional graphics techniques such as empty space skipping [LGL*20] and spatial caching [YLT*21, GKJ*21], or replacing the network with a collection of smaller ones [RPLG21]. We share a similar challenge with these works. To efficiently skip unimportant regions and compute the volumetric visibility of radiance samples, we employ the *PlenOctree* data structure of Yu et al. [YLT*21]. We also use the *PlenOctree* to render our results interactively.

2.2. Non-photorealistic Rendering

NPR techniques automatically simulate artistic styles found in human-made artworks by salient geometric or chromatic features of the data. Previous research has tackled a variety of styles such as painterly [Mei96], cartoonish [LMHB00], and feature lines [ST90, DFRS03] not only for artistic purposes, but also to facilitate visual communication among people [DS02, GGSC98]. Existing works have produced stylized results from various input data types such as meshes [LMHB00], photographs [DS02], and videos [WOG06]. In this work, we address the non-photorealistic stylization of the volumetric radiance field, a data type recently introduced with the rise of neural data processing. Specifically, we extend the recent palette-based approach [CSG21] to the volumetric radiance field. Palette-based color manipulation is especially effective when users do not have free access to other intuitive handles such as surface albedo.

The non-photorealistic visualization of volume data (more specifically, 3D scalar fields) was investigated. Traditional non-photorealistic rendering (NPR) approaches for volume data such as CT or MRI scans [BKR*05, CMH*01] rely on geometric features (e.g., high surface curvature and silhouettes) of the 3D scalar field for the comprehensible visualization. Previous studies visualize such geometric features using various techniques such as directed particles [Sai94], silhouette lines [BKR*05, CMH*01], and stippling points [LME*02]. In contrast to the traditional volume data, the scalar density of the volumetric radiance field is typically noisy, ill-defined, and sensitive to training conditions, as it is trained jointly with the radiance via visual supervision. This makes it challenging to apply the traditional density-based NPR techniques directly. In this study, we rely on rich radiance information to achieve the stylization effect.

2.3. Palette Extraction

Palette-based color manipulation methods [CFL*15, TLG16, TEG18, WLX19, ZXST17, MVH*17] automatically extract a color

palette of an input image and use it as a handle for intuitive color adjustment. To extract the palette, one line of work computes the convex hull of the color samples and simplifies it to find prominent colors as the vertices of the simplified convex hull [TLG16]. Compared to alternative approaches, such as those extract palettes using k-means [CFL*15] or color models [AASP17], the methods based on RGB-space convex hulls [TLG16, TEG18, WLX19, CSG21] can extract more vibrant colors and often allow for simpler color adjustment operations. Wang et al. [WLX19] improved the RGB space convex hull method by defining energy to encourage the convex hull vertices to fit tighter to the color samples, making the palette extraction more robust to outliers. In particular for the palette-based posterization, Chang et al. [CFL*15] presented a simpler outlier filtering method based on k-means. We tailor the palette extraction via RGB space convex hulls to volumetric radiance fields by weighting color samples according to visibility. This balances the effect of radiance samples and prevents noisy but invisible colors from affecting the palette quality.

Several studies have applied the RGB-space convex hulls to extract a palette [DLX*21] or color scheme [KC20] for videos. However, such methods assume that all the frames in the video are available in advance and are thus not applicable to free-view synthesis. Although rendering sample views to use these methods is one possibility, it is unclear how to choose a camera sequence effectively covering the possible colors in the presence of complex occlusion. Instead, we extract the palette by direct sampling from the radiance field to obtain a single palette that covers the entire field.

Researchers have also investigated ways to organize palettes for effective color manipulation. Mellado et al. [MVH*17] used a graph-based representation of palettes suitable for palette modifications such as color harmonization and palette interpolation. Kim et al. [KC21] applied color sorting to find a correspondence between palettes for high-quality color transfer. Our approach is orthogonal to these studies, and the palette extracted using our method can work as a basis for these more sophisticated applications.

3. Background

To make our paper self-contained, this section briefly overviews the existing volumetric radiance field representations and palette-based color manipulation framework on which our non-photorealistic rendering technique is constructed.

3.1. Volumetric Radiance Fields

Volumetric radiance fields typically consist of density and radiance. The density field σ maps the spatial position $\mathbf{x} \in \mathbb{R}^3$ to the opacity $\sigma(\mathbf{x}) \in \mathbb{R}$ at that point. The radiance field L defines the RGB color $L(\mathbf{x}, \mathbf{d}) \in [0, 1]^3$ emitted at point \mathbf{x} in view direction $\mathbf{d} \in \mathbb{R}^3$. These fields are represented using a multilayer perceptron (MLP) [MST*20] in the NeRF. Traditional sparse spatial data structures inside an associated bounding box, such as octrees [YLT*21], have also been explored.

For an arbitrary camera ray $\mathbf{r}(t) = \mathbf{x} + t\mathbf{d}$ the NeRF [MST*20] allows us to compute the color $\mathbf{c}(\mathbf{r}) \in [0, 1]^3$ using the volume-

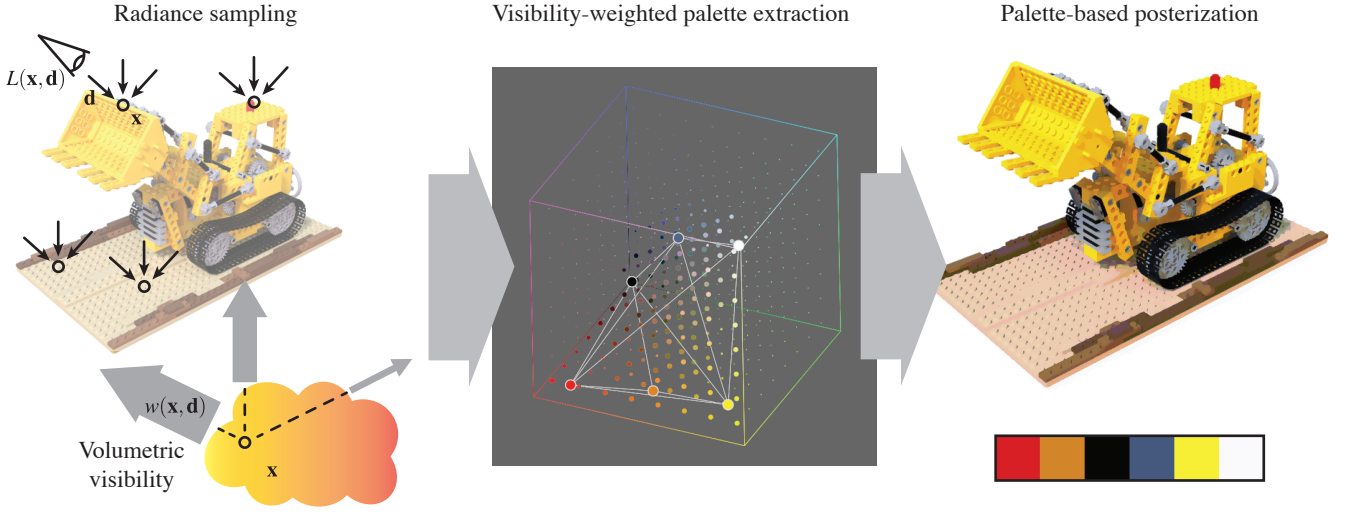


Figure 2: Workflow of our approach. For coherent stylization, we pre-compute a palette of the input volumetric radiance field by sampling the color outputs. Since occluded and noisy colors can affect the palette extraction, we utilize the density field to generate visible color samples. Specifically, we adaptively generate spatial sampling points and filter output colors based on volumetric visibility. At runtime, the extracted palette is used for the non-photorealistic posterization of the synthesized views, which naturally integrates interactive recoloring into the stylized volume rendering.

rendering equation as

$$\mathbf{c}(\mathbf{r}) = \int_0^\infty T(t; \mathbf{r}) \sigma(\mathbf{r}(t)) L(\mathbf{r}(t), \mathbf{d}) dt, \quad (1)$$

where $T(t; \mathbf{r}) \in \mathbb{R}$ denotes the transmittance function

$$T(t; \mathbf{r}) = \exp\left(-\int_0^t \sigma(\mathbf{r}(s)) ds\right). \quad (2)$$

Note that the transmittance is reciprocal, that is, $T(t; \mathbf{r}) = T(t; \mathbf{r}')$ for the inverted ray $\mathbf{r}'(s) = \mathbf{r}(t) - \mathbf{s}\mathbf{d}$.

The integral in (1) is numerically computed using the quadrature rule for volume rendering [Max95]. Specifically, a finite number of points $t_0 = 0 < t_1 < \dots < t_N \in \mathbb{R}$ are first sampled along the ray and the approximated color $\hat{\mathbf{c}}(\mathbf{r}) \in [0, 1]^3$ is computed as a Riemann sum

$$\hat{\mathbf{c}}(\mathbf{r}) = \sum_{i=1}^N T_i (1 - \exp(-\sigma_i \delta_i)) L_i, \quad (3)$$

where

$$T_i = \exp\left(-\sum_{j=0}^{i-1} \sigma_j \delta_j\right), \quad (4)$$

$\sigma_i = \sigma(\mathbf{r}(s_i))$ and $L_i = L(\mathbf{r}(s_i))$ for some $s_i \in [t_i, t_{i+1}]$, where $\delta_i = t_{i+1} - t_i$ are the lengths of the corresponding intervals. The accumulated transmittance T_N is used to composite $\hat{\mathbf{c}}(\mathbf{r})$ and the background, that is, the final color is obtained as $(1 - T_N)\hat{\mathbf{c}}(\mathbf{r}) + T_N\mathbf{c}_0$, where \mathbf{c}_0 is the background color typically given as a hyperparameter.

PlenOctrees In our experiments, we use the PlenOctree [YLT*21] to efficiently generate spatial sampling points, evaluate the density

and radiance functions, and compute volumetric quantities. The PlenOctrees drastically reduced the computational cost of evaluating the sum in (3) by replacing the deep MLP with a hierarchical data structure called PlenOctree to efficiently query the density and radiance at a given spatial point. They specifically use a sparse octree whose leaf node stores the voxel's average density and spherical harmonic (SH) coefficients that parameterize color in the directional domain, making the radiance evaluation extremely lightweight compared to that of large MLPs. As a result, the PlenOctree allows us to efficiently compute (3) by generating the sampling points as ray-voxel intersections and using the information cached at the voxels.

The leaf voxels are chosen from a regular grid by first sampling the density field on the grid and then filtering transparent voxels by thresholding the density values. To further encourage the sparsity, voxels not visible from any training view are eliminated by tracking the alpha values. The final octree structure is then constructed with the remaining leaf voxels each of which stores the averaged density and SH coefficients inside it. For more details on PlenOctree, please refer to the original work [YLT*21].

3.2. Palette-based Color Manipulation

Our posterization workflow adopts a palette-based approach [TLG16, TEG18] that enables intuitive recoloring of images, and has more recently been applied to artistic image posterization [CSG21]. We briefly introduce the relevant terms and notations.

A color palette is a finite set of RGB colors $\mathcal{P} = \{\mathbf{p}_i\}$, where $\mathbf{p}_i \in [0, 1]^3$. It is used to decompose a given pixel color $\mathbf{c} \in [0, 1]^3$

as an additive blending of the palette colors as

$$\mathbf{c} = \sum_i \beta_i \mathbf{p}_i, \quad (5)$$

where $\beta_i \in \mathbb{R}$ are non-negative blending weights that add up to one. Given such decomposition weights, users can recolor the image by modifying the palette color. Specifically, previous studies [TEG18, WLX19, CSG21] allow users to specify a modified palette $\mathcal{P}' = \{\mathbf{p}'_i\}$ and recolor the pixel colors as

$$\mathbf{c}_{\text{recolored}} = \sum_i \beta_i \mathbf{p}'_i. \quad (6)$$

Palette Extraction Tan et al. [TLG16, TEG18] rely on the geometric structure formed by image pixels in the RGB space and automatically extract a plausible palette by simplifying the convex hull of the image pixels. For more details about the palette extraction algorithm, please refer to their original work [TLG16, TEG18]. We adopt their method based on a convex hull to extract a palette from volumetric radiance fields. However, generate high-quality discrete color samples must be generated from a continuous radiance field to analyze the RGB space geometry. In the method section, we explain how to perform the sampling.

Palette-based Artistic Posterization We follow the palette-based approach by Chao et al. [CSG21] to define discrete color labels used for the posterization. They first extract a palette from the input image as the corner of the convex hull in the RGB space. They then define a set of color labels \mathcal{L} as a linear blending of all combinations of two colors in the palette colors. In other words, the color label is defined as

$$\mathcal{L} = \{\gamma_k \mathbf{p}_i + (1 - \gamma_k) \mathbf{p}_j \mid k = 1, 2, \dots, d - 1 \text{ and } i < j\}, \quad (7)$$

where $\mathbf{p}_i, \mathbf{p}_j \in \mathcal{P}$ and $\gamma_k = k/d$ is a discrete blending weight, and $d \in \mathbb{N}^+$ is a user-specified blending step. Finally, the image is posterized by assigning one of the labels in \mathcal{L} to each pixel, and the result can be intuitively recolored via the palette-based manipulation. Although this workflow is effective for images, when applied frame-by-frame, extracting a palette individually for each frame results in flickering. We present a method to overcome this challenge in interactive free view-synthesis using volumetric radiance fields.

4. Method

Our workflow uses a pre-trained volumetric radiance field as the input and artistically posterizes the synthesized views. We employ the palette-based approach based on Chao et al. [CSG21]. However, their method mainly addresses image posterization, and it is difficult to achieve temporal coherency when applied frame-by-frame to the video. Directly applying their method to each frame results in flickering artifacts owing to the incoherence of the palette colors between adjacent frames.

To address this challenge, we precompute a single palette that foresees all visible colors in the volumetric radiance field. At run-time, the extracted palette is used for temporally coherent posterization and recoloring of the rendered images in the screen space. In the remainder of this section, we present our palette extraction from volumetric radiance fields and the details of the run-time posterization. Figure 2 shows the overview of our workflow.

4.1. Palette Precomputation

For the palette extraction, we employ the approach of Tan et al. [TLG16], which based on the RGB-space geometry. However, the previous palette extraction technique is not directly applicable because of several key challenges. First, unlike images or videos, the radiance field is a continuous function, so we need to generate discrete color samples corresponding to the image pixels in order to analyze it through the RGB-space geometry. Second, as the radiance field has view-dependent spatial information, the sampled data are typically several orders of magnitude larger than the image pixel data. This makes the color analysis significantly more expensive if computation is done without reducing the sample count via adaptive sampling or sample filtering. Furthermore, each color sample has a different contribution to the camera ray owing to the occlusion in the volume (1). In other words, not all colors are visible from outside the volume and such regions in the radiance field are typically not optimized well via visual supervision. We need to carefully avoid such unsupervised noisy colors.

Our approach is tailored to overcome these challenges. We introduce the volumetric visibility of each five-dimensional input point in the field to effectively weigh important color samples in the analysis. We also adaptively determine the spatial sampling points to efficiently avoid sampling from irrelevant regions. We further reduce the number of samples using a color histogram to make the computation time of the subsequent stages independent of the number of color samples.

Volumetric Visibility If we evenly sample the entire volumetric radiance field, the palette extraction is easily affected by its noise or under/overestimate some color samples. This is because, whereas all pixels of images and videos are visible as they are, a considerable portion of the radiance field is not visible from the outside owing to the volume rendering which typically remains unsupervised after training (see Figure 3). As the density field is continuous, the extent of visibility varies continuously in the spatial and directional domains. To address this problem, we propose assigning a continuous weight to each color sample based on the density and using it in the palette extraction stages. We refer to the weight *volumetric visibility*.

For each pair of spatial points \mathbf{x} and view direction \mathbf{d} , we define

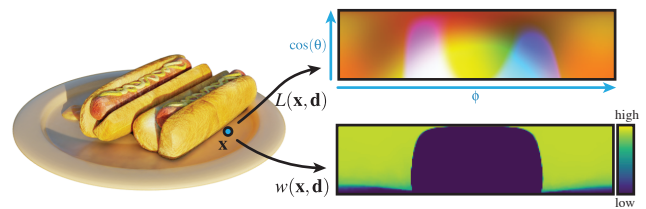


Figure 3: Visualization of radiance and volumetric visibility. There is a low-visibility region as shown in the middle of the right-bottom image owing to the occlusion by the hot dog. The radiance in the corresponding region (right-top image) has noisy colors (e.g., blue, pink, and green). Such unsupervised color noise can be a challenge in automatic palette extraction.

the volumetric visibility $w(\mathbf{x}, \mathbf{d})$ as

$$\begin{aligned} w(\mathbf{x}, \mathbf{d}) &= T(\infty; \mathbf{r})\sigma(\mathbf{x}) \\ &= \exp\left(-\int_0^\infty \sigma(\mathbf{r}(t)) dt\right) \sigma(\mathbf{x}), \end{aligned} \quad (8)$$

where $\mathbf{r}(t) = \mathbf{x} - t\mathbf{d}$ is the ray starting at \mathbf{x} in the inverted direction $-\mathbf{d}$. The reciprocity of the transmittance allows us to consider this quantity as an infinitesimal weight in the volume-rendering integral in (1) of the emitted color $L(\mathbf{x}, \mathbf{d})$ viewed in the direction \mathbf{d} from an infinitely distant observer at $\mathbf{x} + \infty\mathbf{d}$. We hypothesize that this volumetric visibility can be used as an estimate of the importance of the radiance sample in the synthesized views and use it to adjust the contribution of each sample to the palette extraction algorithm. In our experiments, we efficiently computed the value $w(\mathbf{x}, \mathbf{d})$ using the PlenOctree via the quadrature rule in (4).

Density-adaptive Radiance Sampling To analyze the RGB-space geometry, we first need to discretely sample colors from the continuous volumetric radiance field. We draw such samples by generating spatial-directional inputs and evaluating the radiance field for these inputs. We call the color samples generated with this method *radiance samples* to clarify that the colors were produced from a radiance field, and each of them is associated with a spatial-directional point. As brute-force sampling from such a five-dimensional domain can easily produce a prohibitively large number of samples, we leverage the sparsity of the density field and adaptively generate N_{sp} spatial sampling points at locations with a non-zero density value. The N_{dir} viewing directions are then regularly sampled at each of the generated locations. We hence generate $N_{\text{sp}}N_{\text{dir}}$ radiance samples in total. In terms of the volumetric visibility, this adaptive sampling can be justified as any radiance sample at location \mathbf{x} with zero density $\sigma(\mathbf{x}) = 0$ has zero visibility in (8). Because most of the pre-trained volumetric radiance fields used in our experiments were optimized by training views seen from the upper hemisphere, we generate the viewing directions seen from the upper hemisphere.

Spatial sampling points are efficiently generated using the PlenOctree. Because of its construction, spatial positions that are not included in the non-empty leaf nodes of the tree can be considered to have zero density. In contrast, every non-empty leaf voxel has a positive average density. Hence, we perform a tree search and place the spatial sampling points at the center of all the non-empty leaf voxels. In this case, N_{sp} is the number of non-empty leaves in the PlenOctree model, representing the input volumetric radiance field.

Regular sampling of a specified number of directions is not straightforward. In our experiments, we project the hemispherical domain to the unit square using the cylindrical map as in [MGN17] and regularly generate view directions on the square. We further anti-alias the radiance samples through multisampling around each view direction. For specific details of the directional sampling, see Appendix A.

As our experiments use PlenOctree models with 9 spherical harmonic bases per channel, it is not worth sampling substantially more than 9 directions (note that the bases are slightly more expressive because of the sigmoid function applied at the top). Nev-

ertheless, a large number of raw direction samples is still useful considering the geometric complexity. For further efficient generation of the radiance samples, we consider the density and the volumetric visibility under corresponding thresholds to be zero. We use 1×10^{-2} for the thresholds following the similar parameter used for rendering the PlenOctree in [YLT*21].

Visibility-weighted Sample Filtering The radiance samples still contain a considerable portion of invisible (and thus noisy) colors. This is partly because we do not consider volumetric visibility to generate view directions but regularly sample them. It is desirable to remove such noise before extracting a palette via RGB space geometry as the shape of a convex hull can be easily affected by outlier colors, which leads to a palette that does not respect the true color distribution. For palette extraction from images, an outlier removal technique based on k-means clustering [CSG21] exists. However, radiance samples are significantly noisier and several orders of magnitude larger in number than image pixels, which limits the effectiveness of the k-means algorithm in terms of its noise-reduction effect and computational cost.

To overcome this problem, we use a histogram of the radiance samples weighted by their volumetric visibility. Specifically, we compute a histogram in the RGB space where each bin stores the total volumetric visibility of the samples inside it. This not only drastically reduces the number of samples but also lowers the effect of less visible colors. The non-empty bins are then clustered using k-means with a large K to further eliminate remaining outlier colors. In our experiments, we use a histogram with $32 \times 32 \times 32$ bins.

As it is not always straightforward for users to choose an appropriate number K of clusters for a given volumetric radiance field, we offer a method for automatically determining the value of K with respect to a threshold parameter $\tau \in \mathbb{R}$. Specifically, we first compute a coarse histogram with $8 \times 8 \times 8$ bins from the radiance samples weighted by the volumetric visibility. We normalize the bin weights and use the centers of bins whose weights are above τ as the initial cluster centers. Note that such carefully chosen initial cluster centers also help the k-means algorithm to efficiently and robustly find the final cluster centers. We empirically use the threshold $\tau = 8 \times 10^{-3}$ consistently for generating our results.

Finally, the obtained cluster centers are passed to the convex hull analysis and we use the generated palette in our runtime posterization algorithm. We specify the final palette size of the convex hull method to be 6 in our experiments.

4.2. Run-time Posterization

At runtime, we posterize the synthesized views using a pre-computed palette. We follow the approach of Chao et al. [CSG21] to define discrete color labels using a palette in (7). In this study, we consistently use $d = 3$ in (7) and also add the original palette colors to the labels. Posterization is then computed by assigning one of the labels to each pixel. However, producing abstracted images with smooth regional boundaries is not straightforward. Previous works [CSG21, XK08] formulated the label assignment as a combinatorial optimization problem to obtain smooth region boundaries,

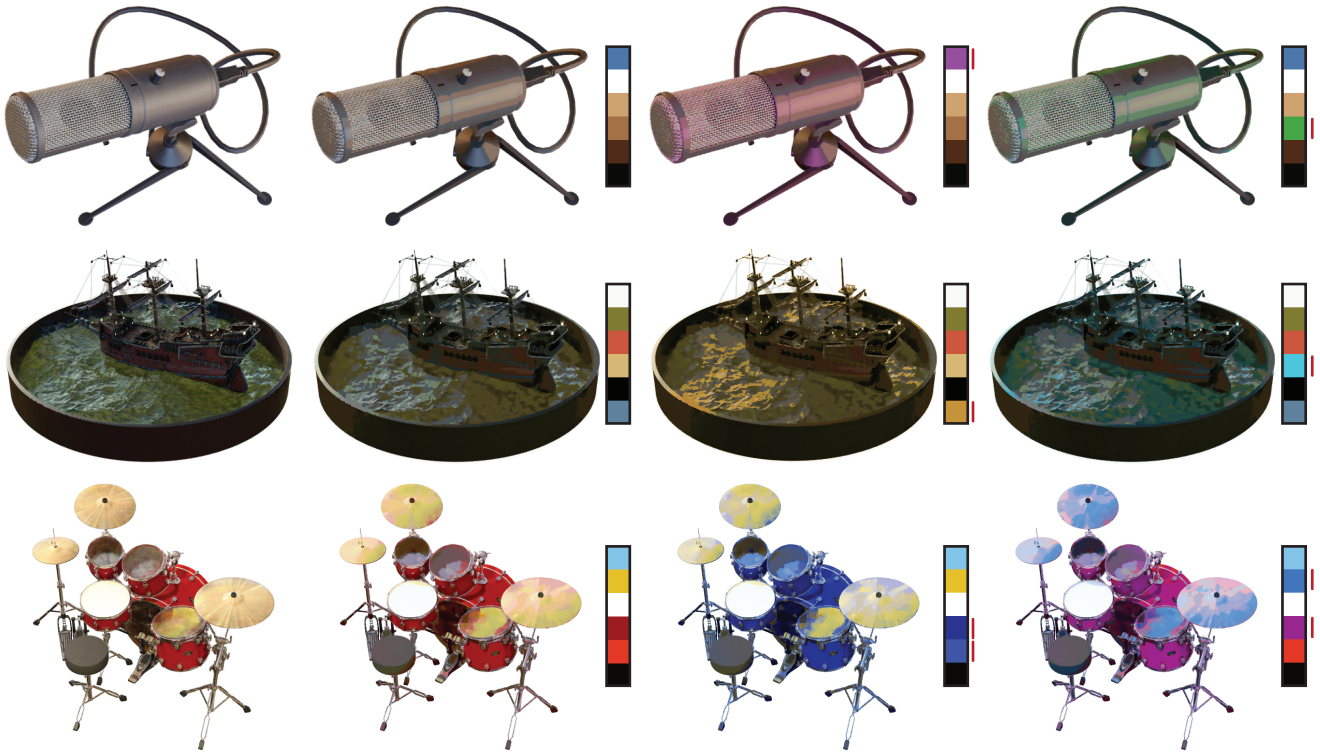


Figure 4: Gallery of stylization results. Original renderings were posterized using the palettes extracted by our visibility-weighted method and recolored changing the marked palette colors (left-to-right).

but it can take minutes to process a single image and can be too costly for the interactive free-view synthesis. Notably, the video stylization approach of Winnemöller et al. [WOG06] utilizes efficient iterative bilateral filtering to obtain smoothly stylized images in real-time. However, their workflow relies on a thresholding of the luminance channel to reproduce a posterized look and thus does not support arbitrary color labels.

Inspired by the video abstraction approach [WOG06], we abstract synthesized frames with iterative bilateral filtering and then assign labels to each pixel using nearest-neighbor projection. We use an efficient separated-kernel approximation [PvV05] of the bilateral filter. For the nearest-neighbor projection, we use the RGB

distance. Although the perceptually uniform CIELAB color space can potentially be used, our color labels are defined using the RGB space geometry, and we sometimes see that the CIELAB space distort the original geometry, resulting in loss of details (Figure 5). Even though the label assignment in the RGB space can potentially produce inaccurate colors, we observe that it is often not severely problematic.

Soft Label Assignment Smooth transition between discrete color labels is useful for reducing abrupt boundaries and a temporal flickering. We incorporate this to our label assignment by smoothly blending the second-nearest label when the distances D_1 and D_2 to the first- and second-nearest labels are nearly equal. We allow the user to specify a step width parameter D and normalize the difference $D_2 - D_1$ as $\bar{\alpha} = 0.5 + (D_2 - D_1)/2D \in [0.5, \infty]$. We then smoothly clamp the $\bar{\alpha}$ to the range $[0.5, 1]$ using a smoothstep function as $\alpha = 3\bar{\alpha}^2(1 - 2\bar{\alpha})$, and assign the linear blending of first- and second-nearest labels with weights α and $1 - \alpha$ to the pixel. We use $D \in [0.1, 0.4]$ to generate the stylization results in our experiment. We evaluate the effect of different values of D in the result section.

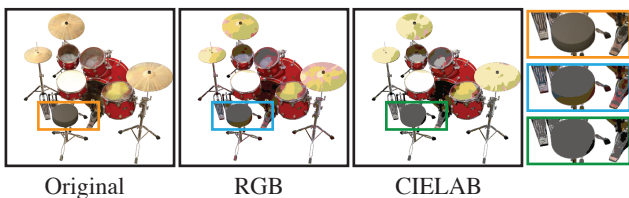


Figure 5: Comparison of label assignment results using the RGB (middle) and the CIELAB (right) color spaces. Both approaches generate a palette in the RGB space. The label assignment in the CIELAB space leads to the loss of details in several regions.

To save computation time, we do not convert the pixels with zero accumulated transmittance T_N in (4) as they already have a constant background color. We also avoid modifying the pixels with an accumulated transmittance T_N below a threshold $\lambda \in \mathbb{R}$. Converting these pixels sometimes allows semi-transparent regions with colors similar to the background to appear in the result as undesirable

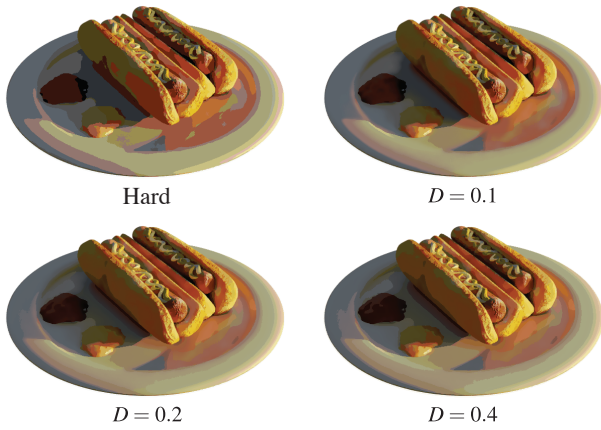


Figure 6: Evaluation of the step width parameter D in our label assignment. Different values of D results in different boundary sharpness. See text for more discussion.

colored floating clouds. Note that such semi-transparent regions are often seen in volumetric radiance fields, owing to the under-constrained nature of the visual supervision. In our experiment, we set $\lambda = 80/255$ empirically.

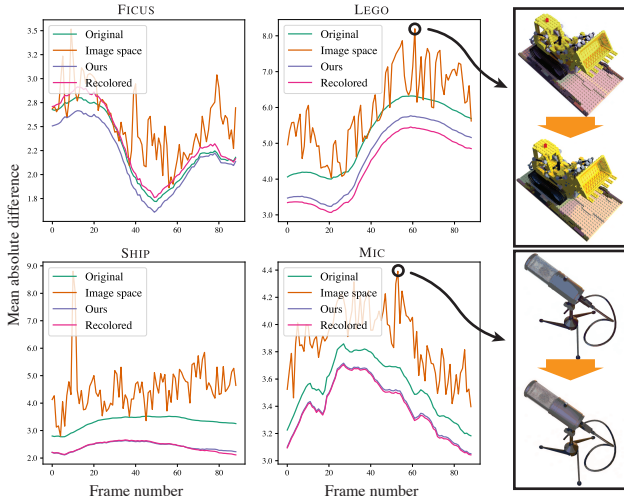


Figure 7: Temporal coherency. The mean absolute difference between adjacent image frames of stylized animations produced using different palette extraction approaches are compared. The animation sequence is generated by rotating the camera by approximately 90° around the center of the model. We show two example frame transitions that causes a flickering in the animation generated by the image-space method. Please refer to our supplemental material for the entire videos.

5. Results

We apply our method to pre-trained PlenOctree models published by the authors of [YLT*21] and show the results in Figures 1 and 4. Our method abstracts the rendering into constant color regions while also fairly preserving fine details such as the fine texture on the MIC and LEGO models. Palette-based color labels naturally allow for localized recoloring of the stylized rendering. In these Figures, we illustrate this by modifying the colors of each palette. For instance, we can see that the red drum can be turned blue or the blue highlight in the SHIP model can be made orange to produce a look and feel of a sunset.

Temporal Coherency We evaluate the temporal coherency of our posterization approach with a single pre-computed palette by comparing it to the baseline approach where a palette is recomputed for

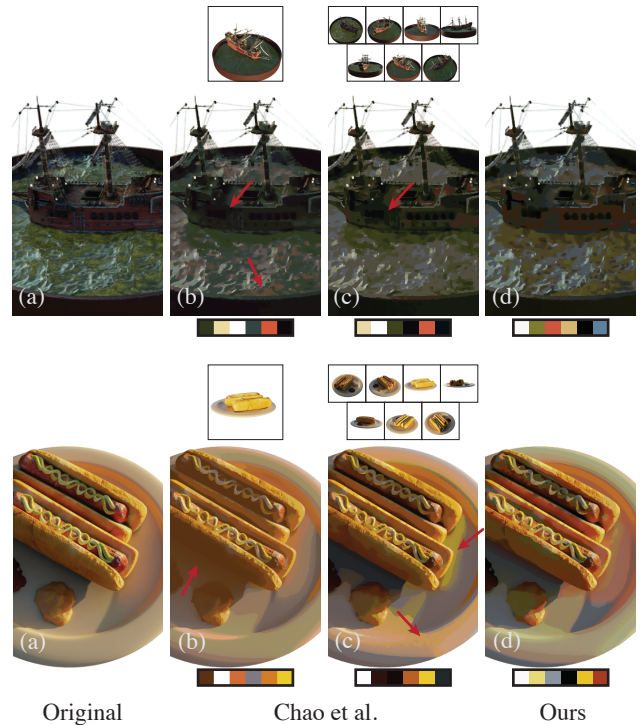


Figure 8: Comparison with palette extraction from example views. We posterized a rendered view (a) using palettes generated by three methods: the method of Chao et al. [CSG21] applied to one view (b), applied to the concatenation of seven views (c), and our method (d). The input views for (c), shown above the corresponding results, were evenly selected from the test poses in the NeRF-synthetic dataset, and one among them is used for (b). Due to directional color variations and spatial effects such as occlusion and camera distortion, we observe that the baseline approaches (b) and (c) result in less accurate colors (see areas indicated by the arrows) than those produced by our method (d). Note that these test poses are not readily available when the models are streamed to end-users or created using generative models [CMK*20]. See Appendix B for specific experimental conditions.

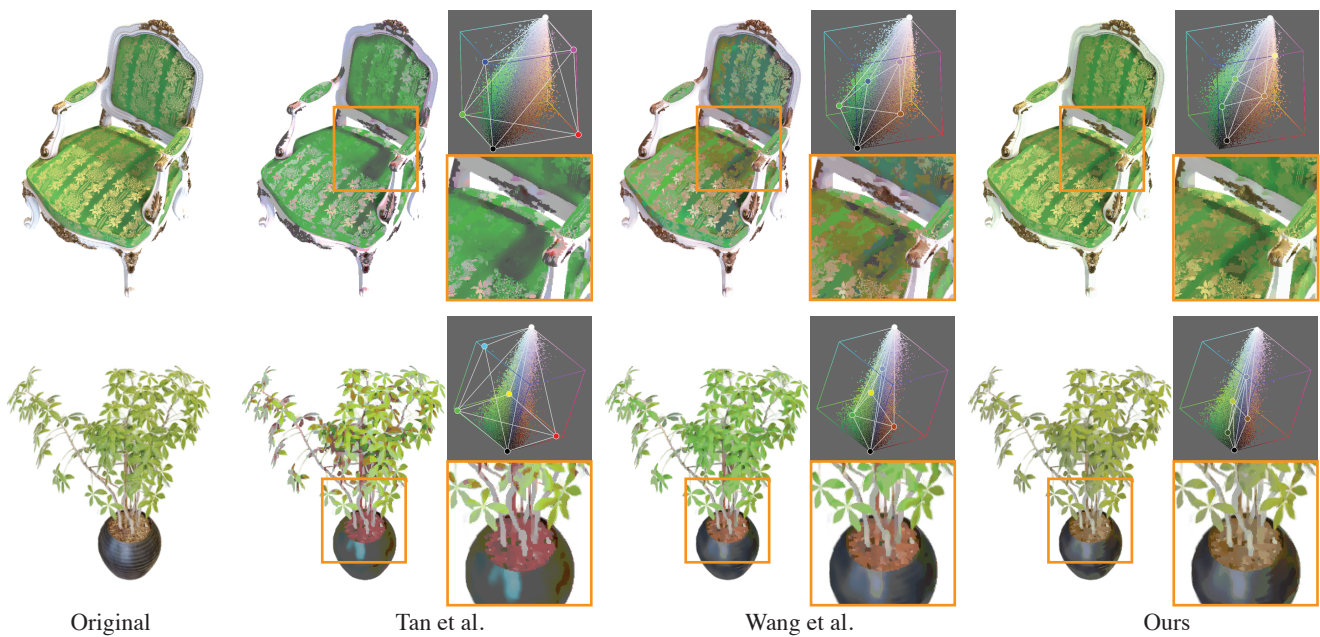


Figure 9: Robustness of different palette extraction approaches to noise. Our method based on volumetric visibility better deals with radiance samples compared to naïve application the algorithm of Tan et al. [TLG16] via the RGB convex hull or a more sophisticated geometric palette extraction approach of Wang et al. [WLG19] based on optimization. To create the results for the baseline methods, we used 640000 ($= 800 \times 800$) colors from the raw radiance samples as input. Consequently, our palette produces stylization effects with more accurate colors than those extracted using other techniques.

each rendered frame. Short animations were created by interpolating key camera poses. We use the method of Tan et al. [TLG16] to extract a palette from the rendered frames. To make the comparison fair and clear, we always use our label assignment based on nearest neighbor projection to generate posterization using the computed palettes.

Figure 7 plots the average magnitude of the pixel-wise difference between adjacent stylized frames produced by the different methods. We can see that the baseline approach generally has a higher difference value and frequent spikes in the plot. This indicates that recomputing a palette frame by frame leads to incoherent and unstable posterization results. In contrast, our single-palette approach consistently exhibits small difference values, suggesting that it can produce more temporary coherent results. We also present an error for the recolored animations and the results show that our method enables temporary coherent recoloring effects. We provide the videos used for creating the plots in our supplemental material. Note that palettes varying each frame also make the recoloring much less straightforward.

Note that one can also pre-compute a palette from one or a set of example views. We observe that this approach can produce similar results to ours when the views are carefully chosen. However, for models with a rich directional color variation or complicated geometry, finding a set of camera projections that accurately represent the distribution of visible colors in the 5D spatial-directional space is not always trivial. In contrast, our approach directly works in the spatial-directional space, allowing users to capture the actual color

distribution without manually specifying representative views. In Figure 8, we compare our method with baseline approaches extracting a palette from one view or the concatenation of seven views. The views were created using camera poses in the NeRF-synthetic dataset [MST*20] which the models in our experiments were trained with. We present a few examples where the baseline methods result in less accurate colors than ours for a view not included in the example views.

Comparison of Palette Extraction Methods We compare our palette extraction based on visible radiance samples to other state-of-the-art approaches for natural images based on the RGB space geometry. The palettes extracted using these methods are shown in Figure 9. Directly applying the method of Tan et al. [TLG16], the simplified convex hull is severely affected by noisy colors and resulting in a palette with spurious colors. This suggests that volumetric radiance fields trained via visual supervision can still contain a considerable portion of noisy colors even though they are not apparent owing to occlusion. As the noise in the raw radiance samples are significantly more intense than that in natural images the optimization-based approach, Wang et al. [WLG19] fails to completely remove the noisy palette colors. In contrast, our visibility-weighted method effectively captures the geometry of the visible colors, which results in posterization results with more accurate colors.

We also conducted an ablation study of visibility weights used in our sample filtering. Figure 10 shows palettes and the correspond-

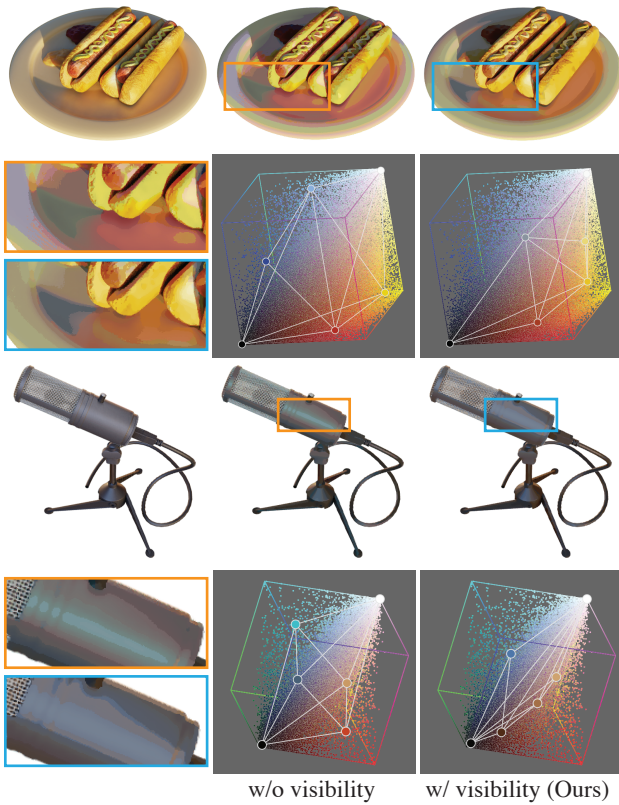


Figure 10: Ablation of the use of visibility weights in our sample filtering based on k -means. For the method without the visibility weights, we run k -means on 640000 ($= 800 \times 800$) colors randomly sampled from the raw radiance samples. For a clear comparison, we use the initial cluster centers generated using our full filtering method also in the baseline method. In the results, we can see that the palettes extracted using our visibility-weighted k -means (right) more plausibly captures the RGB space geometry of the radiance samples than those generated by method without visibility (center). Consequently, the palettes extracted using our full method more faithfully preserve the original colors in the posterization results.

ing stylization results generated using the sample filtering with or without the visibility weights. Although the method without the visibility weights exhibits a noise reduction effect, our full method generates more compact and plausible geometric palettes. Consequently, our palettes results in posterization that more faithfully captures the color of the original renderings.

Evaluation of Soft Label Assignment We evaluate the parameter D of our runtime soft label assignment. In Figure 6, we present posterization results generated using different values of D . Without the soft label assignment, the result exhibits sharp boundaries between colors with high contrast. Our label assignment using a small D allows us to smooth these boundaries by allowing for smooth transitions between color labels. It is interesting to see that a large D introduces hard boundaries again owing to the discontinuity of

Model	N_{sp}	K	Rad.		Hull	Total	FPS
			samp.	Filt.	simp.		
CHAIR	3.0M	38	4.02s	1.61s	0.46s	6.09s	60
DRUMS	3.8M	34	7.56s	2.10s	0.15s	9.82s	30-60
FICUS	3.5M	42	9.87s	1.87s	0.17s	11.9s	30-60
HOT.	4.3M	38	7.84s	2.38s	0.19s	10.4s	30-60
LEGO	4.5M	36	7.71s	2.67s	0.11s	10.5s	30-60
MAT.	3.4M	23	6.12s	1.87s	0.34s	8.33s	30-60
MIC	1.5M	26	2.60s	0.81s	0.07s	3.49s	60
SHIP	5.5M	26	14.0s	3.46s	0.11s	17.6s	20-30

Table 1: Performance breakdown of our palette extraction. We report the execution time of radiance radiance sampling (Rad. sampl.), sample filtering (Filt.), and convex-hull simplification (Hull simpl.), along with the spatial sampling size N_{sp} and the number of the clusters K in sample filtering. We also present typical frame rates (FPS) of our runtime interactive posterization. We implemented the radiance sampling using CUDA and the sample filtering using C++ for the histogram construction and SciPy for k -means. For the convex hull simplification, we use the Python code published by the authors of [TEG18]. The running times were generated on a 3.60 GHz Intel Core i7-6850K and an NVIDIA GeForce GTX 1070. Note that the sample filtering stage takes dozen minutes (or even an hour) without using a histogram. The abbreviations HOT. and MAT. in the table represent HOTDOG and MATERIALS.

second-and-third-nearest neighbors. However, these boundaries are more natural than those generated by the hard label assignment because of intermediate colors additionally introduced as a linear blending of the first-and-second-nearest labels. The parameter D works as a handle for users to tweak between smooth (thus more temporary coherent) boundaries and harder ones that produce a look and feel of cartoon animations.

Performance Table 1 presents the timing analysis of our palette extraction and run-time posterization. Both volumetric visibility in palette pre-computation and an original rendering at runtime can be efficiently computed owing to the sparse hierarchical structure of the PlenOctree. It is noteworthy that the volumetric visibility can be computed fully in parallel for each radiance sample. Our GPU implementation can generate radiance samples within a few seconds. In addition, sample filtering based on a histogram drastically reduces the subsequent K -means clustering and the palette extraction via the RGB space convex hull. As a result, given a PlenOctree model, the entire palette extraction does not take too long (typically around ten seconds).

Our runtime stylization method can be efficiently implemented as a GPU shader, which allows us to interactively posterize and recolor views rendered using the PlenOctree. In our supplemental material, we present a screen capture of our run-time application.

6. Discussion and Future Work

We used volumetric visibility and successfully filtered noise in radiance samples. Although we used simple weighted k -means with a large cluster count, other mode-seeking methods such as mean shift [Che95], which outperforms k -means in several tasks, could

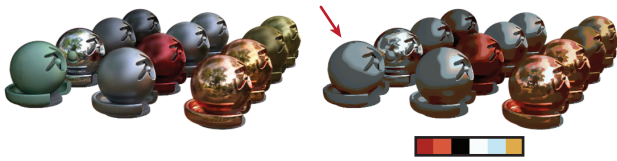


Figure 11: A failure case. Palette extraction based on the RGB convex hull can fail to include an important color.

be used. However, it is not obvious how to adapt these more sophisticated methods for our noise elimination purpose. In the future, we plan to consider these methods to achieve higher-quality results.

Our approach based on convex hull analysis produced plausible palettes in most tested cases. However, palette extraction purely based on a convex hull can fail to find important color that does not have geometric saliency in the RGB space (e.g., colors on or inside the convex hull). We see one such failure case in our experiments (Figure 11). To overcome such situations, incorporating optimization-based approach [WLX19, AASP17] will be a promising direction. We believe that it will be interesting to use the volumetric visibility to extend such optimization-based color analysis to volumetric radiance fields.

Incorporating other non-photorealistic styles is also an interesting future direction. For example, we plan to support the feature lines commonly seen in cartoon animations. To visualize the feature lines consistently across different views, utilizing geometric information in the density field, as done in previous studies [BKR*05] will be interesting. However, the density field is often noisy due to under-constrained visual supervision to train the volumetric radiance field. Addressing such geometric noise will be an important future direction to extend the vocabulary of NPR styles for visualizing volumetric radiance fields.

In this study, we only use synthetic scenes with the constant background color. In a future work, we plan to test our method on models with a rich background trained by real photographs. To support these scenes, we need to devise a more sophisticated radiance sampling strategy. We also plan to apply our method to other data structures, such as the one using small neural networks [RPLG21] to represent the entire volumetric radiance field.

7. Conclusion

We present a method to render volumetric radiance fields with a non-photorealistic posterization style that can be intuitively recolored using a palette. We have shown that precomputing the palette from the volumetric radiance field leads to a coherent stylization across different views. We propose using volumetric visibility to eliminate noisy radiance samples that otherwise affect the palette extraction quality. Compared with alternative approaches, our system produces more temporary coherent and natural posterization effects for interactive free-view synthesis.

Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments and suggestions.

References

- [AASP17] AKSOY Y., AYDIN T. O., SMOLIĆ A., POLLEFEYS M.: Unmixing-based soft color segmentation for image manipulation. *ACM Trans. Graph.* 36, 2 (2017), 19:1–19:19. 3, 11
- [BGP*21] BAATZ H., GRANSKOG J., PAPAS M., ROUSSELLE F., NOVÁK J.: Nerf-tex: Neural reflectance field textures. In *Eurographics Symposium on Rendering* (June 2021), The Eurographics Association. 2
- [BKR*05] BURNS M., KLAWE J., RUSINKIEWICZ S., FINKELSTEIN A., DECARLO D.: Line drawings from volume data. *ACM Trans. Graph.* 24, 3 (jul 2005), 512–518. 3, 11
- [CFL*15] CHANG H., FRIED O., LIU Y., DIVERDI S., FINKELSTEIN A.: Palette-based photo recoloring. *ACM Transactions on Graphics (Proc. SIGGRAPH)* 34, 4 (July 2015). 3
- [Che95] CHENG Y.: Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17, 8 (1995), 790–799. 10
- [CMH*01] CSÉBFALVI B., MROZ L., HAUSER H., KÖNIG A., GRÖLLER E.: Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum* 20, 3 (2001), 452–460. 3
- [CMK*20] CHAN E., MONTEIRO M., KELLNHOFFER P., WU J., WETZSTEIN G.: pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *arXiv* (2020). 2, 8
- [CSG21] CHAO C.-K., SINGH K., GINGOLD Y.: PosterChild: Blend-Aware Artistic Posterization. *Computer Graphics Forum* (2021). 2, 3, 4, 5, 6, 8, 12
- [CTT*22] CHIANG P.-Z., TSAI M.-S., TSENG H.-Y., LAI W.-S., CHIU W.-C.: Stylizing 3d scene via implicit representation and hypernetwork. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (January 2022), pp. 1475–1484. 3
- [DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Trans. Graph.* 22, 3 (jul 2003), 848–855. 3
- [DL21] DELLAERT F., LIN Y.: Neural volume rendering: Nerf and beyond. *CoRR abs/2101.05204* (2021). [arXiv:2101.05204](https://arxiv.org/abs/2101.05204). 2
- [DLX*21] DU Z.-J., LEI K.-X., XU K., TAN J., GINGOLD Y.: Video recoloring via spatial-temporal geometric palettes. *ACM Transactions on Graphics (TOG)* 40, 4 (Aug. 2021). 3
- [DS02] DECARLO D., SANTELLA A.: Stylization and abstraction of photographs. *ACM Trans. Graph.* 21, 3 (jul 2002), 769–776. 2, 3
- [GGSC98] GOOCH A., GOOCH B., SHIRLEY P., COHEN E.: A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, Association for Computing Machinery, p. 447–452. 2, 3
- [GKJ*21] GARBIN S. J., KOWALSKI M., JOHNSON M., SHOTTON J., VALENTIN J.: Fastnerf: High-fidelity neural rendering at 200fps. *arXiv preprint arXiv:2103.10380* (2021). 3
- [KC20] KIM S., CHOI S.: *Automatic Color Scheme Extraction from Movies*. Association for Computing Machinery, New York, NY, USA, 2020, p. 154–163. 3
- [KC21] KIM S., CHOI S.: Dynamic closest color warping to sort and compare palettes. *ACM Trans. Graph.* 40, 4 (jul 2021). 3
- [LGL*20] LIU L., GU J., LIN K. Z., CHUA T.-S., THEOBALT C.: Neural sparse voxel fields. *NeurIPS* (2020). 3
- [LME*02] LU A., MORRIS C., EBERT D., RHEINGANS P., HANSEN C.: Non-photorealistic volume rendering using stippling techniques. In *IEEE Visualization, 2002. VIS 2002.* (2002), pp. 211–218. 3

- [LMHB00] LAKE A., MARSHALL C., HARRIS M., BLACKSTEIN M.: Stylized rendering techniques for scalable real-time 3d animation. In *Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2000), NPAR '00, Association for Computing Machinery, p. 13–20. [2](#), [3](#)
- [Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics* 1, 2 (1995), 99–108. [4](#)
- [Mei96] MEIER B. J.: Painterly rendering for animation. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1996), SIGGRAPH '96, Association for Computing Machinery, p. 477–484. [3](#)
- [MGN17] MÜLLER T., GROSS M., NOVÁK J.: Practical path guiding for efficient light-transport simulation. *Comput. Graph. Forum* 36, 4 (jul 2017), 91–100. [6](#)
- [MST*20] MILDENHALL B., SRINIVASAN P. P., TANCİK M., BARRON J. T., RAMAMOORTHI R., NG R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV* (2020). [1](#), [2](#), [3](#), [9](#)
- [MVH*17] MELLADO N., VANDERHAEGHE D., HOARAU C., CHRISTOPHE S., BRÉDIF M., BARTHE L.: Constrained palette-space exploration. *ACM Trans. Graph.* 36, 4 (jul 2017). [3](#)
- [PFS*19] PARK J. J., FLORENCE P., STRAUB J., NEWCOMBE R., LOVEGROVE S.: DeepSDF: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019). [2](#)
- [PSB*21] PARK K., SINHA U., BARRON J. T., BOUAZIZ S., GOLDMAN D. B., SEITZ S. M., MARTIN-BRUALLA R.: Nerfies: Deformable neural radiance fields. *ICCV* (2021). [2](#)
- [PSH*21] PARK K., SINHA U., HEDMAN P., BARRON J. T., BOUAZIZ S., GOLDMAN D. B., MARTIN-BRUALLA R., SEITZ S. M.: Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.* 40, 6 (dec 2021). [2](#)
- [PvV05] PHAM T., VAN VLIET L.: Separable bilateral filtering for fast video preprocessing. In *2005 IEEE International Conference on Multimedia and Expo* (2005), pp. 4 pp.–. [7](#)
- [RPLG21] REISER C., PENG S., LIAO Y., GEIGER A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2021), pp. 14335–14345. [3](#), [11](#)
- [Sai94] SAITO T.: Real-time previewing for volume visualization. In *Proceedings of the 1994 Symposium on Volume Visualization* (New York, NY, USA, 1994), VVS '94, Association for Computing Machinery, p. 99–106. [3](#)
- [SDZ*21] SRINIVASAN P. P., DENG B., ZHANG X., TANCİK M., MILDENHALL B., BARRON J. T.: Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR* (2021). [2](#)
- [ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, Association for Computing Machinery, p. 197–206. [2](#), [3](#)
- [TEG18] TAN J., ECHEVARRIA J., GINGOLD Y.: Efficient palette-based decomposition and recoloring of images via rgbxy-space geometry. *ACM Transactions on Graphics (TOG)* 37, 6 (Dec. 2018), 262:1–262:10. [3](#), [4](#), [5](#), [10](#)
- [TLG16] TAN J., LIEN J.-M., GINGOLD Y.: Decomposing images into layers via RGB-space geometry. *ACM Transactions on Graphics (TOG)* 36, 1 (Nov. 2016), 7:1–7:14. [2](#), [3](#), [4](#), [5](#), [9](#)
- [TZN19] THIES J., ZOLLHÖFER M., NIESSNER M.: Deferred neural rendering: Image synthesis using neural textures. *ACM Trans. Graph.* 38, 4 (jul 2019). [2](#)
- [WLX19] WANG Y., LIU Y., XU K.: An improved geometric approach for palette-based image decomposition and recoloring. *Computer Graphics Forum* 38, 7 (2019), 11–22. [3](#), [5](#), [9](#), [11](#)
- [WOG06] WINNEMÖLLER H., OLSEN S. C., GOOCH B.: Real-time video abstraction. *ACM Trans. Graph.* 25, 3 (jul 2006), 1221–1226. [2](#), [3](#), [7](#)
- [XK08] XU J., KAPLAN C. S.: Artistic thresholding. In *Proceedings of the 6th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2008), NPAR '08, Association for Computing Machinery, p. 39–47. [6](#)
- [XXH*21] XIANG F., XU Z., HASAN M., HOLD-GEOFFROY Y., SUNKAVALLI K., SU H.: Neutex: Neural texture mapping for volumetric neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2021), pp. 7119–7128. [3](#)
- [YLT*21] YU A., LI R., TANCİK M., LI H., NG R., KANAZAWA A.: PlenOctrees for real-time rendering of neural radiance fields. In *ICCV* (2021). [2](#), [3](#), [4](#), [6](#), [8](#)
- [ZSD*21] ZHANG X., SRINIVASAN P. P., DENG B., DEBEVEC P., FREEMAN W. T., BARRON J. T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. *ACM Trans. Graph.* 40, 6 (dec 2021). [3](#)
- [ZXST17] ZHANG Q., XIAO C., SUN H., TANG F.: Palette-based image recoloring using color decomposition optimization. *IEEE Transactions on Image Processing* 26, 4 (2017), 1952–1964. [3](#)

A. Implementation Details of Directional Sampling

To uniformly sample view directions, we first project the hemispherical domain to the unit square using the cylindrical map $(\theta, \phi) \mapsto (\phi/2\pi, \cos\theta)$ and regularly sample directions on the square. Sampling on a square is much easier than on a sphere, and the cylindrical map ensures that each direction sample covers approximately the same area in the directional domain. We use the resolution ratio 4 : 1 for sampling points on the square to have the same aspect ratio as the original spherical coordinates. Specifically, the directional sampling count is in the form $N_{\text{dir}} = 4n^2$, and the radiance sample is generated at the center of each $4n \times n$ regular grid on the square.

We perform multisampling to generate anti-aliased radiance samples. We regularly sample 2×2 directions in each grid and compute the color and volumetric visibility. The final radiance samples passed to the subsequent filtering stage are obtained as the average of these 2×2 raw samples. The average color is computed using visibility as a weight. In our experiments, we empirically set $N_{\text{dir}} = 16$ ($n = 2$) and use the same sampling scheme for every spatial location.

B. Specific Experimental Conditions

To create Figure 8, we selected the 10, 40, 70, 100, 130, 160, and 190th camera poses in the test data of the NeRF-synthetic dataset as example views. To extract a palette from a single image, we used the 160th view for the SHIP model and the 70th view for the HOTDOG model. We set $K = 40$ for the k-means stage in the method of Chao et al. [CSG21].