Differentiable Design Galleries: A Differentiable Approach to Explore the Design Space of Transfer Functions

Category: Research

Paper Type: algorithm/technique

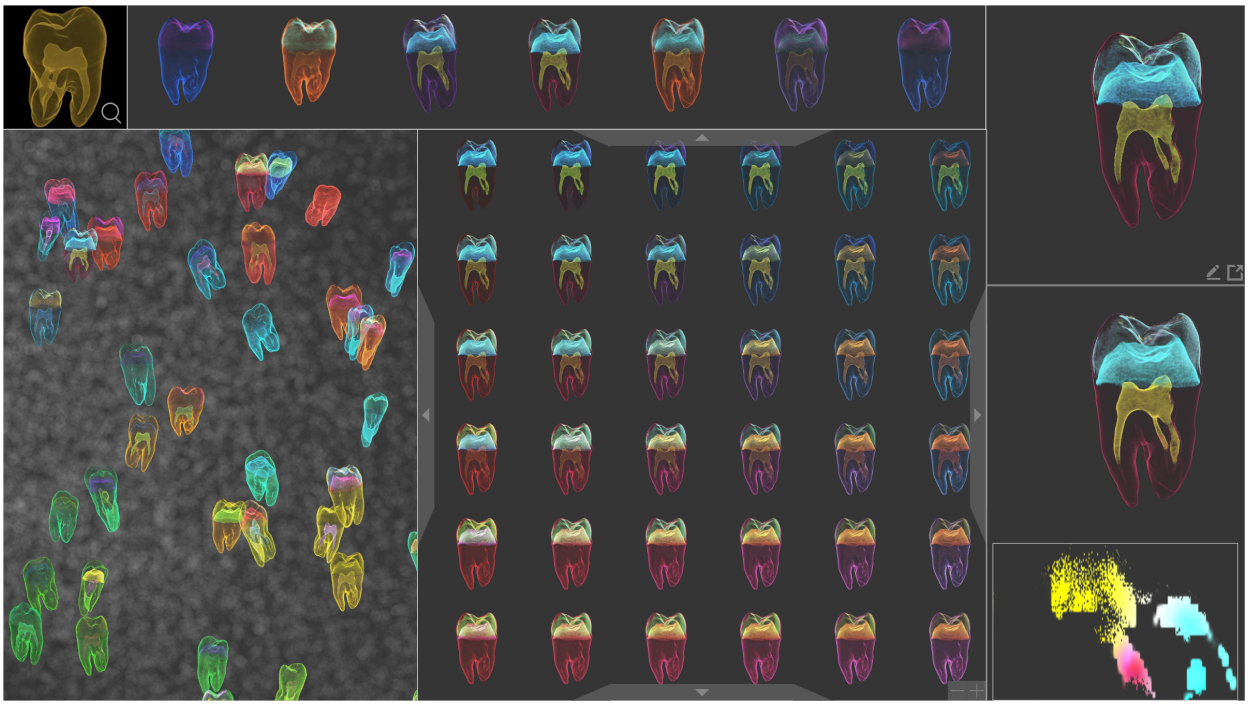

Fig. 1: Overview of the approach

Abstract—The transfer function is crucial for direct volume rendering (DVR) to create an informative visual representation of volumetric data. However, manually adjusting the transfer function to achieve the desired DVR result can be time-consuming and unintuitive. In this paper, we propose Differentiable Design Galleries, an image-based transfer function design approach to help users intuitively explore the design space of transfer functions by taking advantage of the recent advances in deep learning and differentiable rendering. Specifically, we leverage neural rendering to learn a differentiable design space, which is a continuous manifold representing various types of implicit transfer functions. We further provide a set of visual interfaces to support intuitive query, navigation, and modification to obtain the expected DVR result. The expected DVR result can be reconstructed back to an explicit transfer function with a differentiable direct volume renderer. Experimental results on real volumetric data demonstrate the effectiveness of our method.

Index Terms—Transfer function

1 Introduction

Volume rendering is widely used by scientists, engineers, physicians and artists in various applications. Direct volume rendering (DVR) is regarded as one of the most commonly used and effective volume rendering techniques to explore and depict volumetric data. The expressiveness and flexibility of DVR owe to the large design space of the transfer function, which maps different voxel values to different colors and opacity. Mathematically, any function that maps the domain voxel value to the 4-dimensional (RGBA) range value can be a transfer function. With a carefully chosen transfer function, rich information on volumetric data and visually appealing rendering results can be obtained.

However, finding a proper transfer function is not a trivial task. One of the most commonly used tools are transfer function design widgets. Users can specify and adjust the shape of the target transfer function with those widgets. Those widgets are powerful and flexible for experienced engineers or scientists. But the design process with those widgets is usually time-consuming and unintuitive for non-expert users: subtle changes in the transfer function may lead to an unexpected change in the image domain due to the nonlinearity and complexity of the volume rendering process. To address this issue, some researchers [8, 26, 29--32] propose providing Design galleries (see Fig. 2) to help users explore a pre-defined design space of transfer functions. This approach is more friendly and intuitive for non-expert users to find their desired transfer function. However, some problems limit the usability of these approaches: First, when the user already has a target design style in mind, an efficient method to query the target design in the design space is desired. Second, it is unclear how to efficiently navigate this high dimensional transfer function design space [17]. Third, when the users become interested in some exemplar images in the design galleries and want to make further exploration or modification to the transfer function behind it, they have to go back to the traditional transfer function widgets.

To address the problem mentioned above, we propose introducing differentiability to the traditional Design Galleries approaches by taking advantage of the recent advances in neural and differentiable rendering.

With the Differentiable Design Galleries we provide, various new applications to explore the transfer function design space are possible. For example, users can:

- Use a target direct volume rendering image as an input query to find the closest transfer function design in the design space. The image can even come from literature or other similar volumetric data.
- Choose some design exemplars in the Design Gallery and display a much more fine-grained design sub-space between those exemplars. Then navigate this design sub-space with intuitive operations like pan, zoom in, and zoom out.
- Make intuitive modifications on a target design exemplar without using transfer function widgets. And reconstruct the new transfer function when satisfactory modification is reached.
- Generate smooth animations between different transfer function designs, which can be used for purposes like exploration history presentation [24], transfer function morphing creation [44], and focus plus context visualization [45].

The principal challenge of constructing a differentiable Design Gallery is mapping the design space represented by discrete exemplars into a differentiable transfer function design space. To deal with this, we propose a neural DVR analyzer. This neural DVR analyzer can analyze exemplar design image and map it into a vector (we call it latent TF) in the differentiable transfer function design space. To ensure the differentiable transfer function design space is well-organized and perceptually reasonable, we specifically tailor the neural DVR analyzer's training task to imitate a DVR expert's transfer function analysis process.

Once the differentiable transfer function design space is constructed, users can use various applications we provided to explore the design space and find the target transfer function design. This target transfer function design is represented in the form of a neural-rendered exemplar image. To obtain the explicit transfer function for this target design, we propose using a differentiable DVR renderer to do transfer function reconstruction.

Our code and model will be available at https://github.com/ after the review cycle, and we summarize our contributions as follows:

- A novel approach that introduces differentiability to the traditional design galleries method. This approach provide users with more degree of freedom and efficiency during the transfer function design space exploration.
- A neural DVR analyzer and its training strategy that can map the transfer design space represented by discrete exemplars into a well-organized differentiable high-dimensional space.
- Experiments and example applications to demonstrate the effectiveness of our approach.

2 Related Work

2.1 Transfer Function Design

Transfer function is regarded as a central component of direct volume rendering. How to design transfer functions and facilitate the transfer function design process has been continuously discussed in the past three decades.

The most basic and common transfer function is the one-dimensional transfer function, which maps raw scalar intensity voxel value to color and opacity. However, when features of interest have overlapped intensity or are affected by noise, the one-dimensional transfer function is inadequate [23]. Different types of multidimensional transfer function are proposed to better depict volume data. Simple derived attributes like gradients or second derivatives are first exploited to increase the transfer function's classification capability [20]. Later, the community proposed a variety of attributes for transfer function design. For example, curvatures [19] can be used to better deliver surfaces information,

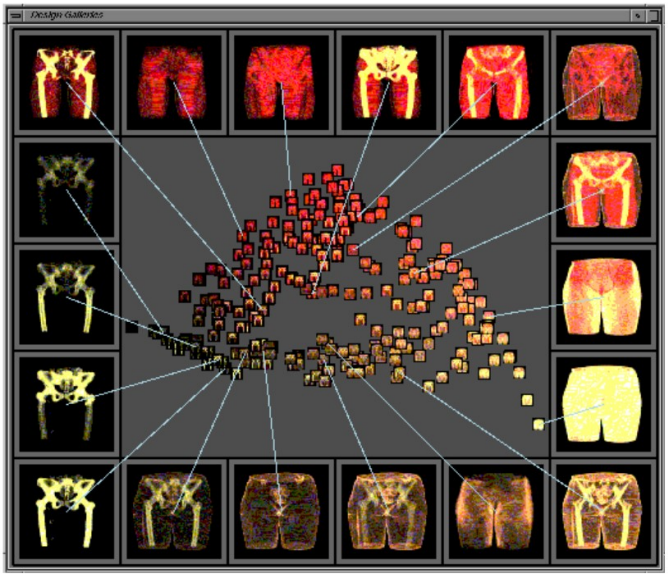

Fig. 2: The design gallery approach by Marks et al. [26] to display the transfer function design space of CT legs volumetric data. Users can pick the rendered result they like instead of manually setting a transfer function.

approximated size [3] helps to locate features with certain size, visibility [5] emphasizes features' visibility from different viewports, and ambient occlusion [4] reveal occlusion relationships.

However, while adding more attributes increases transfer functions' expressiveness, the difficulty of transfer function creation also rises [22], which is challenging for non-expert users. To deal with this, different approaches are proposed to avoid or alleviate the miserable trial-and-error process for the transfer function design. Those approaches can be classified into two main categories: data-centric and image-based methods [28].

A general question shared by those data-centric approaches is: how to effectively constraints the design space of transfer functions based on our prior knowledge about the data. Tzeng and Ma [35] propose an iterative self-organizing data analysis technique to find the clusters in 2D histogram space, then the user's interaction is simplified to modify those clusters. Other strategies to cluster the feature space are also explored, like using kernel density estimation [25], gaussian mixture model [38], or valley cell-based clustering [39]. Dimension reduction methods are utilized to simplify high-dimensional transfer functions to the lower-dimensional space [1, 6, 10, 18, 21, 46]. Semantics-based transfer functions [34] are proposed to aggregate low-level transfer functions for more intuitive transfer function design. Ip et al. [15] leverage information theory to help the user hierarchically cut the 2D gradient intensity histogram into segments.

While the data-centric methods still leave a non-intuitive gap between the transfer function domain and the resulting image. The image-based methods allow users to directly explore the design space from the image side. He et al. [11] propose a semi-automatic approach that lets the users make selection from rendering results, and the users' selection will supervise the genetic algorithm to generate better transfer functions. Guo et al. [9] demonstrated a WYSIWYG volume exploration framework to enable the users to modify the transfer functions by directly editing the rendering result with the tools they provided. Wu and Qu [45] propose an interactive transfer function design approach by blending different direct volume rendered images. Design Galleries [26, 29--32] try to generate enough samples to span the transfer function design space the user is going to explore. The users only need to select among the presented possibilities. However, how to provide the users with finer degree of control when they are not satisfied with the initially generated images remains an unsolved problem. And how to efficiently navigate the high dimensional transfer function

design space represented by the design galleries is also under discussion [17]. Our work tries to deal with those problems by introducing differentiability to the traditional Design Galleries approach.

2.2 Deep learning for Scientific Visualization Generation

DL techniques bring crucial benefits to the SciVis research community. In 2022, Wang and Han [37] made a thorough survey about the application of deep learning techniques for scientific visualization. However, the number of papers for visualization generation is much less than those for data generation.

Berger et al. [2] made the first attempt to use neural DVR renderer with an explicit texture transfer function as input to replace traditional DVR renderer. Their neural DVR renderer can support transfer function sensitivity analysis and provide an overview of possible 1D opacity transfer functions, which is helpful in guild users' volume exploration process. He et al. [12] instead use a neural DVR renderer to directly learn a mapping from simulation parameters to the rendering result, which supports efficient exploration of simulation parameter space. Hong et al. [13] proposes using DVR image as an implicit transfer function input for neural renderer such that user can use image level editing to replace explicit transfer function editing. Weiss and Navab [40] try to integrate the neural network into the traditional DVR pipeline such that the rendering process can be optimized from manually adjusted reference images. Weiss et al. [41] feed low-resolution normal and depth field to neural renderer to obtain high-resolution isosurface maps. Weiss et al. [42] propose a neural rendering framework that uses one sub-network to generate a sparse adaptive sampling structure and another sub-network to generate high resolution image based on the sampling result. Weiss and Westermann [43] present a memory-efficient differentiable DVR rendering solution and show the potential of integrating differentiable DVR rendering with different differentiable loss terms and neural networks.

Our work extends this line of research by integrating neural rendering and differentiable rendering to help users explore the design space of transfer functions.

3 Overview

Fig. 3 presents the workflow of our approach, which consists of three main components. First, given a specific volumetric data and a predefined transfer function design space for it, we randomly generate Design Exemplars using a traditional DVR renderer and transfer functions sampled within the design space. Then we train a Neural DVR Analyzer to generate a differentiable design space based on those Design Exemplars. The training task is specifically tailored such that the generated differentiable design space is well-organized and perceptually reasonable (Sec. 4). Second, we propose a couple of methods to help the users explore this differentiable design space, and integrated them into a interactive interface (see Fig. 1). With our interactive interface, users can (1) Gain a rough overview of the Design Exemplars within the design space just like traditional Design Galleries. (2) Use an image as a Design Exemplar to query the most similar design within the Differentiable Design Space. (3) Navigate the high dimensional Differentiable Design Space with intuitive operations. (4) Make intuitive modifications on a target design without resorting to transfer function widgets (Sec. 5). Third, when the users are satisfied with a certain transfer function design (represented as a Neural Rendered DVR Image), we leverage a Differentiable DVR Renderer to help the user convert that back to explicit transfer functions in texture form (Sec. 6).

4 Differentiable Design Space Generation

In this section, We will discuss how we generate the Differentiable Design Space. The generation process can be regarded as a process for finding a suitable mapping function F, which maps discrete design exemplars into high dimensional vectors (latent TFs). To facilitate the downstream applications for the differentiable design space exploration, we want the mapping FF to have the following properties:

(1) Design Exemplars that are perceptually similar should be map to vectors that are close together in the Differentiable Design Space.

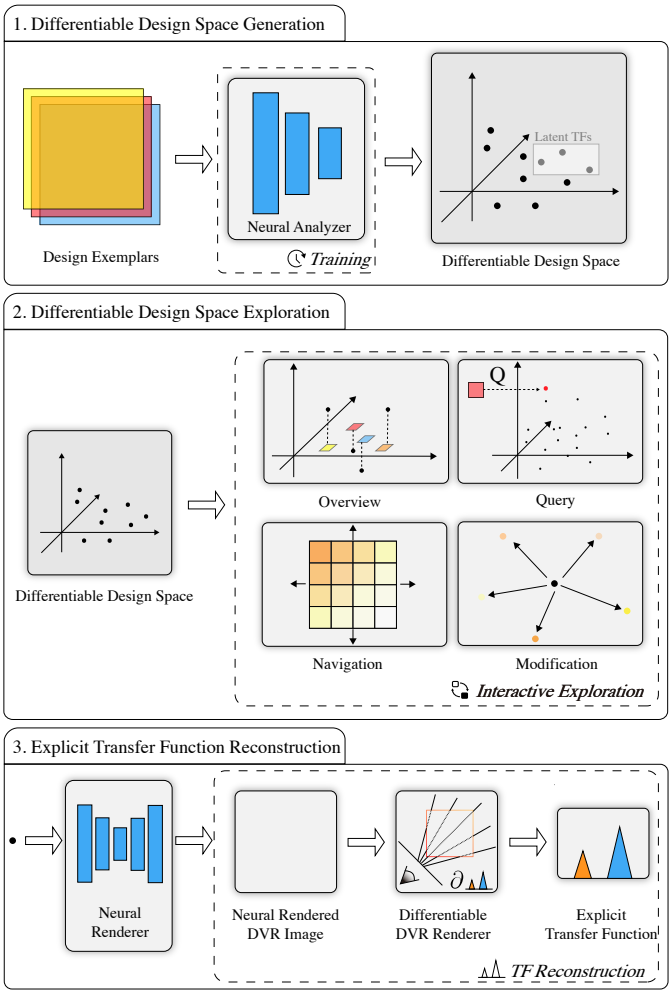

Fig. 3: Approach Overview

(2) The mapping should be invariant with respect to the viewport of the Design Exemplars. For example, two Design Exemplars with different viewport but the same underlying transfer function should be mapped to the same vector.

We propose to use a Neural DVR Analyzer to learn this mapping F, due to the powerful expressiveness of neural network as a function approximator [14]. And the training task for the Neural DVR Analyzer is designed like a DVR expert's analysis process. As shown in Fig. 4, given a DVR image of a volumetric data as a Design Exemplar, a DVR expert familiar with the volumetric data can analyze the features shown in the Design Exemplar and guess the underlying transfer function. If the guess is good enough, the DVR images rendered by a traditional DVR Renderer with this guessed transfer function should be the similar to the Design Exemplar. The Neural DVR Analyzer also try to analyze the features shown in the Design Exemplar and map it into a latent TF. If the mapping is good enough, DVR images rendered by a Neural Renderer with this mapped latent TF should be similar to the Design Exemplar. Due to the high similarity of the Neural DVR Analyzer's task and the task of the DVR expert, the mapping learned by the Neural DVR Analyzer is likely to be perceptually reasonable. And since we change the viewport for each Design Exemplar, the mapping learned should be robust to viewport changes.

4.1 Overall Architecture

Fig. 11 illustrates the overall architecture of our models. During training, there are three sub-networks: the Neural DVR Analyzer, the Neural DVR Renderer, and the Discriminator. The input of the Neural DVR Analyzer is the Design Exemplar. The Design Exemplar is rendered by

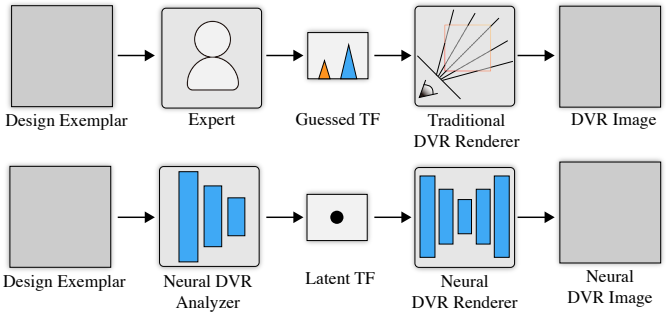

Fig. 4: The design gallery approach by Marks et al. [26] to display the transfer function design space of CT legs volumetric data. Users can pick the rendered result they like instead of manually setting a transfer function.

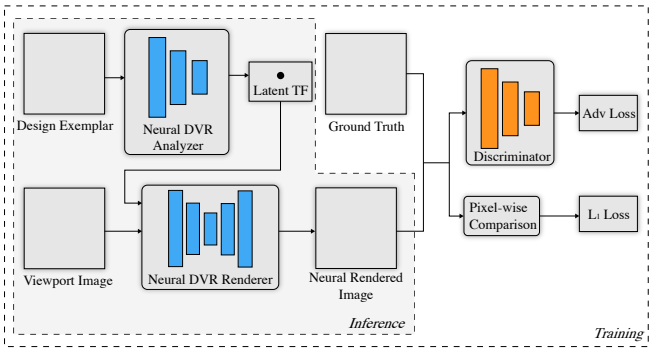

Fig. 5: The design gallery approach by Marks et al. [26] to display the transfer function design space of CT legs volumetric data. Users can pick the rendered result they like instead of manually setting a transfer function.

a traditional DVR Renderer with viewpoint V_{exp} and transfer function TF_{exp}. The Neural DVR Analyzer analyzes the Design Exemplar and maps it into a latent transfer function, which is a high dimensional vector. The input of the Neural DVR Renderer is this latent transfer function and a viewport image. The viewport image is rendered by a traditional DVR renderer with viewpoint V_{view} and an auxiliary transfer function TF_{view} to provide viewport information to the Neural DVR Renderer. The Neural DVR Renderer is expected to render an image that looks like rendered by a traditional DVR renderer with viewpoint V_{view} and transfer function TF_{exp}. To facilitate the training of the Neural DVR Analyzer and the Neural DVR Renderer, we use both pixel-wise comparison and a Neural Discriminator to compute the loss between the Neural Rendered image and the ground truth image. The Neural Discriminator is trained to classify if a given image is from a traditional DVR renderer or a Neural DVR Renderer. The classification result can be used as the adversarial loss [7]. Previous works [2, 12, 13, 16] have shown that this adversarial loss can alleviate the over-smooth problem caused by pixel-wise loss and improve the perceptual quality of the rendered images. Once the training is complete, we do not need the Discriminator anymore, and we can use the Neural DVR Analyzer and the Neural DVR Renderer separately for different downstream tasks.

4.2 Model Details

In this section, we will provide further elaboration on how the Neural DVR Analyzer, Neural DVR Renderer, Discriminator are constructed.

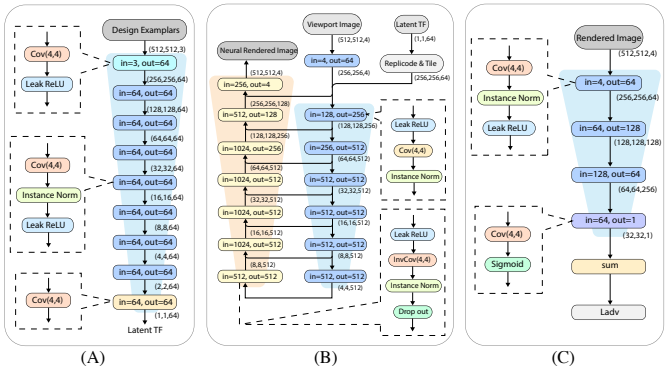

Fig. 6: fig network

4.2.1 Neural DVR Analyzer

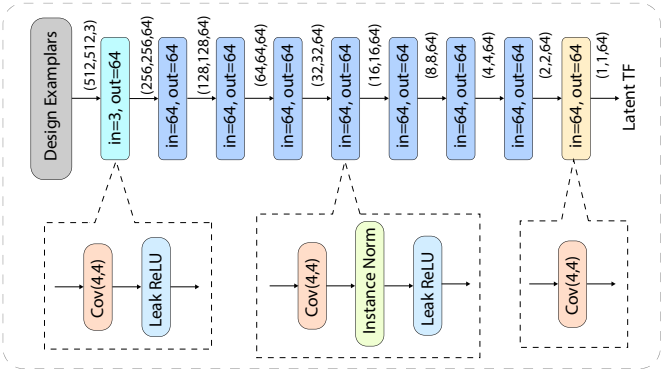

Fig. 7: The design gallery approach by Marks et al. [26] to display the transfer function design space of CT legs volumetric data. Users can pick the rendered result they like instead of manually setting a transfer function.

The architecture of the Neural DVR Analyzer is shown in Fig. 7, which takes a Design Exemplar as input and outputs a 64 dimensional vector, which is the Latent TF. The Design Exemplar is in white or

black background without the alpha channel. We make this design choice since alpha channel is invisible to human, and we want the Neural DVR Analyzer to analyze Design Exemplars like a human DVR expert. The Analyzer consists of a series of down sampling blocks. Each down sampling block have a convolution layer and a leaky ReLu activation layer (except for the last one). The convolution layers use small filters to extract local features from the previous layers, which are commonly used for feature extraction and representation learning tasks. Here we choose the kernel size to be 4×4, the stride to be 2, and the padding to be 1. With such a setting, the height and width of the feature maps will reduce to half after each convolution layer. The leaky ReLu layers introduce non-linearity into the model. We choose leaky ReLu activation function in order to mitigate the "dying ReLu" problem, which can occur when neurons in the network are stuck at zero activation and stop learning. Except for the first and the last down sampling block, we add an extra instance norm layer [36] between the convolution layer and the leaky ReLu layer to stabilize training. We use instance norm layer instead of batch norm layer due to its ability to preserve the variations in style and content features within a single Design Exemplar.

4.2.2 Neural DVR Renderer

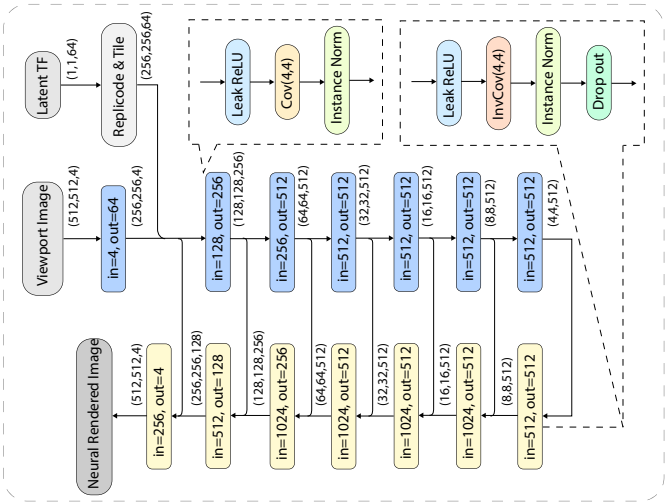

Fig. 8: The design gallery approach by Marks et al. [26] to display the transfer function design space of CT legs volumetric data. Users can pick the rendered result they like instead of manually setting a transfer function.

The architecture of the Neural DVR Renderer is shown in Fig. 8, which takes a latent TF and a viewport image as input, and outputs a neural rendered image. Different from the Design Exemplar which only have RGB channels, the viewport image and the neural rendered image have RGBA channels. We make this design choice such that users can freely decide to display the neural rendered image with white background or black background during the exploration process of the Differentiable Design Space. We use viewport image instead of raw viewport parameters to represent viewport information because previous works [2, 13] show that image input can provide the neural renderer with richer spatial information and increase the stability of training. The overall architecture of the Neural DVR Renderer is a U-net [33], which is consist of a series of down sampling blocks followed by a series of up sampling blocks. Skip connections are added between each down sampling block and up sampling block with the same spatial shape. This helps to enforce spatial consistency in the neural rendered image. The interior structure of the down sampling blocks are similar to the down sampling blocks of the Neural DVR Analyzer. In the up sampling blocks, we use inverse convolution layer to replace the convolution layer. The inverse convolution layer is of 4×4 kernel size, 2 stride and 1 padding. With such a setting, the height and width of the feature maps will double after each inverse convolution layer. Within every up sampling block, a dropout layer is added after the instance norm layer to avoid over-fitting, as suggested by [16].

4.2.3 Discriminator and Losses

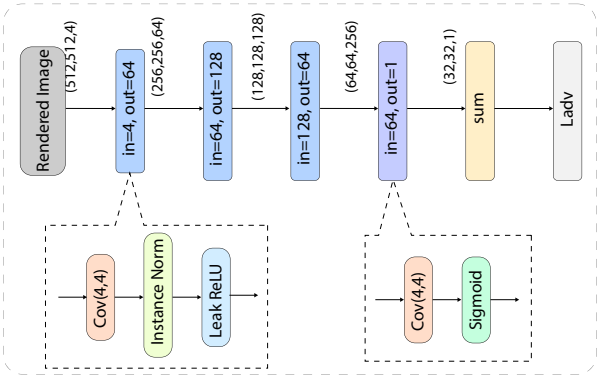

Fig. 9: The design gallery approach by Marks et al. [26] to display the transfer function design space of CT legs volumetric data. Users can pick the rendered result they like instead of manually setting a transfer function.

The architecture of the Discriminator is shown in Fig. 9, which takes the neural rendered imaged I_{pred} or the ground truth image I_{GT} as the input, and output the possibility that this image is a ground truth image. The Discriminator consists of a series of down sampling blocks, which are also used in the Neural DVR Analyzer and the Neural DVR Renderer. In the last down sampling block we use sigmoid activation layer instead the leaky ReLu activation layer because we what the predicted possibility to within range 0-1, where 1 represents that the Discriminator regard the image as a ground truth image. Noted that the output of the last sampling block is still of width 32 and height 32 instead of a single scalar possibility. This is because the discriminator is indeed 32 small discriminators, and each small discriminator has a 32 x 32 receptive field in the input images. Previous work [16] shows that using many small discriminators with small receptive field instead of a large discriminators with global receptive field can improve the performance with the same number of parameters. We use the average of the predicted probabilities of those small discriminators as the finial prediction of the discriminator, which is denoted by $D(I_{pred})$ or $D(I_{GT})$.

We use the method presented in [7] to calculated the adversarial loss as follows:

$$L_{adv} = -\frac{1}{b}\sum_{i=0}^{b-1} \log D(I_{pred}) \tag{1}$$

where b is the batch size. And this adversarial loss can be combined with the l1 (pixel-wise comparison) loss to get the final loss for the neural rendering result:

$$L = L_{adv} + \lambda \|I_{pred} - I_{GT}\|_{loss} \tag{2}$$

where λ is the weight of the L1 loss. In practice we find $\lambda = 1000$ stabilizes training without blurring the neural rendered images. This loss LL is used to optimize the Neural DVR Analyzer and the Neural DVR Renderer together during training.

The Discriminator is also optimized by L_{adv_D} during training, which is defined as follows:

$$L_{adv\_D} = -\frac{1}{b}\sum_{i=0}^{b-1} (\log D_v(I_i) + \log(1 - D_v(I_i)) \tag{3}$$

4.3 Training

4.3.1 Training Data

The set of Design Exemplars used as the training data implicitly define the target design space of transfer functions. Inspired by the transfer

function sampling strategy by Berger et al. [2], we tailor a sampling strategy for Design Exemplar generation. As shown in Fig. 10, the domain of the transfer function is first divided into several feature regions of interest. By doing this, uninterested feature region (e.g. regions of air or noise) can be removed from the target design space. Then, we assign a Gaussian transfer function primitive for each feature region, we call them base GTF (see the black Gaussians drawn with solid line in Fig. 10). For each round of sampling, we randomly choose some feature regions of interest, apply random shift to all of the parameters (color, opacity, mean, std) of the base GTFs in those resign, and combine them together to generate a final transfer function design. The parameters of base GTF and the range for random shift can be manually set to control the target design space. For example, if we want to bias the color of some features of interest toward higher lightness and saturation, we can bias the corresponding base GTFs towards higher lightness and saturation, and decrease the possible range of random shift for lightness and saturation. Once the transfer function for the Design Exemplar TF_{exp} is generated, we render it with a random viewport V_{exp}. The viewport image is rendered with another random viewport V_{view} and the auxiliary transfer function TF_{view}. The auxiliary transfer function is create by assign one representative GTF for each interested feature region and combine them together. The ground truth image is rendered with viewport V_{view} and transfer function TF_{exp}. This sampling strategy can be easily extended to multi-dimensional transfer functions using multi-dimensional GTF proposed by Kniss et al. [**?**]. In our experiment, we use 2D transfer function to generate design space for tooth data since it is better classified with 2D gradient intensity histogram.

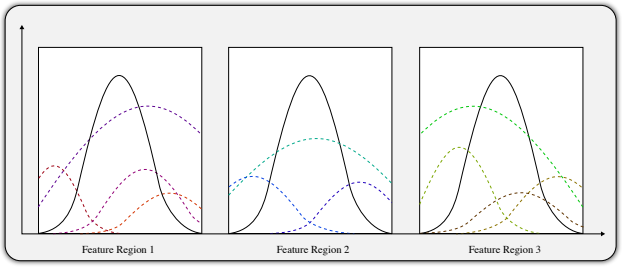

Fig. 10: The design gallery approach by Marks et al. [26] to display the transfer function design space of CT legs volumetric data. Users can pick the rendered result they like instead of manually setting a transfer function.

4.3.2 Training Process

Algorithm 1 Training Process for Differentiable Design Space Generation

Input: Training data includes Design Exemplars $\{I_{exp}\}$, Viewport Images $\{I_{view}\}_{0:N-1}$. and the corresponding ground truth images $\{I_{gt}\}_{0:N-1}\}$, Initial weights u, v, w of the Neural DVR Analyzer, the Neural DVR Renderer R_v, and the Discriminator D_w, respectively.
Output: Optimized weight u and v.

1: **repeat**
2: $\{I_{exp}, I_{view}, I_{GT}\}_{0:b-1}$ sampled from training data
3: $\{latentTF\}_{0:b-1} \leftarrow A_u(\{I_{exp}\}_{0:b-1}\})$
4: $\{I_{pred}\}_{0:b-1}\} \leftarrow R_v(\{LatentTF, I_{view}\}_{0:b-1})$
5: $w \leftarrow Adam(\nabla_w L_{adv\_D}(\{I_{GT}, I_{pred}\}_{0:b-1}; w), w, \alpha, \beta_1, \beta_2)$
6: $v \leftarrow Adam(\nabla_v L(\{I_{GT}, I_{pred}\}_{0:b-1}; v), v, \alpha, \beta_1, \beta_2)$
7: $u \leftarrow Adam(\nabla_u L(\{I_{GT}, I_{pred}\}_{0:b-1}; u), u, \alpha, \beta_1, \beta_2)$
8: **until** convergence

The process of training our Neural DVR Analyzer, Neural DVR Renderer and Discriminator is shown in Algorithm 1. We first initialize the weight of all networks using xavier initialization [**?**]. For each iteration, we sample a batch of image triplets $\{I_{exp}, I_{view}, I_{GT}\}$. The Neural DVR Analyzer first takes the Design Exemplar I_{exp} as input and maps it into a latent TF (line 3). The latent TF and the viewport image is then feed into the Neural DVR Renderer to obtain the predicted image I_{pred} (line 4). The weights of the Discriminator are first updated according to the loss L_{adv_D} defined in Eq. (3) (line 5). Then the weight of the Neural DVR Analyzer and the Neural DVR Renderer are updated according to the loss LL defined in Eq. (2) (line 6,7). We use Adam [**?**] as our weight optimizer with parameters$\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.999$. The batch size b is set to 128 during training.

5 Differentiable Design Space Exploration

To demonstrate the benefits of generating a differentiable design space, we propose a set of interactive tools to help users explore this differentiable design space. With those tools, users can intuitively find their target design without resorting to explicit transfer function widgets.

5.1 Outline view for the Differentiable Design Space

Like previous Design Gallery methods, we use dimension reduction to arrange the Design Exemplars for an outline of the possible designs (see Fig. 1 area b). We perform Principle Component Analysis (PCA) [**?**] on the latent TF of 50000 randomly generated Design Exemplars to project them onto a 2D plane. To avoid the thumbnails of the Design Exemplars occlude each other, we use a hierarchical display strategy. The 2D plane supports zoom in and zoom out like a canvas. In the default mode, all the design exemplars are represented as tiny white dots in the background to provide a rough context for the global scope. Only some white dots (the number is set to 64 during the experiments) are randomly sampled to display their corresponding thumbnail images. If the users do not find any interested Design Exemplars among the current displayed thumbnails, the interface can do re-sampling for them. If the users become interested in a certain area of the projected plane, they can drag a box around that area to drill into a local scope. New thumbnails will be displayed by sampling among the white dots in this local scope. User can cherry pick the Design Exemplars their like from Outline view to the archive area (see Fig. 1 area d).

5.2 Query the Differentiable Design Space

Although the outline view and the hierarchical display strategy provides users with a way to brows through the designs they might want, sometime users already have a target design in mind. In such case, users want to quick locate their target design in the design space, or at least know if the design space contain some similar design as a good start point for them. To address this, we provide the users with a query tool. At any time during exploration, the user can click the "FANGDAJING" in Fig. 1 area a to upload a query image. Then the Neural DVR Analyzer can analyze this query image and map it into a latent TF in the Differentiable Design Space. This latent TF will be projected onto the outline view. The Neural DVR Renderer will also take this latent TF and render it as a new Design Exemplar. **??**

6 Back to Explicit Transfer Function

6.1 The Differentiable DVR Renderer

To gain a better comprehension of our method, we can consider direct volume render process as a function $F(v, \tau)$, where v represents the viewport (i.e. camera position), τ represents the transfer function. Here we assume the volumetric data to be visualized is fixed, so it is not considered as a input of function F. The output of function $F(v, \tau)$ is the rendered image II of this volumetric data at viewport v with transfer function τ. $I(x, y)$ represents the color of the pixel at image coordinate (x, y).

For traditional direct volume rendering, the rendering process tries to simulate the physical process of light accumulation. Every pixel $I(x, y)$ of the image I is computed by the volume rendering integral [27]

$$I(x,y) = \int_a^b g(V(r(t)))e^{-\int_a^b \tau(V(r(u)))du} dt \tag{4}$$

where $r(t)$ is the position of the arc-length parameterized ray within the volumetric data, $V(x)$ represents volumetric data at position x, $g(v)$ and $\tau(v)$ are light emission and absorption terms. The exponential

term correspond to the attenuation of the light as the ray travels the space. Since this integral cannot be solved analytically for arbitrary transfer function, traditional DVR renderer uses the Riemann sum to approximate the light accumulation process:

$$C_{i+1} = C_i + (1-\alpha_i) * C \tag{5}$$

$$\alpha_{i+1} = \alpha_i + (1-\alpha_i) * \alpha \tag{6}$$

where C_i represents the accumulated light emission at sample position i, α_i represents the accumulated opacity at sample position i, C and α represent the light emission and opacity at the current sample, which is mapped by the transfer function.

6.2 The Differentiable DVR Renderer

6.3 The Reconstruction Process

Algorithm 2 Optimization Process for TF Reconstruction with Differentiable DVR Renderer

Parameters: stepsize ∇t, camera *cam*, Volume V, randomly initialized transfer function T

Input: u v the pixel position where to shoot the rays, Neural Rendered Image I_{pred}

repeat
  $color_i = 0$
  for $i = 0, \ldots, N-1$ **do**
    $x_i + i\Delta t\,\omega$ ▷ current position along the ray
    $d_i = f_{interpolate}(x_i, V)$ ▷ Trilinear interpolation
    $c_i = f_{TF}(d_i, T)$ ▷ TF evaluation
    $color_{i+1} = f_{blend}(color_i, c_i)$ ▷ blend of the sample
  end for
  $L_1 = \|color_N - I_{pred}(u,v)\|$ ▷ Loss function on the output image
  $\hat{T} = 0$ ▷ initialize the gradient of T
  $\hat{color_N} = \partial L_1(color_N)$
  for i = N - 1, ..., 0 **do**
    $\hat{color_i}, \hat{c_i} = \partial f_{blend}(color_i, c_i)^T, \hat{color_N}$
    $\hat{T} \pm \partial f_{TF}(d_i, T)^T \hat{c_i}$ ▷ renew the gradient of T
  end for
  $\hat{T} \pm \partial PriorSmooth(T)$ ▷ renew the gradient T for prior smooth
  $T \leftarrow Adam(\hat{T}, T, \alpha, \beta_1, \beta_2)$
until convergence

Output: T

7 Results

In this section, we will show the result of our approach from four aspects: (1) Our implementation details (Sec. 7.1). (2) Evaluation for the generation of the differentiable design space. (Sec. 7.2). (3) Experiments for differentiable design space exploration (Sec. 7.3) through case studies. (4) Experiments for the transfer function reconstruction(Sec. 7.4).

7.1 Implementation Details

The implementation of our approach can be divided to three components: the training of the neural DVR analyzer and neural DVR renderer for differentiable design space generation, the interactive applications for differentiable design space exploration, and the transfer function reconstruction. We discuss the implementation details of the three components in the following.

Our experiments used the following volume datasets: the Engine block, the CT-chest, the Torso Phantom, the Bonsai, and the Tooth**??**. The DVR renderer was adapted from the CUDA implementation of Weiss [43], which is based on basic emission-absorption model and Beer-Lambert blending. This DVR renderer can switch between two modes: (1) in non-differentiated mode, the renderer just perform like a traditional DVR renderer. (2) in differentiated mode, the renderer becomes a differentiable DVR renderer that allows to calculate the derivatives of transfer function parameters with respect to the rendering result. Since the tooth dataset is better classified with 2D transfer functions, we extended Weiss's renderer such that it can support 2 dimensional transfer functions for both non-differentiated mode and differentiated mode. We used this extended renderer with non-differentiated mode as a traditional DVR renderer to generate all the images for the neural network training. We used this extended renderer with differentiated mode as the differentiable DVR renderer for transfer function reconstruction. The neural DVR analyzer and neural DVR renderer was implemented in PyTorch**??** and trained on a symmetric multiprocessing node with 8 NVIDIA 3090 GPUs. The training time ranges from 10-30 hours based on the complexity of the dataset and the pre-defined transfer function design space. After training, the Neural DVR Analyzer took 10ms to inference, and the Neural DVR Renderer took 20ms to inference. The inference time was tested on a single NVIDIA 2080Ti GPU and was stable across all the datasets. All the images generated during our experiments were of 512*512 resolution. The interactive applications were implemented based on a web server/client framework. The interface was implemented with D3.js on the client side. All the rendering, network inference, and algorithm execution are done on the server side. The server contains an Intel Core i5-11400 CPU and an NVIDIA 2080Ti GPU.

7.2 Evaluation for Differentiable Design Space Generation

We first evaluate our generation of the differentiable design space. As described in Sec. 4, we use an "imitation task" as the auxiliary task to train the Neural DVR Analyzer for differentiable design space generation. We show the Neural DVR Analyzer's performance on this task across different volume datasets in Sec. 7.2.1. We then conduct a hyper-parameter study on the Neural DVR Analyzer in **??**. We evaluate the validity of the differentiable design space in Sec. 7.2.2.

7.2.1 Neural DVR Analyzer's Performance Across Datasets

We evaluate the performance of Neural DVR Analyzer for the "imitation task" on a separate set of 1,000 volume-rendered images, which are not seen during training phase. We use peak signal-to-noise ratio (PSNR), structural similarity index measure (SSIM) and color histogram's earth mover's distance (Color EMD) as our evaluation metrics. These three indicators complement each other: PSNR calculates the average mean squared error between the pixels of two images to measure their difference. A higher PSNR value indicates that the images being compared have a higher pixel-wise similarity. SSIM takes into account the structural information of images, such as edges, textures, and contrast, and measures the similarity between them based on these features. Color EMD can measures the similarity of color distribution between two images.

On the right of Tab. 1 we report the evaluation result of average PSNR, SSIM and EMD for all datasets. Those metrics indicates that overall our Neural DVR Analyzer can successfully analyse the given design exemplar and find a latent TF that can let the Neural Rendering Result be similar to the ground truth. The qualitative experimental results in Fig. 12 also confirm this.

Since human DVR experts usually like to find a suitable viewport for their transfer function design. We also conduct an experiment to explore how the viewport of the exemplar design image will affect the performance of our Neural DVR Analyzer. #TODO

7.2.2 Differentiable Design Space Validation

In Sec. 7.2.1 we verify that the Neural DVR Analyzer can effectively analyze the design exemplar images and successfully perform the "imitation task". However, we still do not know whether the Neural DVR Analyzer have successfully generated a differentiable design space or just memorize each image in the training set. Therefore, we (1) Feed every design exemplar image in the training set into the Neural DVR Analyzer to obtain the corresponding latent TF, we call them *train latent TF*. (2) Randomly sample a latent TF in the differentiable design space, we call it *sampled latent TF*. (3) Calculate the cosine distance and Euclidean distance between the *sampled latent TF* and every *train latent TF* (4) Find the top 5 *train latent TF* which have the closest distance with the *sampled latent TF*. (5) Use the Neural DVR Renderer

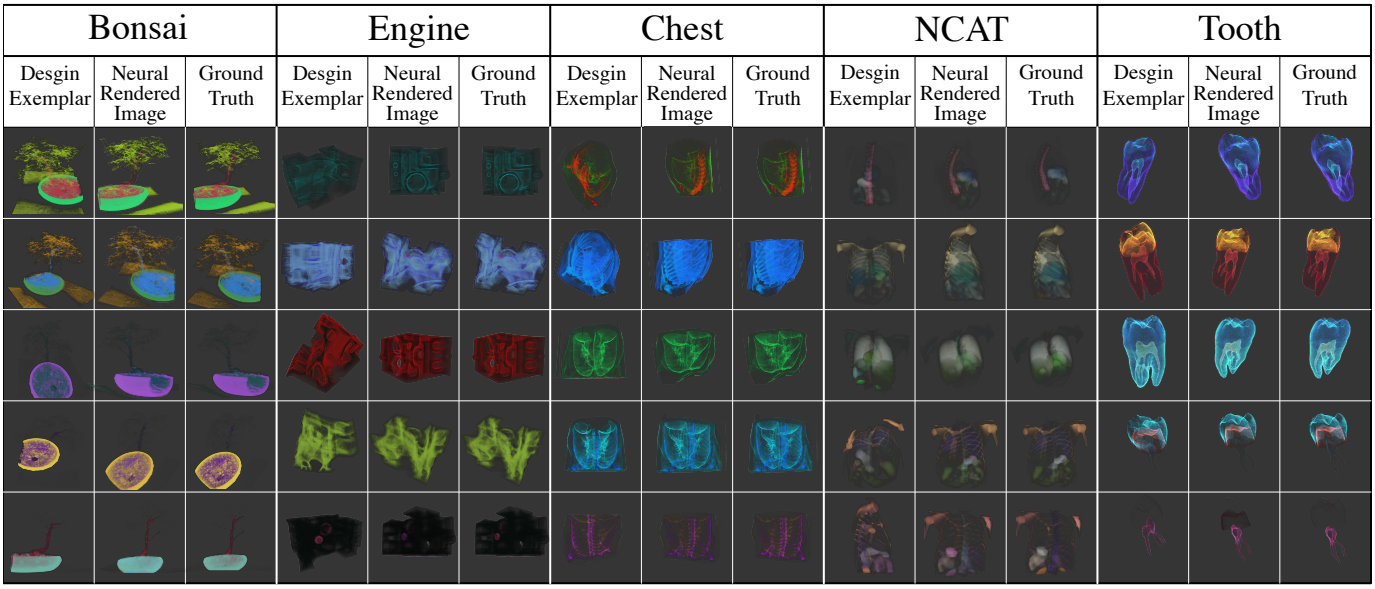

Fig. 11: result

Table 1: Evaluation Result of the Neural DVR Analyzer's Performance Across Datasets.

Dataset	Resolution	Precision	Size (MB)	Image SSIM	Image PSNR	Color EMD
Engine block	$256 \times 256 \times 110$	byte	6.87	0.983	35.0	0.00322
CT-chest	$384 \times 384 \times 240$	byte	33.7	0.990	36.7	0.00308
Torso Phantom	$256 \times 256 \times 256$	byte	16.0	0.983	30.1	0.00472
Bonsai	$256 \times 256 \times 256$	byte	16.0			
Tooth	$256 \times 256 \times 161$	byte	10.0	0.988	34.8	0.00297

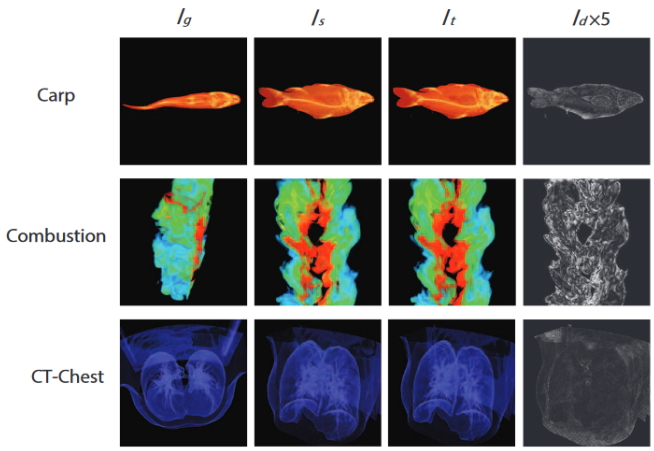

Fig. 12: We show qualitative results comparing synthesized images to ground truth volume renderings produced without illumination. The bottom row shows typical artifacts, such as incorrect color mapping and lack of detail preservation.

to visualize the *sampled latent TF* and the top 5 *train latent TF* for comparison.

Figure 14 shows the result.

7.3 Experiments for Differentiable Design Space Exploration

This section demonstrates how the users can explore the Differentiable Design Space with the set of visual interfaces we provided through case studies. We first show the exploration process of a user with a target goal on the CT-Chest data set, and then show the open-end exploration process of a user on the Tooth dataset. In both cases the users do not need to do any explicit transfer function edition.

7.3.1 Design the Target Transfer Function

7.3.2 Open End Transfer Function Design

7.4 Experiments for Transfer Functions Reconstruction

Weiss and Westermann [43] have proved that Differentiable DVR renderer can successfully perform transfer function reconstruction task. In their experiment, a traditional renderer with manually set transfer function is first used to render a reference DVR image. Then, they set the viewport of the Differentiable DVR renderer to be the same as the reference image and randomly initialize the transfer function of the Differentiable DVR renderer. Pixel-wise difference between the reference image and the image rendered by the Differentiable DVR renderer is used as the loss function to optimize the transfer function of the Differentiable DVR renderer. After convergence, the image rendered by the Differentiable DVR renderer will be close to the reference image.

The procedure of our reconstruction experiment is similar to Weiss and Westermann's. The key difference is that the reference is now rendered by our Neural DVR Renderer. Fig. 15 demonstrate the reconstruction result across different datasets. An interesting finding is that sometimes the Neural DVR renderer and the Differentiable DVR Renderer can complement each other: The rendering result of the Neural DVR renderer provide a rough design direction of the transfer function, but may create unreliable artifacts for details; The Differentiable DVR Renderer can optimize its transfer function toward the rough design direction provided by the Neural DVR Rendered result, but will remove the artifacts since the its rendering process is physically based (see the boxed area in Fig. 15).

8 Discussion

9 Conclusion

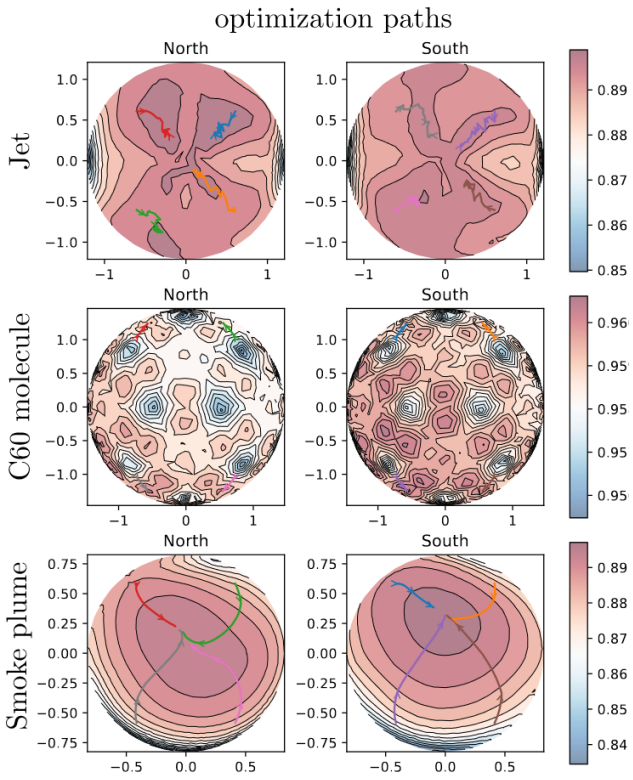

Fig. 13: Viewport Experiment.

References

[1] A. Abbasloo, V. Wiens, M. Hermann, and T. Schultz. Visualizing tensor normal distributions at multiple levels of detail. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):975--984, 2016. doi: 10.1109/TVCG.2015.2467031 2

[2] M. Berger, J. Li, and J. A. Levine. A generative model for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 25(4):1636--1650, 2019. doi: 10.1109/TVCG.2018.2816059 3, 4, 5, 6

[3] C. Correa and K.-L. Ma. Size-based transfer functions: A new volume exploration technique. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1380--1387, 2008. doi: 10.1109/TVCG.2008.162 2

[4] C. Correa and K.-L. Ma. The occlusion spectrum for volume classification and visualization. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1465--1472, 2009. doi: 10.1109/TVCG.2009.189 2

[5] C. D. Correa and K.-L. Ma. Visibility histograms and visibility-driven transfer functions. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):192--204, 2011. doi: 10.1109/TVCG.2010.35 2

[6] F. de Moura Pinto and C. M. Freitas. Design of multi-dimensional transfer functions using dimensional reduction. In *Proceedings of the 9th Joint Eurographics/IEEE VGTC conference on Visualization*, pp. 131--138, 2007. doi: 10.2312/VisSym/EuroVis07 2

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. arxiv 2014. *arXiv preprint arXiv:1406.2661*, 2014. 4, 5

[8] H. Guo, W. Li, and X. Yuan. Transfer function map. In *2014 IEEE Pacific Visualization Symposium*, pp. 262--266, 2014. doi: 10.1109/PacificVis.2014.24 1

[9] H. Guo, N. Mao, and X. Yuan. Wysiwyg (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2106--2114, 2011. doi: 10.1109/TVCG.2011.261 2

[10] M. Haidacher, S. Bruckner, A. Kanitsar, and M. E. Gröller. Information-based transfer functions for multimodal visualization. In *VCBM*, pp. 101--108. Citeseer, 2008. doi: 10.2312/ VCBM/VCBM08 2

[11] T. He, L. Hong, A. Kaufman, and H. Pfister. Generation of transfer functions with stochastic search techniques. In *Proceedings of Seventh Annual IEEE Visualization '96*, pp. 227--234, 1996. doi: 10.1109/VISUAL.1996.568113 2

[12] W. He, J. Wang, H. Guo, K.-C. Wang, H.-W. Shen, M. Raj, Y. S. G.

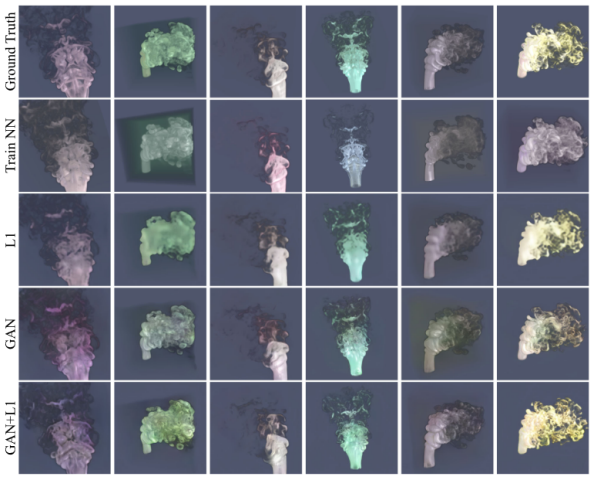

Fig. 14: We show qualitative results comparing synthesized images to ground truth volume renderings produced without illumination. The bottom row shows typical artifacts, such as incorrect color mapping and lack of detail preservation.

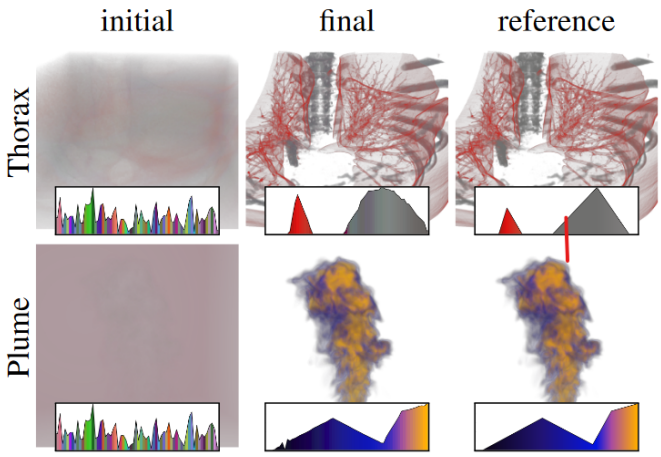

Fig. 15: We show qualitative results comparing synthesized images to ground truth volume renderings produced without illumination. The bottom row shows typical artifacts, such as incorrect color mapping and lack of detail preservation.

Nashed, and T. Peterka. Insitunet: Deep image synthesis for parameter space exploration of ensemble simulations. *IEEE Transactions on Visualization and Computer Graphics*, 26(1):23--33, 2020. doi: 10.1109/TVCG.2019.2934312 3, 4

[13] F. Hong, C. Liu, and X. Yuan. Dnn-volvis: Interactive volume visualization supported by deep neural network. In *2019 IEEE Pacific Visualization Symposium (PacificVis)*, pp. 282--291, 2019. doi: 10.1109/PacificVis.2019.00041 3, 4, 5

[14] K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251--257, 1991. 3

[15] C. Y. Ip, A. Varshney, and J. JaJa. Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2355--2363, 2012. doi: 10.1109/TVCG.2012.231 2

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. doi: 10.48550/arXiv.1611.07004 4, 5

[17] D. Jönsson, M. Falk, and A. Ynnerman. Intuitive exploration of volumetric data using dynamic galleries. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):896--905, 2016. doi: 10.1109/TVCG.2015.2467294 1, 3

[18] H. S. Kim, J. P. Schulze, A. C. Cone, G. E. Sosinsky, and M. E. Martone. Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets. *Information visualization*, 9(3):167--180, 2010. doi: 10.1057/ivs.2010.6 2

[19] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller. Curvature-based transfer functions for direct volume rendering: methods and applications. In *IEEE Visualization, 2003. VIS 2003.*, pp. 513--520, 2003. doi: 10.1109/VISUAL.2003.1250414 2

[20] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270--285, 2002. doi: 10.1109/TVCG.2002.1021579 2

[21] J. Kniss, R. Van Uitert, A. Stephens, G.-S. Li, T. Tasdizen, and C. Hansen. Statistically quantitative volume visualization. In *VIS 05. IEEE Visualization, 2005.*, pp. 287--294, 2005. doi: 10.1109/VISUAL.2005.1532807 2

[22] P. Ljung, J. Krüger, E. Groller, M. Hadwiger, C. D. Hansen, and A. Ynnerman. State of the art in transfer functions for direct volume rendering. In *Computer Graphics Forum*, vol. 35, pp. 669--691. Wiley Online Library, 2016. doi: 10.1111/cgf.12934 2

[23] C. Lundstrom, P. Ljung, and A. Ynnerman. Local histograms for design of transfer functions in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1570--1579, 2006. doi: 10.1109/TVCG.2006.100 2

[24] K.-L. Ma. Image graphs-a novel approach to visual data exploration. In *Proceedings Visualization '99 (Cat. No.99CB37067)*, pp. 81--88, 1999. doi: 10.1109/VISUAL.1999.809871 2

[25] R. Maciejewski, I. Woo, W. Chen, and D. Ebert. Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1473--1480, 2009. doi: 10.1109/TVCG.2009.185 2

[26] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, et al. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 389--400, 1997. doi: 10.1145/258734.258887 1, 2, 4, 5, 6

[27] N. Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99--108, 1995. doi: 10.1109/2945.468400 6

[28] H. Pfister, B. Lorensen, C. Bajaj, G. Kindlmann, W. Schroeder, L. Avila, K. Raghu, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21(3):16--22, 2001. doi: 10.1109/38.920623 2

[29] F. D. M. Pinto and C. M. Freitas. Two-level interaction transfer function design combining boundary emphasis, manual specification and evolutive generation. In *2006 19th Brazilian Symposium on Computer Graphics and Image Processing*, pp. 281--288, 2006. doi: 10.1109/SIBGRAPI.2006.451, 2

[30] F. d. M. Pinto and C. M. Freitas. Volume visualization and exploration through flexible transfer function design. *Computers & Graphics*, 32(4):420--429, 2008. doi: 10.1016/j.cag.2008.04.004 1, 2

[31] J. Prauchner, C. Freitas, and J. Comba. Two-level interaction approach for transfer function specification. In *XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'05)*, pp. 265--272, 2005. doi: 10.1109/SIBGRAPI.2005.52 1, 2

[32] J. Prauchner, C. Freitas, and J. Comba. Two-level interaction approach for transfer function specification. In *XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'05)*, pp. 265--272, 2005. doi: 10.1109/SIBGRAPI.2005.52 1, 2

[33] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention--MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pp. 234--241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28 5

[34] C. R. Salama, M. Keller, and P. Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1021--1028, 2006. doi: 10.1109/TVCG.2006.148 2

[35] F.-Y. Tzeng and K.-L. Ma. A cluster-space visual interface for arbitrary dimensional classification of volume data. 2004. 2

[36] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The

missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. doi: 10.48550/arXiv.1607.08022 5

[37] C. Wang and J. Han. Dl4scivis: A state-of-the-art survey on deep learning for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, pp. 1--1, 2022. doi: 10.1109/TVCG.2022.3167896 3

[38] Y. Wang, W. Chen, J. Zhang, T. Dong, G. Shan, and X. Chi. Efficient volume exploration using the gaussian mixture model. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1560--1573, 2011. doi: 10.1109/TVCG.2011.97 2

[39] Y. Wang, J. Zhang, D. J. Lehmann, H. Theisel, and X. Chi. Automating transfer function design with valley cell-based clustering of 2d density plots. In *Computer Graphics Forum*, vol. 31, pp. 1295--1304. Wiley Online Library, 2012. doi: 10.1111/j.1467-8659.2012.03122.x 2

[40] J. Weiss and N. Navab. Deep direct volume rendering: Learning visual feature mappings from exemplary images. *arXiv preprint arXiv:2106.05429*, 2021. 3

[41] S. Weiss, M. Chu, N. Thuerey, and R. Westermann. Volumetric isosurface rendering with deep learning-based super-resolution. *IEEE Transactions on Visualization and Computer Graphics*, 27(6):3064--3078, 2021. doi: 10.1109/TVCG.2019.2956697 3

[42] S. Weiss, M. Işık, J. Thies, and R. Westermann. Learning adaptive sampling and reconstruction for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(7):2654--2667, 2022. doi: 10.1109/TVCG.2020.3039340 3

[43] S. Weiss and R. Westermann. Differentiable direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):562--572, 2022. doi: 10.1109/TVCG.2021.3114769 3, 7, 9

[44] H.-C. Wong, U.-H. Wong, and Z. Tang. Direct volume rendering by transfer function morphing. In *2009 7th International Conference on Information, Communications and Signal Processing (ICICS)*, pp. 1--4, 2009. doi: 10.1109/ICICS.2009.5397587 2

[45] Y. Wu and H. Qu. Interactive transfer function design based on editing direct volume rendered images. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1027--1040, 2007. doi: 10.1109/TVCG.2007.1051 2

[46] X. Zhao and A. Kaufman. Multi-dimensional Reduction and Transfer Function Design using Parallel Coordinates. In R. Westermann and G. Kindlmann, eds., *IEEE/ EG Symposium on Volume Graphics*. The Eurographics Association, 2010. doi: 10.2312/VG/VG10/069-076 2