DOI: 10.1111/cgf.14587
Eurographics Symposium on Rendering 2022
A. Ghosh and L.-Y. Wei
(Guest Editors)

Volume 41 (2022), Number 4

Automatic Feature Selection for Denoising Volumetric Renderings

Xianyao Zhang[†][1,2], Melvin Ott[2], Marco Manzi[2], Markus Gross[1,2] and Marios Papas[†][2]

[1]ETH Zürich, Switzerland
[2]DisneyResearch|Studios, Switzerland

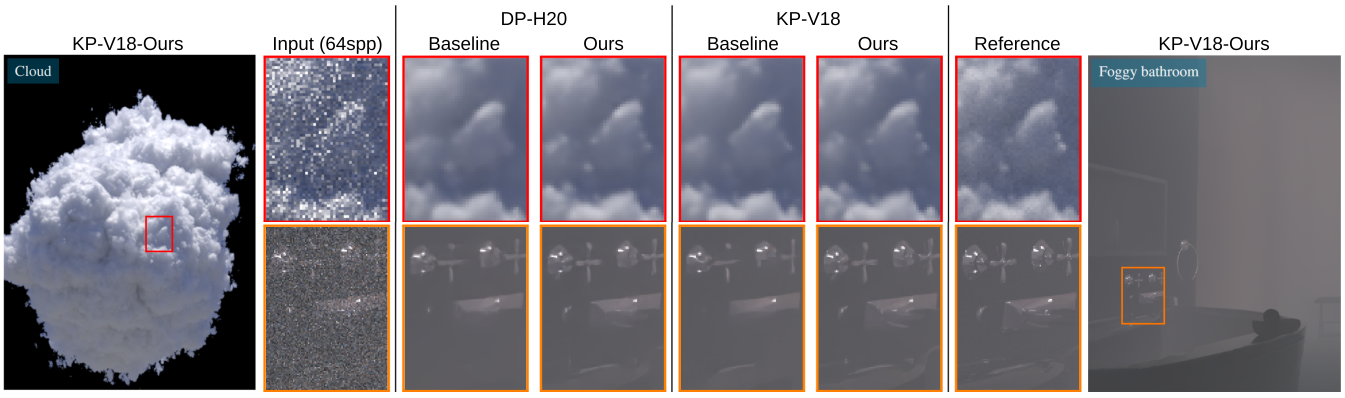


Figure 1: *We propose an automatic feature selection method to find near-optimal auxiliary feature subsets that maximize the denoising quality of neural denoisers. In this figure, we demonstrate our approach by comparing the quality obtained with our automatically selected feature set to not using any volumetric features (KP-V18 [VRM*18]), or to the hand-crafted feature sets proposed by the authors (DP-H20 [HMES20]). Compared to baseline feature sets, our selected feature sets can lead to much improved results with the same underlying neural network architecture on different types of volumetric effects. © Disney / Pixar*

Abstract

We propose a method for constructing feature sets that significantly improve the quality of neural denoisers for Monte Carlo renderings with volumetric content. Starting from a large set of hand-crafted features, we propose a feature selection process to identify significantly pruned near-optimal subsets. While a naive approach would require training and testing a separate denoiser for every possible feature combination, our selection process requires training of only a single probe denoiser *for the selection task. Moreover, our approximate solution has an asymptotic complexity that is quadratic to the number of features compared to the exponential complexity of the naive approach, while also producing near-optimal solutions. We demonstrate the usefulness of our approach on various state-of-the-art denoising methods for volumetric content. We observe improvements in denoising quality when using our automatically selected feature sets over the hand-crafted sets proposed by the original methods.*

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ ***Ray tracing; Feature selection; Neural networks;***

1. Introduction

Volumetric effects such as fog, smoke, and clouds play an important role in animated movies and visual effects. However, those are also among the computationally most expensive effects to accurately render [FWKH17]. The most widely used algorithms for production rendering rely on path tracing due to its generality and simplicity [KFF*15]. The main drawback of path tracing is its computation cost. For complex scenes—especially those containing volumetric effects—hundreds of hours of computation

[†] xianyao.zhang@inf.ethz.ch, marios.papas@disneyresearch.com

time can be required to render a single frame at final-production quality. If the rendering process is stopped prematurely, the image will contain error that manifests as undesirable noise artifacts. A commonly-used remedy for reducing computation time to reach a desired quality is denoising. Before the rendering converges to a noise-free state, the process is stopped early, and a comparatively fast denoising post-process can be applied to remove the noise in the intermediate image. State-of-the-art denoisers [VRM*18, CKS*17, XZW*19, ZMV*21, HMES20, HHCM21] are based on training deep convolutional neural networks to produce noise-free images from noisy inputs. While training denoising networks on large datasets can be very time consuming, once trained, these networks can be used to very efficiently produce clean images from noisy data.

One key aspect to denoising Monte Carlo renderings is that renderers can be instrumented to output additional information, called *auxiliary feature maps*. Passing these feature maps to the denoiser can often help preserve scene details more accurately. There are established sets of features that are commonly used to improve the denoising quality of surfaces [RMZ13, VRM*18]. However, while various auxiliary features for denoising volumetric content have been proposed in the past [HMES20, XSW*20, HHCM21], there is no established feature set for volume denoising.

Designing a set of features that yields the best denoising quality for a given state-of-the-art volume denoising method is not a trivial task. The main challenges are first proposing a set of good features with a high probability of increasing denoising quality and then identifying which feature combination will yield the highest quality. As we will later demonstrate, the contribution of each feature to the denoising quality is dependent on which other features are selected. This correlation between features makes it hard to predict the impact of each feature independently without considering possible combinations with other features. Finding the optimal feature set in a brute-force manner quickly becomes intractable even for a handful of features since a new denoiser should be trained and evaluated for every feature combination. On the other hand, using many unnecessary features can decrease the denoising quality of neural methods and significantly increase disk storage required per frame. The storage overhead is further amplified in large-scale production environments by the demand to increase frame rate, frame resolution, and multi-view rendering. In such scenarios, even a small overhead can quickly become a significant and potentially prohibitive cost factor. Additionally, using a smaller feature set reduces computation overhead slightly, which can be valuable considering the cost savings across many frames.

In this paper, we propose an algorithm to identify sets of features that significantly improve the quality of various state-of-the-art methods [VRM*18, HMES20, HHCM21] that can be used for volume denoising. We progressively expand our selection of features from a more extensive hand-crafted set to identify features that will yield the best improvement in the denoising quality. We predict the impact of each feature combination on the denoising quality by evaluating our proposed *probe denoiser* on different feature sets and measuring their quality difference. Our probe denoiser shares the same network architecture as the baseline denoiser, but we train it with random feature dropout to simulate the process of feature selection. After a once-off training, the probe denoiser becomes a capable oracle for predicting the quality benefit of different feature sets. We then propose a progressive construction process for feature selection, which incrementally builds a feature set based on predictions from the probe denoiser and runs in quadratic time with respect to the number of features. After feature selection, the denoiser can be retrained using the selected feature sets, yielding results with improved quality.

We provide feature sets obtained by our method for a selection of state-of-the-art interactive and offline denoisers for volumes [VRM*18, HMES20, HHCM21]. We demonstrate that these sets improve the denoising quality compared to the hand-selected feature sets initially proposed by the authors of the respective methods. Our evaluation reveals that the proposed selection process can account for redundancies between features, allowing for significant pruning without denoising quality degradation over the complete hand-crafted feature set. We provide a series of ablation studies analyzing the impact of our design choices on the accuracy and stability of our selection method. We further validate our findings by evaluating denoising quality improvements due to the resulting feature sets on a diverse set of volumetric scenes involving homogeneous and heterogeneous volumes in different lighting scenarios. We provide a visual comparison of the denoising quality improvements with our selected feature sets for state-of-the-art architectures in Figure 1.

In summary, our main contributions are the following:

- a specially trained probe denoiser that is capable of predicting the impact of different feature sets on the denoising quality,
- a feature selection algorithm that uses a greedy optimization and a probe denoiser to compute near-optimal sets of features in quadratic time without requiring re-training during the selection process, and
- an extensive set of hand-crafted volumetric features along with near-optimal feature sets for various state-of-the-art denoising architectures.

2. Related work

Our method builds on existing work in physically based volume rendering, Monte Carlo denoising, and feature selection, all of which are active research areas. In this section we provide a review of the most relevant literature to our work in each of these fields.

Volume rendering The prevalent algorithm to render volumetric effects for production purposes is volumetric path tracing based on radiative transfer equation [Cha13], where the light paths are scattered and absorbed inside of participating media. Rendering scenes with volumetric effects can result in a much larger number of scattering events per path than when rendering scenes with only surface interactions, leading to excessive noise or prolonged rendering time. Therefore, various attempts have been made to accelerate volumetric rendering, *e.g.*, through various types of photon mapping [BJ17], improvements of the free flight distance sampling [NSJ14], transmittance estimation [dN21], path guiding [HZE*19] or modelling higher-order scattering effect with neural networks [KMM*17]. For a thorough overview of the available volume rendering techniques, we refer the interested reader to surveys by Cerezo et al. [CPP*05] and Novák et al. [NGHJ18].

In this work, we adopt the basic volumetric path tracing algorithm with next event estimation [PJH16] because of its generality and popularity in production [FWKH17].

Monte Carlo Denoising Denoising can often reduce the render time needed to reach a desired quality by orders of magnitude [FHH*19]. We refer the readers to the survey from Zwicker et al. [ZJL*15] for an overview of traditional denoising and reconstruction methods. In particular, Rousselle et al. [RMZ13], He et al. [HST13] and Moon et al. [MJL*13] propose the use of various auxiliary feature buffers for the denoising task. Recent years saw the rise of deep learning approaches that leverage information in large datasets. Our method is specifically tailored to improve the quality of such deep-learning-based denoisers. Bako et al. [BVM*17] use a kernel-predicting convolution network (KPCN) to predict per-pixel filtering kernels from the noisy image and auxiliary feature buffers, achieving significant improvement over traditional approaches. Vogels et al. [VRM*18] extend KPCN by reusing temporal information and suggest a multi-scale reconstruction approach with U-Nets [RFB15] as the backbone. Zhang et al. [ZMV*21] further extend this approach by learning decompositions of the noisy color and per-component auxiliary features before denoising. Taking a different route, Xu et al. [XZW*19] train models with adversarial losses that directly predict the denoised pixel values (without relying on filter kernel), and Back et al. [BHHM20] combine unbiased and biased estimates for better final reconstruction quality. Finally, sample-based denoisers have been recently proposed that exploit additional information contained in individual radiance samples [GLA*19, MH20, CHY21, IFME21] for the purpose of interactive or even real-time frame rates. To the best of our knowledge, we are the first to use a systematic approach to select the best auxiliary features from a large feature set for Monte Carlo denoising, as previous work focused on manually creating good features [RMZ13, MJL*13] or improving the neural network architecture [BVM*17, ZMV*21, XZW*19].

Volume Denoising The success of the aforementioned denoising methods is demonstrated mainly in scenes with predominantly surface interactions. Recently, the task of denoising renderings with volumetric effects has also been studied. Hofmann et al. [HMES20] introduce a denoiser specialized for scenes with volumes stemming from medical data but with simplified light transport. The auxiliary features for denoising volumetric effects used in this work inspired our proposed features, and we include them in our handcrafted set. Subsequently, a pipeline for interactive volume rendering was proposed [HHCM21] which includes a lightweight neural denoiser that uses a different set of auxiliary features. In the context of gradient-domain volumetric photon density estimation, Xu et al. [XSW*20] proposed a method using auxiliary volume-specific feature buffers like transmittance and photon density to denoise global homogeneous volumes. Among the more traditional methods, Iglesias-Guitian et al. [IGMM20] propose a real-time denoising pipeline for volumetric renderings using a history buffer. Vicini et al. [VAN*19] extends the non-local means denoiser to denoise deep images, in which each pixel contains multiple bins; volumetric renderings are one common use case for deep images.

Feature selection Feature selection methods for machine learning models aim at using minimal feature sets to achieve good prediction accuracy, and the feature selection problem is considered NP-hard in general, due to the combinatorial number of potential feature sets [Wel82]. It is possible to perform feature selection without knowledge of the machine learning model used, which is often termed model-agnostic feature selection [BCSMAB15]. Minimum Redundancy and Maximum Relevance (mRMR) is a representative class of model-agnostic methods. Using a *forward selection* scheme, mRMR includes one additional feature at each step to the selected feature set when this feature provides the best relevance-redundancy balance [DP03, ZAW19]. Here, relevance means the similarity between a feature and the ground truth, and redundancy is measured by the similarity between a new feature and already selected features.

Model-aware methods are opposite to the model-agnostic methods, which incorporate the model into feature selection. Frequently appearing in these methods is the concept of Shapley value [Sha16], which represents the contribution of a feature while considering all possible combinations of other features. Cohen et al. [CRD05] propose to use Shapley values for feature selection for traditional machine learning algorithms like decision trees. Castro et al. [CGT09] propose and analyze the method to speed up Shapley values computation by sampling feature permutations, but evaluating Shapley values is still computationally expensive.

Feature selection methods have also been proposed for deep neural networks [SN20, KVSF20], and convolutional neural networks (CNNs) [AOG19, CSWJ18, YJV19]. We adopt the commonly used forward selection regime in our method, but the probe denoiser in our method bears novelty compared to previous work. First, previous approaches, when applied on convolutional neural networks, evaluate the contribution of each pixel [AOG19, CSWJ18] or patch [YJV19] to the result. Second, though related works on network pruning or feature ranking [WC20, GGZ*18] have considered the feature dropout idea, we use dropout in our probe denoiser to select entire input feature maps for a convolutional network, which is different from previous work to the best of our knowledge.

3. Background

In this section, we revisit the fundamental concepts in volumetric path tracing (Section 3.1) and Monte Carlo denoising with neural networks (Section 3.2), as preparation for describing our proposed feature selection method.

3.1. Volumetric path tracing

In this work, we follow a common parameterization of participating media to render volumetric effects [PJH16], using the spatially varying *scattering coefficient* $\sigma_s(\cdot)$ and *absorption coefficient* $\sigma_a(\cdot)$, and the *phase function* $f_p(\cdot,\cdot,\cdot)$. The coefficients describe the strength of the scattering and absorption within the volume along an infinitesimal beam, and the phase function defines the distribution of the direction of scattered light paths. Though the current work only considers scattering and absorption effects and assumes no volumetric emission, our feature selection method can be naturally extended to select emission-related features.

The incident radiance $L_i(\mathbf{x},\vec{\omega})$ at point $\mathbf{x}$ from direction $\vec{\omega}$ can be defined by the following recursive integrals, assuming that the closest surface along the ray is at point $\mathbf{x}_0 = r(\mathbf{x},\vec{\omega}) = \mathbf{x}+t_{\max}\vec{\omega}$:

$$L_i(\mathbf{x},\vec{\omega}) = \underbrace{T_r(\mathbf{x}_0 \leftrightarrow \mathbf{x})L_o(\mathbf{x}_0,-\vec{\omega})}_{L_{i,\mathrm{surf}}(\mathbf{x},\vec{\omega})} + \underbrace{\int_0^{t_{\max}} T_r(\mathbf{x}+t\vec{\omega} \leftrightarrow \mathbf{x})L_s(\mathbf{x}+t\vec{\omega},-\vec{\omega})\mathrm{d}t}_{L_{i,\mathrm{vol}}(\mathbf{x},\vec{\omega})}, \tag{1}$$

$$L_s(\mathbf{x},\vec{\omega}) = \sigma_s(\mathbf{x})\int_{S^2} f_p\left(\mathbf{x},\vec{\omega},\vec{\omega}'\right)L_i\left(\mathbf{x},\vec{\omega}'\right)\mathrm{d}\vec{\omega}', \tag{2}$$

$$L_o(\mathbf{x},\vec{\omega}) = L_e(\mathbf{x},\vec{\omega}) + \int_{H^2} f_r\left(\mathbf{x},\vec{\omega},\vec{\omega}'\right)L_i\left(\mathbf{x},\vec{\omega}'\right)\cos\theta'\mathrm{d}\vec{\omega}'. \tag{3}$$

Among the radiance terms, $L_s(\mathbf{x},\vec{\omega})$ is the in-scattered radiance, $L_o(\mathbf{x},\vec{\omega})$ is the total outgoing radiance, and $L_e(\mathbf{x},\vec{\omega})$ is the emitted radiance at location $\mathbf{x}$ in direction $\vec{\omega}$. Additionally, $T_r(\mathbf{x} \leftrightarrow \mathbf{y})$ is the *transmittance* between $\mathbf{x}$ and $\mathbf{y}$, $f_r\left(\mathbf{x},\vec{\omega},\vec{\omega}'\right)$ is the *bidirectional scattering distribution function* (BSDF) characterizing surface scattering, and S^2 and H^2 are the unit sphere and hemisphere, respectively. More specifically, transmittance is defined as:

$$T_r(\mathbf{x} \leftrightarrow \mathbf{y}) = \exp\left\{-\int_0^l \sigma_t(\mathbf{x}+t\vec{\omega}_0)\mathrm{d}t\right\}, \tag{4}$$

with $\vec{\omega}_0 = \frac{\mathbf{y}-\mathbf{x}}{\|\mathbf{y}-\mathbf{x}\|_2}$ being the unit vector pointing from $\mathbf{x}$ to $\mathbf{y}$, $l = \|\mathbf{y}-\mathbf{x}\|_2$ the distance between the two points, and $\sigma_t(\mathbf{x}) = \sigma_s(\mathbf{x}) + \sigma_a(\mathbf{x})$ the (spatially varying) *extinction coefficient*. One can also define the *optical thickness*, τ, from the exponent in the above equation, as $\tau = \int_0^l \sigma_t(\mathbf{x}+t\vec{\omega}_0)\mathrm{d}t$. Volumetric path tracing uses Monte Carlo integration to compute the recursive integral in Equation (1). Each sample from volumetric path tracing will result in a light path consisting of N points, represented by $\{\mathbf{x}^{(i)}\}_{i=0}^{N-1}$, with the virtual camera as the 0-th point $\mathbf{x}^{(0)}$. At each point $\mathbf{x}^{(i)}$ where $i > 0$, we can record additional information related to either the current point or the previous path segment $\overline{\mathbf{x}^{(i-1)}\mathbf{x}^{(i)}}$, which can be used as auxiliary denoising features. The features are collected at each path within a pixel and averaged into images that accompany the noisy color image. We refer to these auxiliary feature images by omitting scene-space position in the expression. For instance, the image corresponding to the scattering coefficient at the 3rd point, $\sigma_s(\mathbf{x}^{(3)})$, is written as $\sigma_s{}^{(3)}$.

Also, from Equation (1), we can observe that the incident radiance $L_i(\mathbf{x},\vec{\omega})$ can be decomposed into a *surface* part $L_{i,\mathrm{surf}}(\mathbf{x},\vec{\omega})$ and a *volume* part $L_{i,\mathrm{vol}}(\mathbf{x},\vec{\omega})$, depending on the type of the path's first interaction, leading to a surface–volume decomposition of the rendered image.

Our work focuses on extracting useful auxiliary features for denoising the volume part, and the surface part can be denoised using well-established features and methods [VRM*18].

The volumetric interaction $L_{i,\mathrm{vol}}(\mathbf{x},\vec{\omega})$ can be further decomposed into single scattering $L_{i,\mathrm{ss}}$ and multiple scattering $L_{i,\mathrm{ms}}$, according to the number of volume scattering interactions:

$$L_{i,\mathrm{ss}}(\mathbf{x},\vec{\omega}) = \int_0^{t_{\max}} T_r(\mathbf{x}+t\vec{\omega} \leftrightarrow \mathbf{x})L_{s,\mathrm{ss}}(\mathbf{x}+t\vec{\omega},-\vec{\omega})\mathrm{d}t, \tag{5}$$

$$L_{s,\mathrm{ss}}(\mathbf{x},\vec{\omega}) = \sigma_s(\mathbf{x})\int_{S^2} f_p\left(\mathbf{x},\vec{\omega},\vec{\omega}'\right)T_r\left(\mathbf{x}_0' \leftrightarrow \mathbf{x}\right)L_e\left(\mathbf{x}_0',-\vec{\omega}'\right)\mathrm{d}\vec{\omega}', \tag{6}$$

$$L_{i,\mathrm{ms}}(\mathbf{x},\vec{\omega}) = L_{i,\mathrm{vol}}(\mathbf{x},\vec{\omega}) - L_{i,\mathrm{ss}}(\mathbf{x},\vec{\omega}). \tag{7}$$

3.2. Neural denoising with auxiliary features

The task for a denoiser g is to predict the reference image $\bar{\mathbf{I}}$ from noisy color image $\mathbf{I}$. When processing a noisy color image $\mathbf{I}$, a denoiser g also takes auxiliary feature maps $\mathbf{F}$ for estimating the reference image $\bar{\mathbf{I}}$, *i.e.*, $\hat{\mathbf{I}} = g(\mathbf{I},\mathbf{F}) \approx \bar{\mathbf{I}}$. Neural denoisers, which are often implemented with convolution layers, are trained on a sizeable dataset $\mathcal{D}_d$ of noisy–clean data pairs $([\mathbf{I},\mathbf{F}],\bar{\mathbf{I}})$ to minimize the average denoising error $\varepsilon(\hat{\mathbf{I}},\bar{\mathbf{I}})$ between the reference image $\bar{\mathbf{I}}$ and the prediction $\hat{\mathbf{I}}$.

Using a convolutional neural network, we can directly predict the denoised image $\hat{\mathbf{I}}$ at the final convolution layer [XZW*19, HMES20], or alternatively use the kernel-predicting approach [VRM*18, ZMV*21, HHCM21]. For the latter, the network model predicts for every pixel p kernel weights w_{pq} for its neighborhood pixels $q \in \mathcal{N}(p)$. Denoting the trainable parameters in the network with θ, the value of pixel p in the predicted image $\hat{\mathbf{I}}$ is computed as

$$\hat{\mathbf{I}}_p = g_p(\mathbf{I},\mathbf{F};\theta) = \sum_{q\in\mathcal{N}(p)} w_{pq}(\mathbf{I},\mathbf{F};\theta)\,\mathbf{I}_q.$$

Our work focuses on identifying auxiliary features sets for volume denoising rather than suitable volume denoising network architectures. Therefore, we examine the effect of our proposed feature set and the feature selection method on existing direct-predicting [HMES20] and kernel-predicting [HHCM21, VRM*18] denoisers. Note that two of the three denoiser architectures [HMES20, HHCM21] are proposed in the context of volume denoising, with their own sets of volumetric features. Our comparisons with these baselines show that with our method we can identify feature sets that achieve significant denoising quality improvements compared to the author-proposed features.

4. Volume-related features

Our goal is to identify a set of features that will improve the denoising quality of state-of-the-art methods on volumes. To arrive at a concise and beneficial feature set, we first hand-craft a large set of features and then select feature subsets from those programmatically; the following two sections will discuss these two steps.

Our criteria for producing this hand-crafted set are the following: we select features that can be extracted without significant computation overhead during volumetric path tracing, exhibit structures that correlate with the clean color image, and have lower noise levels than the corresponding noisy color image. Table 1 lists the complete set of 29 volumetric features.

To begin with, we select two features that correspond to path-space decompositions of volumetric light transport. The single-scattering feature ($L_{i,\mathrm{ss}}$, Equation (5)) corresponds to illumination

Symbol	Feature Name	Range	Transform
$L_{i,\text{ss}}$	Single scattering	$\mathbb{R}_+$	$\log(\lvert x\rvert+1)$
$L_{i,\text{ms}}$	Multiple scattering	$\mathbb{R}_+$	$\log(\lvert x\rvert+1)$
ρ_{vol}	Volume albedo	$[0,1]$	Equation 10 from [FPWW89]
σ_s	Scattering coefficient	$\mathbb{R}_+$	$\text{sign}(x-\mu)\log(\frac{\lvert x-\mu\rvert}{\sigma}+1)$
T_r	Transmittance	$[0,1]$	x
z_{v}	Volume interaction length	$\mathbb{R}_+$	$\frac{(x-\mu)}{\sigma}$
$\mathbf{v}$	Velocity direction	$[-1,1]$	x
$\lVert\hat{\mathbf{v}}\rVert$	Velocity magnitude	$\mathbb{R}_+$	$\text{sign}(x-\mu)\log(\frac{\lvert x-\mu\rvert}{\sigma}+1)$
$\mathbf{n}_{\text{vol}}$	Density gradient direction	$[-1,1]$	x
$\lVert\hat{\mathbf{n}}_{\text{vol}}\rVert$	Density gradient magnitude	$\mathbb{R}_+$	$\text{sign}(x-\mu)\log(\frac{\lvert x-\mu\rvert}{\sigma}+1)$
$\mathbf{p}$	Object-space position	$\mathbb{R}$	x
z_{cam}	Distance to camera	$\mathbb{R}_+$	x
τ	Optical thickness	$\mathbb{R}_+$	x
r_{sct}	Scattering ratio	$[0,1]$	x
σ_t	Extinction coefficient	$\mathbb{R}_+$	x

Table 1: *Volumetric features used in this paper, following notation in Section 3.1. We record all features at the first and second volume interactions, except for $L_{i,\text{ss}}$, $L_{i,\text{ms}}$, and z_{cam}, which only applies at the first interaction location. Denoising networks receive transformed features according to the rightmost column, where μ and σ denote a feature's mean and standard deviation over the training set, respectively. We set disabled features to a default value of 0, except for transmittance which we set to 1. For completeness, we include the estimated standard deviation for first-bounce σ_t and z_{cam}, as proposed by Hofmann et al. [HHCM21], but do not include variance or standard deviation for other features.*

that experienced exactly one scattering interaction with the volume(s) directly visible at each pixel. The multiple-scattering feature ($L_{i,\text{ms}}$, Equation (7)) corresponds to illumination that experiences more than one scattering interaction with the volume(s).

The remaining 27 features correspond to quantities evaluated during volumetric path tracing (see Section 3.1), and are collected both at the first and second volumetric interactions unless mentioned otherwise. Some of the features are inspired by previous volume denoising works [HMES20, HHCM21]. We include the scatter position ($\mathbf{p}$), scattering albedo ($\rho_{\text{vol}} = \sigma_s/\sigma_t$), scattering coefficient (σ_s), and extinction coefficient (σ_t).We also consider the length of the path segment prior to the interaction, starting either from the camera (z_{cam}, only at the first interaction) or from the volume bounding shape (z_{v}, both first and second interactions). Next, the estimated transmittance (T_r) and optical thickness (τ) are obtained along the path segments, and we record the scattering ratio (r_{sct}), which represents the ratio of scattering interactions over the total number of paths in the pixel. In heterogeneous volumes, we record the volume density gradient direction ($\mathbf{n}_{\text{vol}}$) and magnitude ($\lVert\hat{\mathbf{n}}_{\text{vol}}\rVert$) akin to surface normals. Finally, when available, the velocity direction ($\mathbf{v}$) and magnitude ($\lVert\hat{\mathbf{v}}\rVert$) of the underlying physical simulation are also extracted as features. Hofmann et al. [HHCM21] include the estimated standard deviation of $z_{\text{cam}}^{(1)}$ and $\sigma_s{}^{(1)}$ as features, and we follow this approach to include these two features for a fair comparison. We do not include the variance or standard deviation estimates for other features.

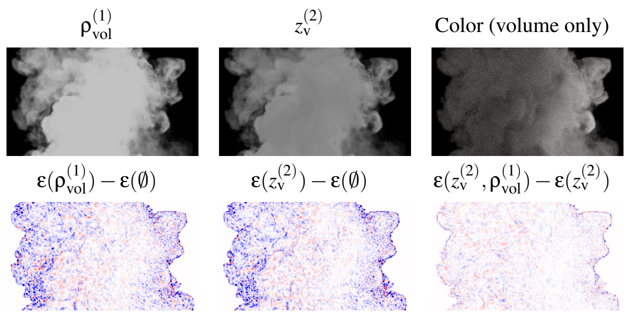

$\rho_{\text{vol}}^{(1)}$
$z_{\text{v}}^{(2)}$
Color (volume only)
$\varepsilon(\rho_{\text{vol}}^{(1)}) - \varepsilon(\emptyset)$
$\varepsilon(z_{\text{v}}^{(2)}) - \varepsilon(\emptyset)$
$\varepsilon(z_{\text{v}}^{(2)}, \rho_{\text{vol}}^{(1)}) - \varepsilon(z_{\text{v}}^{(2)})$

Figure 2: *Top row shows the 1st volume albedo $\rho_{\text{vol}}^{(1)}$ and 2nd volume interaction length $z_{\text{v}}^{(2)}$. The bottom row shows the **change** in denoising error ε when adding a new feature compared not using that feature. Here ε = SMAPE and increases/decreases in error are shown in red/blue. Note how both features affect ε similarly when used in isolation compared to not using any feature. Due to their strong correlation however, using both features improves ε only marginally compared to using a single feature.*

Each of these features can potentially improve the quality of a denoiser. In the next section, we will propose a rigorous method for evaluating their impact, especially in combination with other features. Note that our feature selection method is independent of the starting feature set, and it can be applied to different features than the 29 mentioned above.

5. Feature selection

Using a large set of features can improve denoising quality over not using any features, but it also incurs additional storage cost and data loading overhead. As we will later demonstrate, some of the features proposed in Section 4 are found to be redundant or offer little additional useful information when combined with other features, for which Figure 2 shows an example. This leads to diminishing returns when adding more features as input to a neural denoiser, and the benefit of using multiple features is not a simple sum of the benefit of using each individual feature. In fact, we demonstrate that a relatively small subset of the proposed feature set is sufficient to reap most of the benefits in terms of denoising quality. It is, however, challenging to identify such a subset because of the difficulty to predict the effect of various input feature combinations on denoising quality.

Given a set of features $\mathbf{P}$, we seek the subset $\mathbf{S}^\star \subseteq \mathbf{P}$ that leads to the best denoising quality. We also discuss alternative ways to formulate the problem as tradeoff between denoising quality and number of features in Section 5.3. Without any assumptions about the features nor a good oracle for predicting the quality improvement due to different feature sets, we would have to resort to a brute-force solution of training multiple denoisers with different feature sets $\mathbf{S} \subseteq \mathbf{P}$ as input and pick the feature set leading to a denoiser with lowest average denoising error on a large *selection dataset* $\mathcal{D}_s$ with denoising examples (Section 6.1). We denote a denoiser trained with feature set $\mathbf{S}$ by $g_{\mathbf{S}}$. This solution quickly becomes infeasible even for relatively small feature sets, as the number of subsets to consider—and thus denoisers to train—is $2^{|\mathbf{P}|}$.

We propose an efficient solution with low approximation error, requiring training only one denoiser for feature selection. We use this *probe denoiser* as an oracle for predicting the impact of feature combinations on the final denoising error. Additionally, we propose a greedy feature set selection algorithm that only requires testing at most $O(|\mathbf{P}|^2)$ subsets with the probe denoiser, avoiding the combinatorial complexity at the cost of some approximation error. Our method is performed in four steps:

1. Train a probe denoiser $\tilde{g}$ with the desired network architecture to measure the quality of different combinations of feature sets $\mathbf{S} \subseteq \mathbf{P}$ without retraining (Section 5.1).
2. Construct a sequence of *candidate feature sets* $\bar{\mathbf{P}} = \{\mathbf{S}_i : i = 0, 1, \ldots, |\mathbf{P}|\}$ with $\mathbf{S}_0 \subset \mathbf{S}_1 \subset \cdots \subset \mathbf{S}_{|\mathbf{P}|} = \mathbf{P}$ using a forward selection algorithm, where feature sets of progressively increasing sizes are evaluated with the probe denoiser $\tilde{g}$ (Section 5.2).
3. Select the *best performing feature set* $\tilde{\mathbf{S}}^\star \subseteq \bar{\mathbf{P}}$ that yields either the best denoising error or the desired trade-off between denoising quality and feature set size (Section 5.3).
4. Train a specialized denoiser $g_{\tilde{\mathbf{S}}^\star}$ from scratch with the resulting feature set $\tilde{\mathbf{S}}^\star$.

5.1. Probe denoiser training

Identifying good feature sets for denoising relies on an efficient evaluation of the denoising quality impact between different feature subsets $\mathbf{S} \subseteq \mathbf{P}$. As mentioned earlier, training specialized denoisers for different feature sets quickly becomes impractical, and it is thus desirable to evaluate different feature subsets with the same trained denoiser. It is possible to evaluate the impact on the denoising error of a feature set $\mathbf{S}$ using a denoiser trained with the full feature set $\mathbf{P}$, *i.e.*, $g_{\mathbf{P}}$, and manually turning off other features $q \in \mathbf{P}\backslash\mathbf{S}$ during inference. We turn off a feature by setting it to its default value—a constant feature map containing 0 or 1 (see Table 1 caption). With a slight abuse of notation, we write this evaluation as $g_{\mathbf{P}}(\mathbf{I}, \mathbf{S})$. However, inputs $[\mathbf{I}, \mathbf{S}]$ with smaller feature subsets will inevitably be out-of-distribution for $g_{\mathbf{P}}$. This undermines the reliability of the measured denoising error between different feature subsets.

To mitigate the out-of-distribution issue, we use *random feature dropout* during the training of a denoiser that we will use for feature-set probing. For each training example, each feature $q \in \mathbf{P}$ is disabled independently with a predefined probability. We use a probability of 50% in our experiments such that during training each feature subset is chosen with equal probability. We refer to a denoiser trained in this fashion as a probe denoiser. The trained probe denoiser can more reliably predict the denoising quality obtained by using smaller feature subsets as we train it on feature sets with missing features.

Despite the difference in how a probe denoiser is trained compared to a regular denoiser, we later demonstrate (Figure 4) that its denoising error for a specific feature set correlates quite well with the denoising error of a specialized denoiser trained only on that set. On the other hand, the error of a denoiser trained without dropout does not have a good correlation with the final denoising error, especially on smaller feature subsets.

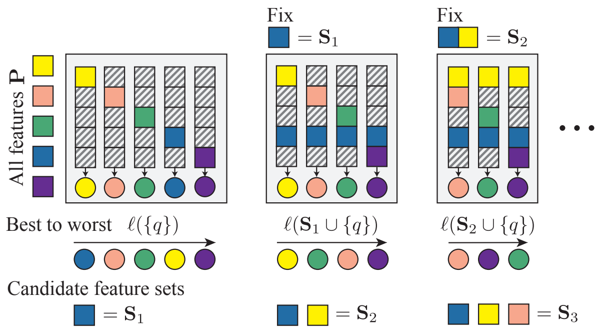

Fix
$= \mathbf{S}_1$
Fix
$= \mathbf{S}_2$
All features $\mathbf{P}$
Best to worst $\ell(\{q\})$
$\ell(\mathbf{S}_1 \cup \{q\})$
$\ell(\mathbf{S}_2 \cup \{q\})$
Candidate feature sets
$= \mathbf{S}_1$
$= \mathbf{S}_2$
$= \mathbf{S}_3$

Figure 3: *A schematic view of our progressive construction proposed in Section 5.2. We show the construction of $\mathbf{S}_1$, $\mathbf{S}_2$ and $\mathbf{S}_3$ from a set $\mathbf{P}$ with $|\mathbf{P}| = 5$. Each feature is depicted by a uniquely colored square. Slanted squares depict disabled features and circles depict the losses $\ell(\mathbf{S}_i \cup \{q\})$ (Equation (9)) with $q \in \mathbf{P}\backslash\mathbf{S}_i$.*

5.2. Candidate feature set construction

To enable early termination and facilitate tradeoffs between quality and feature set size, we progressively build a set $\bar{\mathbf{P}} = \{\mathbf{S}_i : i = 0, 1, \ldots, |\mathbf{P}|, |\mathbf{S}_i| = i\}$ by identifying *candidate feature sets* $\mathbf{S}_i$ of different sizes. The candidate feature set of size i yields the minimum average denoising error ℓ of the probe denoiser $\tilde{g}$ among all feature sets of size i, over the selection dataset $\mathcal{D}_s$, formally

$$\mathbf{S}_i = \operatorname*{argmin}_{\mathbf{S}} \ell(\mathbf{S}) \quad \text{subject to} \quad \mathbf{S} \subseteq \mathbf{P} \quad \text{and} \quad |\mathbf{S}| = i, \tag{8}$$

where

$$\ell(\mathbf{S}) := \sum_{k \in \mathcal{D}_s} \varepsilon\left(\tilde{g}\left(\mathbf{I}^k, \mathbf{S}^k\right), \bar{\mathbf{I}}^k\right) / |\mathcal{D}_s|. \tag{9}$$

Here, $\mathbf{I}^k$, $\mathbf{S}^k$ and $\bar{\mathbf{I}}^k$ are the noisy color, auxiliary features and reference color of an example in $\mathcal{D}_s$, respectively, and $\varepsilon(\cdot,\cdot)$ is the error between the denoised image and reference image according to some error metric, *e.g.*, SMAPE [BVM*17] or DSSIM which we define as (1-SSIM) [WBSS04].

Although we can make use of a probe denoiser, finding the global optimum of the optimization problem in Equation (8) requires evaluating the loss ℓ on all subsets, making it intractable even for moderately sized feature sets. For example, given a probe denoiser with an inference time of 230ms, a set of 29 features, and a selection set with 93 denoising examples, we would need approximately 11 million evaluations or 30 days to find the set of size 5 with the lowest error. Feature sets with up to 6 features would need about 3 months, and sets up to 9 features would need about a year to finish.

Therefore, we propose a greedy solution to construct the elements of $\bar{\mathbf{P}}$ in an incremental fashion, such that $\emptyset = \mathbf{S}_0 \subset \mathbf{S}_1 \subset \ldots \subset \mathbf{S}_{|\mathbf{P}|} = \mathbf{P}$.

Starting from an empty feature set, we progressively construct a

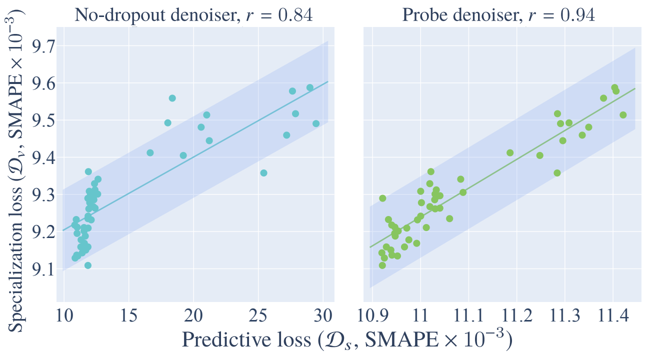


Figure 4: *We show a scatter plot of the predictive loss (x-axis) against the final specialization loss (y-axis) for a no-dropout denoiser (left) and our probe denoiser (right). We evaluate these losses on various feature subsets with different sizes, and the same feature sets are evaluated for both denoisers. Around the linear regression trend line, we indicate the measured standard deviation of the retraining noise of KP-H21 with a shaded area. Despite the noise due to retraining, we measure a higher correlation between the predictive loss and specialization loss for the probe denoiser ($r = 0.94$) over the no-dropout denoiser ($r = 0.84$). Moreover, we observe abnormally high predictive loss ($15-30$) for the no-dropout denoiser when evaluated on smaller feature sets since it was never trained with missing features.*

set of features by always selecting the one feature that improves denoising quality the most, as measured by the probe denoiser, when adding it to the set of already selected features. A more formal description of our feature set construction is given below:

1. Start from an empty set of selected features $\mathbf{S}_0 = \emptyset$ and set $i = 0$.
2. For all *remaining* features $q \in \mathbf{P} \backslash \mathbf{S}_i$ compute $\ell(\mathbf{S}_i \cup \{q\})$ according to Equation (9).
3. Set $\mathbf{S}_{i+1} = \mathbf{S}_i \cup \{q_i^*\}$ where $q_i^* = \operatorname{argmin}_q \ell(\mathbf{S}_i \cup \{q\})$, increment i and repeat step 2 and 3 until $\mathbf{S}_{i+1} = \mathbf{P}$.

We provide a visual description in Figure 3 and pseudo code in Appendix A. To facilitate the final feature set selection in Section 5.3, we store the average denoising error of each candidate set $\ell(\mathcal{D}_s, \mathbf{S}_i)$. Note, that each added feature is always selected based on its added benefit on top of the already selected ones. Thus, by construction, the resulting candidate feature sets avoid containing redundant features. We note that our method falls within the category of forward selection methods in feature selection literature [BCSMAB15]. Moreover, when experimenting on a smaller set of 18 features, our greedy selection method was able to precisely match the results of the brute force approach for up to 5 features, reinforcing our belief that our approximate method can produce near-optimal feature sets for a given probe denoiser. Our construction approach requires evaluating Equation (9) at most $\sum_{i=0}^{N-1}(|\mathbf{P}|-i)$ times, making this algorithm of quadratic complexity in the number of features.

5.3. Final selection and denoiser specialization

Once the candidate feature sets for each size are established, we can select the feature set achieving a desired tradeoff between cost and quality. We define this tradeoff with two constraining parameters: the maximum number of affordable features $1 \leq N_{\max} \leq |\mathbf{P}|$, and the minimal acceptable denoising quality gain ξ, relative to the gain using the best performing candidate set compared to using no features. Given these constraints, we can use the following simple procedure to find the *approximate optimal feature set* $\tilde{\mathbf{S}}^{\star}$. Starting from $i = 0$ we check if the relative quality gain is over the user-defined threshold

$$\frac{\ell(\emptyset) - \ell(\mathbf{S}_i)}{\ell(\emptyset) - \min_k \ell(\mathbf{S}_k)} \geq \xi. \tag{10}$$

where ℓ is the average denoising error defined in Equation (9). We perform this check by gradually incrementing i, until either we reached the desired number of features $i = N_{\max}$ (early termination) or when Equation (10) is satisfied; then we stop and report the approximate optimal feature set $\tilde{\mathbf{S}}^{\star} = \mathbf{S}_i$. We visualize the results of our selection method in Figure 5 and we discuss them in Section 7.3. Note that this selection procedure has almost zero overhead, because all involved loss quantities are already computed and stored when constructing the candidate feature sets, as mentioned in the previous subsection.

After carrying out the selection, we need to make use of the selected feature set $\tilde{\mathbf{S}}^{\star}$. The simplest option is to directly use the probe denoiser $\tilde{g}$ without modification with feature set $\tilde{\mathbf{S}}^{\star}$. We can do this since the probe denoiser is expected to yield reasonable results for the feature set $\tilde{\mathbf{S}}^{\star}$. However, in our experiments we observed that we can achieve better denoising quality by specializing a denoiser for $\tilde{\mathbf{S}}^{\star}$ instead of using the general probe denoiser that works well with any feature set. That is, we propose retraining a specialized denoiser on $\tilde{\mathbf{S}}^{\star}$, without feature dropout. This denoiser can then be used to denoise images that are accompanied with the selected set of features and the other features no longer need to be generated.

6. Experiment Setup

6.1. Data

We generate our datasets using the `volpath` integrator of the Mitsuba [Jak10] renderer for both our input and reference images. The volumetric effects present in our datasets include homogeneous media, smoke plumes, and clouds. To introduce cases where surface and geometric effects coexist, we augmented 20 publicly available scenes with effects from a set of 300 volumetric assets. To focus the experiments on volume denoising, we decompose the volume and the surface parts according to Equation (1), and apply our methods only to the volume part.

Training, selection, and validation sets. We use three disjoint datasets from the same random scene generator to evaluate our feature selection pipeline. First, all denoising networks, including the probe denoiser and specialized denoisers, are trained on an extensive training set ($\mathcal{D}_d$). After the probe denoiser training is finished, the feature selection method is carried out on a selection set ($\mathcal{D}_s$). Finally, a validation set ($\mathcal{D}_v$) is used to assess the denoising quality of the specialized denoisers after retraining.

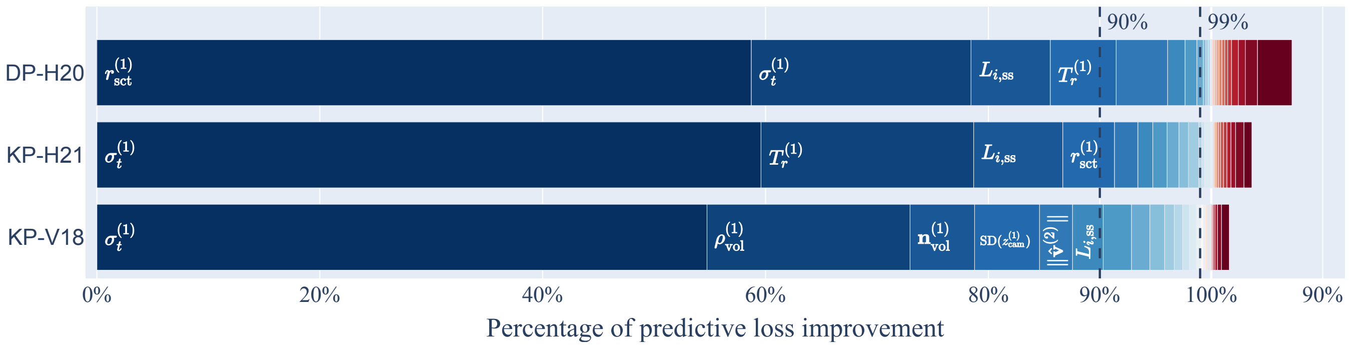


Figure 5: *For the 29-feature full set* ($|\mathbf{P}| = 29$) *and the three denoiser architectures, we show the percentage of predictive loss improvement (left-hand side of Equation (10)) for candidate feature sets* $\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_{|\mathbf{P}|}$. *We also display the feature names that are predicted to provide 90% of the relative improvement for each of the architectures. The candidate feature sets are constructed in the order from left to right* (i.e., *darkest blue denotes the first feature and darkest red denotes the last). To the right of the* 100% *mark, adding more features shrinks the predicted benefit. For all three architectures, the predicted benefit of using more features diminishes quickly with the size of the feature set, and eventually, adding more features decreases the denoising quality.*

In order to create diverse lighting effects, we randomize various scene parameters. For volumes, we randomly sample their properties such as the extinction coefficient (σ_t), albedo (ρ_{vol}), and mean cosine of the Henyey-Greenstein phase function (g_{HG}) [HG41]. Additionally, we apply random translations and rotations to heterogeneous volumes and scale their density according to a random factor. We render noisy input images and feature maps with 16, 64, and 256 samples per pixel (spp) for each scene, and we render our reference images with 16384 samples per pixel. In total, we generate 711 randomized scenes, from which 650 are used for training, 31 for selection, and 30 for validation. Note that the distribution of different volumetric effects (homogeneous, smoke plumes, clouds) is similar in these three datasets.

Test set. Eventually, the specialized denoisers will be applied to volumetric renderings not originating from our random scene generator. To simulate this scenario, we design a test set $(\mathcal{D}_t)$ containing 12 handcrafted scenes, rendered at the same sampling levels as the training set but with more than 100000 samples per pixel for the references to ensure the reliability of our metrics. These scenes cover a wide variety of volumetric effects and are closer to realistic authoring scenarios than the randomly generated scenes.

6.2. Denoiser models

To validate the generality of our method, we implement three neural denoisers from the literature and run our feature selection pipeline on each of them. We denote the three denoisers as KP-V18 [VRM*18], DP-H20 [HMES20], and KP-H21 [HHCM21], respectively. Below we briefly introduce the different denoiser architectures.

First, KP-V18 is a single-frame variant of the improved kernel-predicting convolutional network (KPCN) by Vogels et al. [VRM*18]. We follow the authors' suggestion to use a U-Net as the backbone. At each of the 5 U-Net scales, we use two residual blocks [HZRS16] on both encoder and decoder sides. The decoder of the U-Net predicts 5×5 kernels that will be applied to the noisy input at each scale, resulting in an effective kernel size of about 80 pixels. We combine the reconstructions between two scales using a multi-scale combiner [VRM*18]. In total, this model has about 16.8M trainable parameters.

Related to KP-V18 is the KP-H21 denoiser, a lightweight kernel-predicting architecture with only 433k trainable parameters, proposed in an interactive frame rate volume rendering pipeline [HHCM21]. It also consists of a U-Net, with fewer layers for the encoder than for the decoder. Though the U-Net has 6 scales itself, reconstruction kernels (also 5×5) are only predicted at the finest three scales, and they are combined similarly to KP-V18.

Finally, DP-H20 is a direct-predicting network proposed to denoise medical volumetric data [HMES20]. Instead of using one U-Net to receive all input features, the authors propose a dual-U-Net architecture, where the first- and second-bounce features are fed into the first and second U-Net respectively, with the second U-Net also receiving the output of the first. The exact backbone U-Net for DP-H20 is similar to that for KP-H21, with 6 scales, but more trainable parameters are used for DP-H20 since it does not have a frame rate constraint. This model has 2.22M trainable parameters in total.

6.3. Training scheme

We use the same training scheme for all three architectures, including losses, learning rate, number of iterations, and other hyperparameters, to focus on the feature selection pipeline instead of different training schemes for different denoisers. More specifically, our training loss is the symmetric mean sbsolute percentage error (SMAPE) [VRM*18] due to its stable behavior with high-dynamic-range (HDR) images. The loss at pixel p is defined as:

$$\varepsilon(\bar{\mathbf{I}}_p, \hat{\mathbf{I}}_p) = \frac{|\bar{\mathbf{I}}_p - \hat{\mathbf{I}}_p|}{|\bar{\mathbf{I}}_p| + |\hat{\mathbf{I}}_p| + c}, \qquad c = 10^{-2}, \tag{11}$$

and we can produce the SMAPE for an image by averaging across all image pixels and color channels.

The denoisers are implemented in TensorFlow [AAB*15]. We use the Adam [KB14] optimizer with a base learning rate of 0.0001 and set all other optimizer parameters to their default values. During training, each batch consists of 12 patches of size 128 × 128 from the training set images. The models are trained until 1.8M iterations, during which the learning rate is decayed by a factor of 10 at 1.5M and 1.65M iterations.

Before passing the features to the network, we sometimes apply a transform (see Table 1) to control the features' range. For the special case of albedo (ρ_{vol}), we found that converting it to reflectance with a simple analytic model [FPWW89] improves our results by providing increased resolution of albedo values close to 1. Even when a feature is turned off, we apply the transform on its default value before forwarding it to the network. We only use one empirically designed transform for each feature and leave the study of identifying useful transforms to future work.

6.4. Complete pipeline

In this section, we provide details of the complete pipeline of our method. All experiments were performed on a machine with an NVIDIA RTX 2080Ti (11GB) and an Intel Xeon E5-2630v4 CPU (64GB) limited to 8 to 16 cores. Though most of the computation is performed on the GPU, the CPU cores are responsible for the data loading pipeline, which becomes a bottleneck with the full feature set **P**.

We first train the probe denoiser on all features (**P**) where feature dropout with probability 0.5 is applied on the volume features. The approximate training time until 1.8M steps for the three denoisers is 2 days (KP-H21), 3 days (DP-H20), and 5 days (KP-V18).

We proceed to run the candidate feature set construction algorithm described in Section 5.2, with SMAPE as our error metric on the set **P** with all 29 features (Table 1) and our selection set with 93 images from 31 scenes with 3 different sample count levels each. The runtime of our progressive selection method also depends on the inference time of the probe denoiser. Specifically for computing all candidate feature sets it requires approximately 2.5 hours for KP-H21, 3 hours for DP-H20 and about 4 hours for KP-V18. Given the resulting sequence of candidate feature sets ($\bar{\mathbf{P}}$) we select the approximate optimal feature sets $\tilde{\mathbf{S}}^{\star}$ with three values for quality threshold ξ (Equation (10)), namely $\xi \in \{90\%, 99\%, 100\%\}$. Finally, specialized denoisers are trained without feature dropout on the respective selected feature sets $\tilde{\mathbf{S}}^{\star}$, following the same training schedule as the probe denoisers and taking a similar amount of time as data loading is the bottleneck.

7. Results

In this section, we show the experimental justification of our proposed method. We start with an evaluation of the probe denoiser, showing its ability to predict the quality of feature sets reliably (Section 7.1). Then we proceed to demonstrate that, with the help of the probe denoiser, our progressive construction algorithm can result in superior denoising quality compared to a model-agnostic feature selection method that does not consider the denoiser (Section 7.2). Finally, we provide qualitative and quantitative comparisons to demonstrate the effectiveness of our proposed volume-

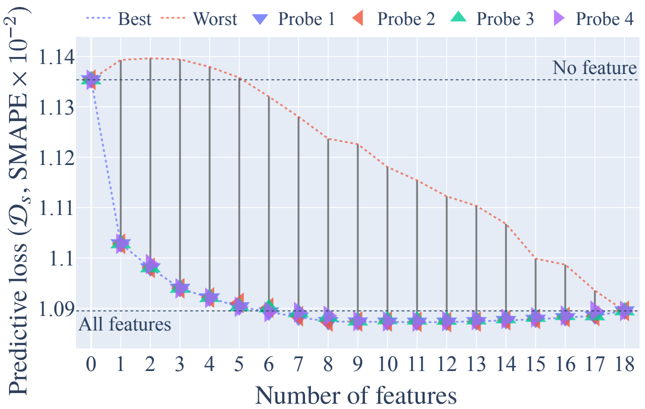


Figure 6: *We evaluate the stability of our probe denoiser training and feature selection methods by repeating the probe denoiser training and feature selection four times with different seeds. We evaluate the predictive loss of candidate feature sets produced by all differently trained probe denoisers on one of them (differently oriented triangles). The best feature sets reported by this probe denoiser is connected by the blue dashed line, and we also run a greedy search for the worst feature sets, shown as the red dashed line. The distance between them (gray lines) thus indicate the range of the performance of possible feature sets at each size. Even though the selected sets are not identical across different probe denoisers, they lead to similar predictive losses during selection, indicating the stability of our method under repetitions. See Section 7.1.2 for a detailed discussion.*

related feature sets on the denoising task (Section 7.3) for various state-of-the-art denoising architectures.

7.1. Probe denoiser evaluation

We evaluate the predictive power and stability of our probe denoiser as described in Section 5.1 with two experiments.

7.1.1. Effect of feature dropout

For a denoiser to be used as an oracle to predict the quality of different feature sets, its predictive loss (*i.e.*, Equation (9)) on the selection set $\mathcal{D}_s$ should highly correlate with the specialization loss on the validation set $\mathcal{D}_v$ after retraining a denoiser using the selected feature sets. We retrain multiple times specialized denoisers (KP-H21) for these experiments on various subsets from a set with 18 features. We intentionally use a smaller feature set with 18 features and a denoiser that trains quickly to allow more training runs with the same compute budget. To validate that our probe denoiser is a suitable oracle, we measure the correlation between the specialization loss and the probe denoiser's predictive loss. We also evaluate the predictive loss of the same feature subsets with a denoiser trained with all 18 features always present (*no-dropout*).

In Figure 4, we display a scatter plot of the predictive loss (x-axis) against the final specialization loss (y-axis) for a denoiser that does not use feature dropout during training (left) and our probe denoiser (right). We evaluate these losses on the same feature sets for both denoisers. Despite the noise due to retraining which we indicate by a shaded area, we can measure a higher correlation between the predictive loss and specialization loss for the probe denoiser ($r = 0.94$) over the no-dropout denoiser ($r = 0.84$). Also, note how the no-dropout denoiser reports abnormally high predictive loss values ($15 - 30$) on small feature sets, validating the presence of the out-of-distribution problem, which motivated our need for the probe denoiser (Section 5.1). These findings support our decision to select feature sets based on their performance on a probe denoiser.

7.1.2. Probe denoiser and feature selection stability

In Figure 6 we demonstrate that the probe denoisers are relatively stable to random fluctuation that can happen when training multiple sibling runs, *i.e.*, runs that only differ in the random seed. More specifically, we train four sibling runs for the KP-H21 probe denoiser with a total of 18 features and construct candidate feature sets for all sizes and probe denoisers using our feature selection process. Even though their candidate feature sets can disagree, we show that these feature sets yield similar predictive losses and are equally good candidates for specialization.

We evaluate the predictive loss of candidate feature sets produced by all different probe denoiser siblings on one of them to demonstrate the stability of the process. To provide context with the range of the predictive loss of different feature sets, we use a probe denoiser to greedily select the *worst* feature sets of each size and mark the range from the best to worst predictive losses with a vertical black bar. Even though different probe denoisers can construct different candidate feature sets, they yield similar predictive losses at each size. This result indicates that the probe denoiser's behavior is stable under repetitions.

7.2. Comparing candidate feature set construction methods

We compare our candidate feature set construction method, as described in Section 5.2, with other feature set construction methods.

The first method is minimum redundancy maximum relevance (mRMR) [DP03], which determines the usefulness of features by examining the input and reference images without any knowledge of the denoiser. MRMR ranks the feature according to redundancy and relevance metrics, minimizing the former and maximizing the latter. We use the linear correlation between a feature and the reference as the relevance metric and the average linear correlation between a feature and already selected features as the redundancy metric. Similar to our average denoising metrics, we compute the correlation first for each image over all pixels and then average across all images in the selection set $\mathcal{D}_s$. The second is independent construction, which only considers single features' effects without any combinations. It essentially ranks the single features from best to worst and constructs candidate feature sets of size i using the features ranked top i in isolation. The third is a variation of our method used in Section 7.1.1 that the same feature selection algorithm but instead of a probe denoiser, it uses a denoiser trained without feature dropout. The fourth alternative method is performing a brute-force search that evaluates all possible feature sets using our probe denoiser. Due to this method's excessive computation time requirements, we were only able to compute candidate feature sets with up to 5 features.

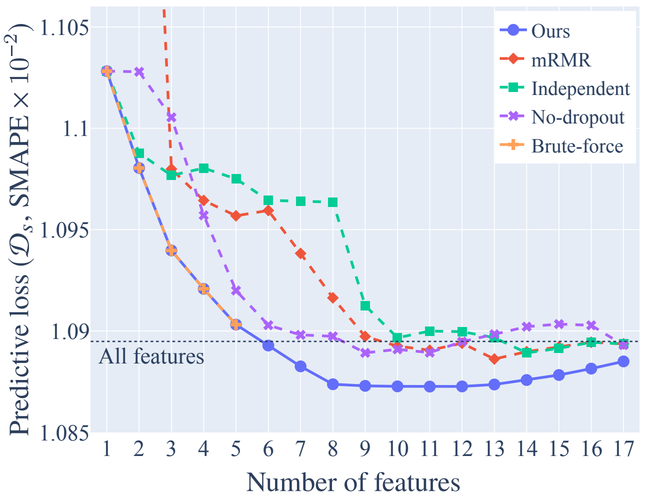


Figure 7: *We show the predictive loss of their candidate feature sets on a probe denoiser for various feature set construction methods. The no-dropout construction method (purple) uses our progressive construction algorithm on a denoiser trained without feature dropout. The independent construction method (green) uses a probe denoiser but does not consider feature interactions. Both methods lead to suboptimal feature sets. The mRMR (red) algorithm only examines the input and reference data and starts with a choice of features similar to the reference, but they do not improve denoising. Our construction method (blue), which uses the probe denoiser and a progressive construction algorithm, results in the lowest predictive error on $\mathcal{D}_s$ for all feature set sizes and matches the brute force approach (orange) up to 5 features.*

From a set of 18 features, we collect the candidate feature sets created by these methods for the KP-H21 architecture and evaluate the predictive loss of a probe denoiser on them. We plot these results in Figure 7. Ideally, we would perform these comparisons on specialized denoisers trained with each method's resulting candidate feature sets. As also observed in Figure 4, there is significant noise when retraining specialized denoisers. To mitigate this noise, we would need to retrain many training runs and average their denoising error. Instead, based on our observations in Section 7.1 regarding the predictive power and stability of our probe denoiser, we use that as our oracle to predict the final quality of feature sets resulting from each method.

We observe a perfect match between our greedy method (blue) and the brute force (orange) for the first five features. Our method could compute all candidate feature sets in 50 minutes, whereas the brute force method required 2.6 days for the first 5 features and would require 65 days for all 18 features. Note the diminishing returns as the selected feature set size increases, which indicates even if the greedy approach not able to find the exact globally optimal feature set after 5 features, we do not expect it to have a dramatic impact in denoising quality.

For the other methods, mRMR (red) starts with a terrible selec-

]

			Metrics ($\times 10^{-3}$)			
			SMAPE		DSSIM	
Denoiser	Features	$\lvert\mathbf{S}\rvert$	$\mathcal{D}_v$	$\mathcal{D}_t$	$\mathcal{D}_v$	$\mathcal{D}_t$
KP-V18	Baseline	0	9.09	8.07	29.6	11.8
	Ours (90%)	6	8.51	6.64	27.3	7.46
	Ours (99%)	14	**8.47**	6.56	**27.1**	7.13
	Ours (100%)	22	8.48	6.62	**27.1**	7.19
	All	29	**8.47**	**6.55**	**27.1**	**7.08**
DP-H20	Baseline	10	9.21	7.60	29.7	9.68
	Ours (90%)	4	8.97	7.34	29.2	9.15
	Ours (99%)	8	**8.92**	7.14	29.3	9.10
	Ours (100%)	13	8.94	**7.13**	**28.9**	**8.77**
	All	29	9.11	7.64	29.3	9.60
KP-H21	Baseline	6	9.22	7.73	29.0	9.55
	Ours (90%)	4	9.23	7.54	29.0	9.42
	Ours (99%)	11	9.12	7.37	28.8	9.08
	Ours (100%)	15	9.15	7.36	**28.7**	8.97
	All	29	**9.11**	**7.30**	**28.7**	**8.90**

Table 2: *For each of the three denoiser architectures, we report the metrics from three specialized denoisers from our method, on the feature sets predicted to give 90%, 99%, and 100% of the benefit over using no features. We compare these denoisers with the ones specialized on the baseline feature set and all-feature set, and report their SMAPE and DSSIM errors averaged respectively over the validation ($\mathcal{D}_v$) and test ($\mathcal{D}_t$) datasets. Our selected feature sets consistently improve the denoising quality over the baseline feature sets, and lead to comparable denoising error when compared with the all-feature denoisers while requiring a lot less features.*

tion as it chooses multiple scattering and single scattering initially as they correlate well with the reference, but as indicated by the probe denoiser, they do not help with the denoising task. In contrast, the independent selection (green) manages to select the best performing feature at the beginning but quickly falls behind as it cannot account for redundancy between features. Finally, we also observe the suboptimal performance of using a denoiser trained without dropout (purple) as an oracle compared to our probe denoiser. Even though the discrepancy between methods diminishes for a high relative number of features, we note that no other approximate method was able to outperform our feature selection method on this task.

7.3. Feature set selection with different architectures

For three denoiser architectures, KP-H21, DP-H20, and KP-V18, we run our selection pipeline on the 29-feature set to identify near-optimal feature subsets. In Figure 5, we show the feature selection results by plotting the features' relative gain to the predictive loss reduction in the order of selection, where the relative quality gain is computed according to Equation (10). We then use $\xi = 90\%$ as the quality criterion to select a small feature set that our method predicts to reap most of the benefit and another $\xi = 99\%$ to select a set that is moderately sized but near to the predicted optimal set. The figure shows that the feature set to reach 90% relative quality gain contains the same 4 features for KP-H21 and DP-H20, despite different ordering, but KP-V18 reports a different set with 6 features. For 99% of the benefit, the three probe denoisers predict sets with 8 (DP-H20), 11 (KP-H21) and 14 (KP-V18) features. We also keep the candidate feature sets with the lowest predictive loss (*i.e.*, $\xi = 100\%$), which are also different for the three denoisers. This shows the possibility to identify commonly useful features for different denoiser architectures, even though the selection sequence for each architecture most likely will not be identical. Also, note that KP-V18 reports a positive contribution to denoising error for up to 22 features, while the other two report optimal feature sets much earlier. This is potentially due to the difference in network capacity (see Section 6.2), as we believe that larger capacity networks will be able to make use of more features.

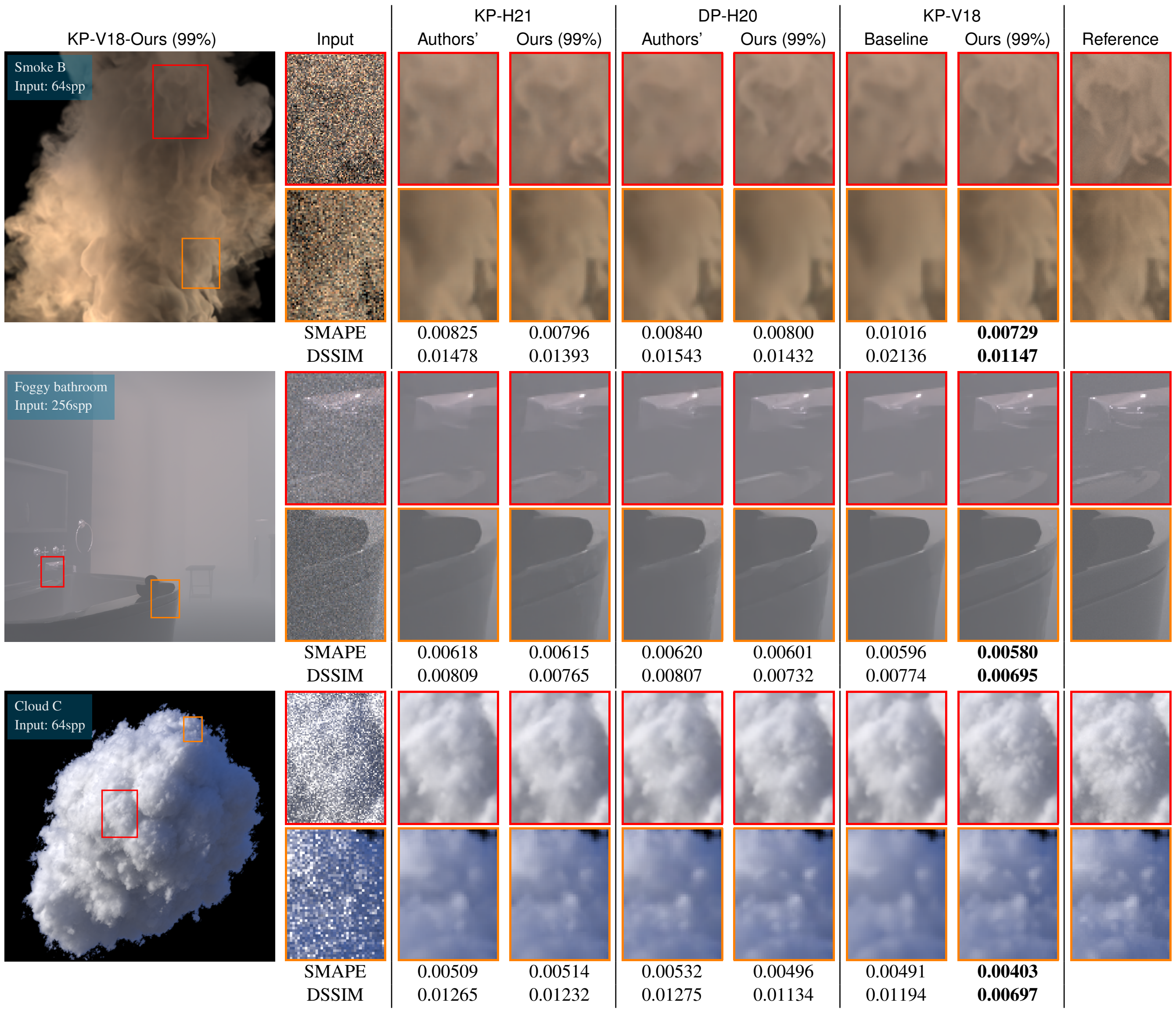
After selecting from the candidate feature sets, we specialize each denoiser architecture on the respective selected feature sets. In Table 2 we compare the error of specialized denoisers on the randomized validation set $\mathcal{D}_v$ and the hand-picked test set $\mathcal{D}_t$.

Overall, we observe that our method's selected volumetric feature sets yield lower errors than the baselines on both metrics for $\mathcal{D}_t$ and DSSIM on $\mathcal{D}_v$ for all three architectures, and lead to lower SMAPE on $\mathcal{D}_v$ in the majority of cases. Notably, the 99% sets contain less than half of the complete set of hand-crafted features and can even slightly outperform denoisers trained on the entire set (All) in some cases, highlighting the redundancies within the 29 features. Furthermore, with the help of our selection method, we can identify feature sets (Ours 90%) with less than half the size of the best-performing sets that, in most cases, are still improving over the baselines. Still interesting is the fact that, when using all features, DP-H20 degrades significantly, while the two kernel-predicting denoisers do not degrade as much. This could support the choice of kernel-predicting approaches for denoising as they seem more capable of dealing with redundant features.

We provide a selection of visual volume denoising comparisons from our test set $\mathcal{D}_t$ in Figure 8. We recommend viewing these results on a screen. These comparisons demonstrate the quality improvements with a set of features identified by our method over the original volumetric features proposed by each baseline denoiser. Overall, we observe increased volumetric details in heterogeneous volumes (top and bottom row) both on smoke and clouds. For homogeneous volumes (middle row), the features help reconstruct silhouettes of objects in the scene better. Accurate reconstructions of these silhouettes are essential for preserving object edges when composed with the surface component.

The baseline with the most significant improvement is KP-V18, as it was originally proposed without any volumetric features. With the addition of 14 features, we observe a significant increase in denoising quality compared to the baseline. In our comparisons, when augmented with our feature set, this offline denoiser yields the highest overall denoising quality. For DP-H20 and KP-H21 we observe a more moderate increase in the denoising quality when using our 99% feature sets, compared to the original feature sets, which is still visually discernible.

Finally, in Figure 9 we provide visual comparisons between denoisers specialized on our 99% feature sets and ones trained on all features. Despite using less than half of all the features, denoisers trained with our feature sets can perform on par with all-feature denoisers.



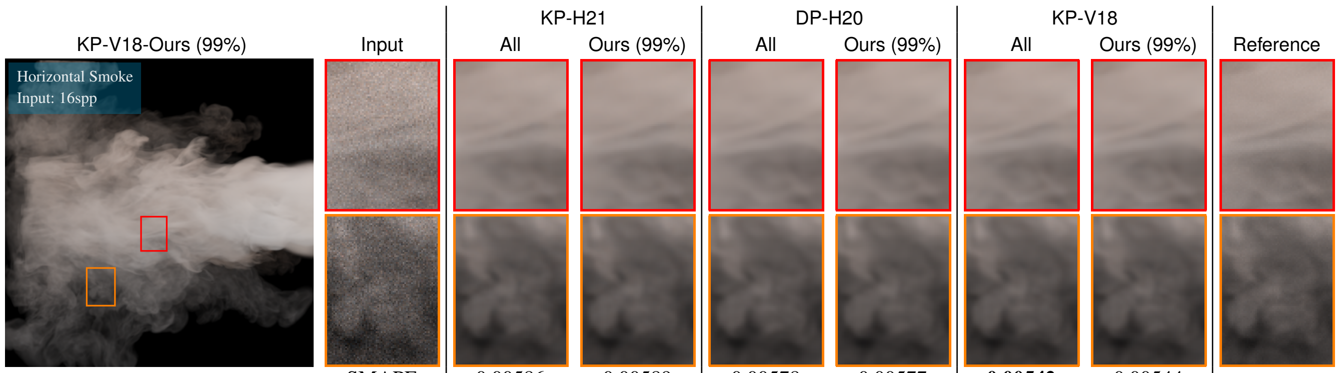
		KP-H21 Authors'	KP-H21 Ours (99%)	DP-H20 Authors'	DP-H20 Ours (99%)	KP-V18 Baseline	KP-V18 Ours (99%)
Smoke B	SMAPE	0.00825	0.00796	0.00840	0.00800	0.01016	**0.00729**
	DSSIM	0.01478	0.01393	0.01543	0.01432	0.02136	**0.01147**
Foggy bathroom	SMAPE	0.00618	0.00615	0.00620	0.00601	0.00596	**0.00580**
	DSSIM	0.00809	0.00765	0.00807	0.00732	0.00774	**0.00695**
Cloud C	SMAPE	0.00509	0.00514	0.00532	0.00496	0.00491	**0.00403**
	DSSIM	0.01265	0.01232	0.01275	0.01134	0.01194	**0.00697**

Figure 8: *Using our selected feature sets improves the denoising quality compared to baselines, which either use no features (KP-V18) or author-proposed features (DP-H20) and (KP-H21). Our selected set is able to recover finer details in a variety of volumetric effects. The exposure of some crops is adjusted to better illustrate the results. © Disney / Pixar*

8. Limitations and Future Work

More advanced feature selection method. Our proposed pipeline includes a greedy feature selection method that only observes the effect of one additional feature at each step. The greedy construction method captures redundancies between the features, which plays an important role in denoising quality, but in principle it could overlook features that only perform well in combination but not in isolation. More sophisticated feature selection approaches, such as ones based on Shapley values [CRD05, AOG19], might be favorable in this situation as they consider the effect of additional feature sets with more than one element. Note that the computation cost of these approaches is often much higher than our greedy approach, and random sampling might be needed [CGT09]. Nevertheless, we are not able to identify such interactions between features in our current feature sets, but this could be the case for other feature sets and tasks.

Aggregation over pixels. In the course of our study, we decided to base our feature selection on per-feature-set loss values, which are aggregated across pixels for each image and across images in the selection set. This approach conforms with the convention of evaluating denoisers' performance by comparing their average denoising error over a test set [XZW*19, IFME21]. However, this might not always be desirable for feature selection because features can provide significant benefits in small regions but not for most of the pixels, resulting in low average contribution. Therefore, it might be interesting to try other aggregation methods based on peak or



	KP-H21 All	KP-H21 Ours (99%)	DP-H20 All	DP-H20 Ours (99%)	KP-V18 All	KP-V18 Ours (99%)
SMAPE	0.00586	0.00589	0.00578	0.00577	**0.00542**	0.00544
DSSIM	0.00592	0.00600	0.00596	0.00602	**0.00526**	0.00530

Figure 9: *Our method can select feature sets that result in denoising quality similar to that of denoisers trained on the full 29-feature set for all three architectures. By employing our method, we can save more than half of the features without losing quality. The exposure of some crops is adjusted to better illustrate the results.*

worst-case performance, *e.g.*, by using quantiles over the error distributions.

Other volumetric effects. We consider three types of volumetric effects in this work, namely ambient homogeneous, smoke plumes, and clouds, as they appear frequently in production scenarios [FWKH17]. There are also other volumetric effects such as volumetric emission (*e.g.*, fire) and subsurface scattering (*e.g.*, skin) [NGHJ18]. Denoisers for renderings of medical data have also been studied [HMES20]. Our method is not special to particular types of volumetric effects, and in principle can be used to discover good feature sets for denoising these effects as well.

Temporal denoising for volumetric effects. In this work, we show that our selected auxiliary features effectively reduce the denoising error when used for denoising static images with volumetric effects. Still, in production scenarios, it is often desirable to have denoisers that work with sequences because of the high correlation between neighboring frames [VRM*18,CKS*17]. It can be that the most important features for temporal denoising are not identical to the best features selected in this work. For example, the velocity field can be an excellent candidate to indicate correspondences between neighboring frames inside volume regions. We leave the exploration of temporal information to future work and believe that our feature selection pipeline can also be applied there.

Other reconstruction tasks. Though we select the best features targeting the denoising task on volumetric scenes, our proposed feature set can serve as a good starting point for other rendering-related reconstruction tasks on volumetric effects, such as super-resolution, adaptive sampling, and frame interpolation. Moreover, the proposed feature selection pipeline provides a principled approach to selecting the most useful features for the task and can be employed for various tasks where auxiliary features are used. For example, we can expand the set of auxiliary features for surfaces, *e.g.*, with virtual flash image [MJL*13], or gradient-domain features [MKD*16]. As these features are often expensive to compute, their computation cost could also be considered in a cost-benefit tradeoff.

9. Conclusion

In this work, we propose volumetric feature sets that improve the quality of various state-of-the-art denoisers on volumetric renderings. Improvements in the denoising quality of volumes can translate to significant savings for production volume rendering scenarios with massive computation requirements. Moreover, we designed a pipeline for selecting the most helpful set of auxiliary features by using a probe denoiser with feature dropout to alleviate the need for training multiple denoisers during feature selection. We have experimentally demonstrated that our probe denoisers can be good oracles for predicting the resulting denoising quality of various feature sets. In addition, our proposed feature set construction method accounts for redundancies with other features and outperforms other feature selection methods we tested for our task. We demonstrate that from a broad spectrum of proposed features, our method can identify feature sets that improve the denoising quality of state-of-the-art denoising methods compared to the original feature sets proposed by the authors. On the other hand, for large-scale production scenarios where the data storage is a significant cost factor, we demonstrate that our method can help by identifying significantly pruned subsets that are as effective as the entire set. Finally, we believe that our proposed feature selection process is not limited to volumetric features and that it opens the door for a more principled study of the impact of features on other tasks.

Acknowledgements

We thank Thijs Vogels, Shilin Zhu, and the anonymous reviewers for their constructive feedback, and thank Mark Meyer and the Pixar Animation Studios for providing the cloud volumes used throughout the project. We thank Doriano vanEssen for creating the bathroom scene in “Foggy bathroom”.

References

[AAB*15] ABADI M., AGARWAL A., BARHAM P., ET AL.: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL: http://tensorflow.org/. 8

[AOG19] Ancona M., Oztireli C., Gross M.: Explaining deep neural networks with a polynomial time algorithm for shapley value approximation. In *Proceedings of the 36th International Conference on Machine Learning* (09–15 Jun 2019), Chaudhuri K., Salakhutdinov R., (Eds.), vol. 97 of *Proceedings of Machine Learning Research*, PMLR, pp. 272–281. URL: https://proceedings.mlr.press/v97/ancona19a.html. 3, 12

[BCSMAB15] Bolón-Canedo V., Sánchez-Maroño N., Alonso-Betanzos A.: *Feature selection for high-dimensional data*. Springer, 2015. 3, 7

[BHHM20] Back J., Hua B.-S., Hachisuka T., Moon B.: Deep combiner for independent and correlated pixel estimates. *ACM Trans. Graph. 39*, 6 (nov 2020). URL: https://doi.org/10.1145/3414685.3417847, doi:10.1145/3414685.3417847. 3

[BJ17] Bitterli B., Jarosz W.: Beyond points and beams: Higher-dimensional photon samples for volumetric light transport. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 1–12. 2

[BVM*17] Bako S., Vogels T., Mcwilliams B., Meyer M., Novák J., Harvill A., Sen P., Derose T., Rousselle F.: Kernel-predicting convolutional networks for denoising Monte Carlo renderings. *ACM Trans. Graphics (Proc. SIGGRAPH) 36*, 4 (July 2017), 97:1–97:14. 3, 6

[CGT09] Castro J., Gómez D., Tejada J.: Polynomial calculation of the shapley value based on sampling. *Comput. Oper. Res. 36*, 5 (May 2009), 1726–1730. doi:10.1016/j.cor.2008.04.004. 3, 12

[Cha13] Chandrasekhar S.: *Radiative transfer*. Courier Corporation, 2013. 2

[CHY21] Cho I.-Y., Huo Y., Yoon S.-E.: Weakly-supervised contrastive learning in path manifold for monte carlo image reconstruction. *ACM Transactions on Graphics (TOG) 40*, 4 (2021), 38:1–38:14. URL: https://doi.org/10.1145/3450626.3459876. 3

[CKS*17] Chaitanya C. R. A., Kaplanyan A. S., Schied C., Salvi M., Lefohn A., Nowrouzezahrai D., Aila T.: Interactive reconstruction of Monte Carlo image sequences using a recurrent denoising autoencoder. *ACM Trans. Graphics (Proc. SIGGRAPH) 36*, 4 (July 2017), 98:1–98:12. 2, 13

[CPP*05] Cerezo E., Pérez F., Pueyo X., Seron F. J., Sillion F. X.: A survey on participating media rendering techniques. *The Visual Computer 21*, 5 (2005), 303–328. 3

[CRD05] Cohen S., Ruppin E., Dror G.: Feature selection based on the shapley value. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence* (San Francisco, CA, USA, 2005), IJCAI'05, Morgan Kaufmann Publishers Inc., pp. 665–670. 3, 12

[CSWJ18] Chen J., Song L., Wainwright M. J., Jordan M. I.: Learning to explain: An information-theoretic perspective on model interpretation. In *35th International Conference on Machine Learning, ICML 2018* (2018), vol. 2, pp. 1386–1418. arXiv:1802.07814. 3

[dN21] d'Eon E., Novák J.: Zero-variance transmittance estimation. In *Eurographics Symposium on Rendering* (June 2021), The Eurographics Association. 2

[DP03] Ding C., Peng H.: Minimum redundancy feature selection from microarray gene expression data. In *Computational Systems Bioinformatics. CSB2003. Proceedings of the 2003 IEEE Bioinformatics Conference. CSB2003* (2003), pp. 523–528. doi:10.1109/CSB.2003.1227396. 3, 10

[FHH*19] Fascione L., Hanika J., Heckenberg D., Kulla C., Droske M., Schwarzhaupt J.: Path tracing in production: part 1: modern path tracing. In *ACM SIGGRAPH 2019 Courses*. 2019, pp. 1–113. 3

[FPWW89] Flock S. T., Patterson M. S., Wilson B. C., Wyman D. R.: Monte Carlo Modeling of Light Propagation in Highly Scattering Tissues–I: Model Predictions and Comparison with Diffusion Theory. *IEEE Transactions on Biomedical Engineering 36*, 12 (1989), 1162–1168. doi:10.1109/TBME.1989.1173624. 5, 9

[FWKH17] Fong J., Wrenninge M., Kulla C., Habel R.: Production volume rendering: Siggraph 2017 course. In *ACM SIGGRAPH 2017 Courses* (New York, NY, USA, 2017), SIGGRAPH '17, Association for Computing Machinery. doi:10.1145/3084873.3084907. 1, 3, 13

[GGZ*18] Gomez A. N., Gal Y., Zhang I., Swersky K., Hinton G. E.: Targeted Dropout. *32nd Conference on Neural Information Processing Systems (NIPS 2018)*, Nips (2018), 1–4. 3

[GLA*19] Gharbi M., Li T.-M., Aittala M., Lehtinen J., Durand F.: Sample-based monte carlo denoising using a kernel-splatting network. *ACM Trans. Graph. 38*, 4 (2019), 125:1–125:12. 3

[HG41] Henyey L. C., Greenstein J. L.: Diffuse radiation in the galaxy. *Astrophysical Journal 93* (1941), 70–83. 8

[HHCM21] Hofmann N., Hasselgren J., Clarberg P., Munkberg J.: Interactive path tracing and reconstruction of sparse volumes. *Proc. ACM Comput. Graph. Interact. Tech. 4*, 1 (apr 2021). URL: https://doi.org/10.1145/3451256, doi:10.1145/3451256. 2, 3, 4, 5, 8

[HMES20] Hofmann N., Martschinke J., Engel K., Stamminger M.: Neural denoising for path tracing of medical volumetric data. *Proc. ACM Comput. Graph. Interact. Tech. 3*, 2 (Aug. 2020). doi:10.1145/3406181. 1, 2, 3, 4, 5, 8, 13

[HST13] He K., Sun J., Tang X.: Guided image filtering. *IEEE Transactions on Pattern Analysis and Machine Intelligence 35*, 6 (2013), 1397–1409. doi:10.1109/TPAMI.2012.213. 3

[HZE*19] Herholz S., Zhao Y., Elek O., Nowrouzezahrai D., Lensch H. P., Křivánek J.: Volume path guiding based on zero-variance random walk theory. *ACM Transactions on Graphics 38*, 3 (2019), 25. doi:10.1145/3230635. 2

[HZRS16] He K., Zhang X., Ren S., Sun J.: Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition* (2016), IEEE Computer Society, pp. 770–778. 8

[IFME21] Işik M., Fisher M., Mullia K., Eisenmann J.: Interactive Monte Carlo Denoising using Affinity of Neural Features. *ACM Trans. Graph 40* (2021). doi:10.1145/3450626.3459793. 3, 12

[IGMM20] Iglesias-Guitian J. A., Mane P. S., Moon B.: Real-Time Denoising of Volumetric Path Tracing for Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics* (2020). URL: http://www.j4lley.com, doi:10.1109/TVCG.2020.3037680. 3

[Jak10] Jakob W.: Mitsuba renderer, 2010. 7

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *CoRR abs/1412.6980* (2014). arXiv:1412.6980. 9

[KFF*15] Keller A., Fascione L., Fajardo M., Georgiev I., Christensen P. H., Hanika J., Eisenacher C., Nichols G.: The path tracing revolution in the movie industry. In *ACM SIGGRAPH 2015 Courses* (New York, NY, USA, 2015), SIGGRAPH '15, ACM Press, pp. 24:1–24:7. 1

[KMM*17] Kallweit S., Müller T., McWilliams B., Gross M., Novák J.: Deep scattering: Rendering atmospheric clouds with radiance-predicting neural networks. *ACM Trans. Graph. (Proc. of Siggraph Asia) 36*, 6 (Nov. 2017). doi:10.1145/3130800.3130880. 2

[KVSF20] Kumar I. E., Venkatasubramanian S., Scheidegger C., Friedler S. A.: Problems with Shapley-value-based explanations as feature importance measures. In *37th International Conference on Machine Learning, ICML 2020* (2020), vol. PartF16814, pp. 5447–5456. arXiv:2002.11097. 3

[MH20] Munkberg J., Hasselgren J.: Neural denoising with layer embeddings. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 1–12. 3

[MJL*13] Moon B., Jun J. Y., Lee J., Kim K., Hachisuka T., Yoon S.-E.: Robust image denoising using a virtual flash image for

Monte Carlo ray tracing. *Computer Graphics Forum 32*, 1 (2013), 139–151. 3, 13

[MKD*16] MANZI M., KETTUNEN M., DURAND F., ZWICKER M., LEHTINEN J.: Temporal gradient-domain path tracing. *ACM Transactions on Graphics (TOG) 35*, 6 (2016), 1–9. 13

[NGHJ18] NOVÁK J., GEORGIEV I., HANIKA J., JAROSZ W.: Monte carlo methods for volumetric light transport simulation. 551–576. 3, 13

[NSJ14] NOVÁK J., SELLE A., JAROSZ W.: Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph. 33*, 6 (2014), 179–1. 2

[PJH16] PHARR M., JAKOB W., HUMPHREYS G.: *Physically Based Rendering: From Theory to Implementation (3rd ed.)*, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, Oct. 2016. 3

[RFB15] RONNEBERGER O., FISCHER P., BROX T.: U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention - 18th International Conf.* (2015), Springer, pp. 234–241. 3

[RMZ13] ROUSSELLE F., MANZI M., ZWICKER M.: Robust denoising using feature and color information. *Computer Graphics Forum 32*, 7 (2013), 121–130. 2, 3

[Sha16] SHAPLEY L. S.: *17. A Value for n-Person Games*. Princeton University Press, 2016, pp. 307–318. doi:doi:10.1515/9781400881970-018. 3

[SN20] SUNDARARAJAN M., NAJMI A.: The many shapley values for model explanation. In *37th International Conference on Machine Learning, ICML 2020* (nov 2020), vol. PartF16814, PMLR, pp. 9210–9220. URL: https://proceedings.mlr.press/v119/sundararajan20b.html, arXiv:1908.08474. 3

[VAN*19] VICINI D., ADLER D., NOVÁK J., ROUSSELLE F., BURLEY B.: Denoising Deep Monte Carlo Renderings. *Computer Graphics Forum 38*, 1 (2019), 316–327. URL: https://doi.org/10.1111/cgf.13533., doi:10.1111/cgf.13533. 3

[VRM*18] VOGELS T., ROUSSELLE F., MCWILLIAMS B., RÖTHLIN G., HARVILL A., ADLER D., MEYER M., NOVÁK J.: Denoising with kernel prediction and asymmetric loss functions. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018) 37*, 4 (2018), 124:1–124:15. doi:10.1145/3197517.3201388. 1, 2, 3, 4, 8, 13

[WBSS04] WANG Z., BOVIK A., SHEIKH H., SIMONCELLI E.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (April 2004), 600–612. 6

[WC20] WOJTAS M. A., CHEN K.: Feature importance ranking for deep learning. In *Advances in Neural Information Processing Systems* (2020), vol. 2020-Decem. URL: https://github.com/maksym33/FeatureImportanceDL, arXiv:2010.08973. 3

[Wel82] WELCH W. J.: Algorithmic Complexity: Three NP-Hard Problems in Computational Statistics. *Journal of Statistical Computation and Simulation 15*, 1 (1982), 17–25. doi:10.1080/00949658208810560. 3

[XSW*20] XU Z., SUN Q., WANG L., XU Y., WANG B.: Unsupervised image reconstruction for gradient-domain volumetric rendering. *Computer Graphics Forum 39*, 7 (2020), 193–203. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.14137. 2, 3

[XZW*19] XU B., ZHANG J., WANG R., XU K., YANG Y.-L., LI C., TANG R.: Adversarial monte carlo denoising with conditioned auxiliary feature. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2019) 38*, 6 (2019), 224:1–224:12. 2, 3, 4, 12

[YJV19] YOON J., JORDON J., VAN DER SCHAAR M.: Invase: Instance-wise variable selection using neural networks. In *7th International Conference on Learning Representations, ICLR 2019* (2019). 3

[ZAW19] ZHAO Z., ANAND R., WANG M.: Maximum relevance and minimum redundancy feature selection methods for a marketing machine learning platform. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)* (2019), pp. 442–452. doi:10.1109/DSAA.2019.00059. 3

[ZJL*15] ZWICKER M., JAROSZ W., LEHTINEN J., MOON B., RAMAMOORTHI R., ROUSSELLE F., SEN P., SOLER C., YOON S.-E.: Recent advances in adaptive sampling and reconstruction for Monte Carlo rendering. *Computer Graphics Forum (Proc. Eurographics) 34*, 2 (May 2015), 667–681. 3

[ZMV*21] ZHANG X., MANZI M., VOGELS T., DAHLBERG H., GROSS M., PAPAS M.: Deep Compositional Denoising for High-quality Monte Carlo Rendering. *Computer Graphics Forum 40*, 4 (2021), 1–13. doi:10.1111/cgf.14337. 2, 3, 4

Appendix A: Pseudo code

In Algorithm 1, we provide pseudo code of our greedy candidate feature set construction algorithm.

Algorithm 1: Greedy construction

```
Input: $\mathbf{P}$, set of all features
       $\mathcal{D}_s$, selection dataset
       $\tilde{g}$, trained probe denoiser
       $\varepsilon$, loss function
Output: $\bar{\mathbf{P}} = \{\mathbf{S}_0, \mathbf{S}_1, \ldots, \mathbf{S}_{|\mathbf{P}|}\}$, candidate feature sets at each size
        $\bar{L} = \{\ell_0, \ell_1, \ldots, \ell_{|\mathbf{P}|}\}$, losses of each candidate feature set

/* Define mean loss over selection dataset (Eq. 9). */
$\ell(\mathbf{S}) := \frac{1}{|\mathcal{D}_s|} \sum_{(\mathbf{I},\bar{\mathbf{I}}) \in \mathcal{D}_s} \varepsilon\big(\tilde{g}(\mathbf{I},\mathbf{S}), \bar{\mathbf{I}}\big)$
/* Initialize with empty feature set */
$\mathbf{S}_0 \leftarrow \emptyset$
$\ell_0 \leftarrow \ell(\mathbf{S}_0)$
$\bar{\mathbf{P}} \leftarrow \{\mathbf{S}_0\}$
$\bar{L} \leftarrow \{\ell_0\}$
/* Gradually expand the candidate feature sets */
for $i \leftarrow 1, 2, \ldots, |\mathbf{P}|$ do
    // Select the best one additional feature
    $L \leftarrow \{\}$
    for $p \in \mathbf{P} \setminus \mathbf{S}_{i-1}$ do
        $\ell' \leftarrow \ell(\mathbf{S}_{i-1} \cup \{p\})$
        $L$.append$\big((\ell', p)\big)$
    end
    $(\ell_i, p_i) \leftarrow \min(L)$
    $\mathbf{S}_i \leftarrow \mathbf{S}_{i-1} \cup \{p_i\}$
    $\bar{\mathbf{P}}$.append$(\mathbf{S}_i)$
    $\bar{L}$.append$(\ell_i)$
end
return $\bar{\mathbf{P}}$, $\bar{L}$
```