

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Image-based TF colorization with CNN for direct volume rendering

SEOKYEON KIM¹, YUN JANG¹, AND SEUNG-EOCK KIM²

¹Department of Computer Engineering and Convergence Engineering for Intelligent Drone, Sejong University (e-mail:ksy0586@gmail.com, jangy@sejong.edu)

²Department of Civil and Environmental Engineering, Sejong University (e-mail:sekim@sejong.ac.kr)

Corresponding author: Yun Jang (e-mail: jangy@sejong.edu).

ABSTRACT In the direct volume rendering (DVR), it often takes a long time for a novice to manipulate the transfer function (TF) and analyze the volume data. To lessen the difficulty in volume rendering, several researchers have developed deep learning techniques. However, the existing techniques are not easy to apply directly to existing DVR pipelines. In this study, we propose an image-based TF colorization with CNN to automatically generate a direct volume rendering image (DVRI) similar to a target image. Our system includes CNN model training, TF labeling, image-based TF generation, and volume rendering by matching the target image. We introduce a technique for training CNN and labeling the TF with images similar to the input volume dataset. Moreover, we extract the primary colors from the target image according to the labels classified with the CNN model. We render the volume data with the TF to produce the DVRI reproducing the prominent colors in the target image.

INDEX TERMS Volume Rendering, CNN, TF colorization

I. INTRODUCTION

Direct volume rendering (DVR) is a technique to visualize 3D volume data by projecting it into a 2D plane. In general, to acquire direct volume-rendered images (DVRI) from discrete volume data, a camera is placed in a space where 3D volume data exists, and sets of color and opacity are assigned to all voxels. Then, the renderer visualizes the 3D volume data with the camera and voxel information by applying direct volume rendering techniques such as ray casting. In this process, a transfer function (TF) is manipulated to determine the optical characteristics of voxels within the volume renderer. The user then interacts with the TF to adjust the color and transparency of a voxel. Properly manipulated TF can efficiently convey data type and domain characteristics and help the user explore the area of interest in the volume data.

The Intensity-Gradient Magnitude (I-GM) TF is the most commonly used two-dimensional TF. A two-dimensional histogram is generated with volume intensity in the x-axis and its gradient magnitude in the y-axis for the I-GM TF. The volume intensity represents the medium characteristics, and its value increases as the medium becomes more firm. The gradient magnitude reveals the medium change. In general, since the same medium has a relatively uniform intensity, the gradient magnitude is low within the same material.

Due to these properties of intensity and gradient magnitude, the I-GM TF contains arcuate patterns linking the smallest and largest intensities within the intensity range of the same medium. Therefore, to manipulate the I-GM TF, users identify arch segments, explore the material boundaries, and assign optical properties in the volume. However, it often happens that the arches that define material boundaries in 2D TFs overlap each other. Also, in some cases, this overlap makes it difficult to separate the medium within the I-GM TF.

Selecting the area of interest while adjusting the TF for the volume data requires understanding and know-how of the TF, and it takes considerable time for the user to specify the appropriate TF setting. Therefore, until recently, studies have been conducted to design the interaction with TFs utilizing various attributes obtained directly from volumetric data [1], [2]. Among the studies on TF, Maciejewski et al. [3] propose a technique to visualize the relationships between volumetric data attributes and rendering in the attribute space. The proposed model bundles voxels into several clusters based on the user-specified attribute space. When rendering with the semi-automatically generated TF using the attribute space, a DVRI is obtained for each cluster, which represents a characteristic of the cluster. In this work, we propose a technique to acquire DVRI using clustered TF in the attribute space, which is inspired by the study of Maciejewski et al. [3] and classify

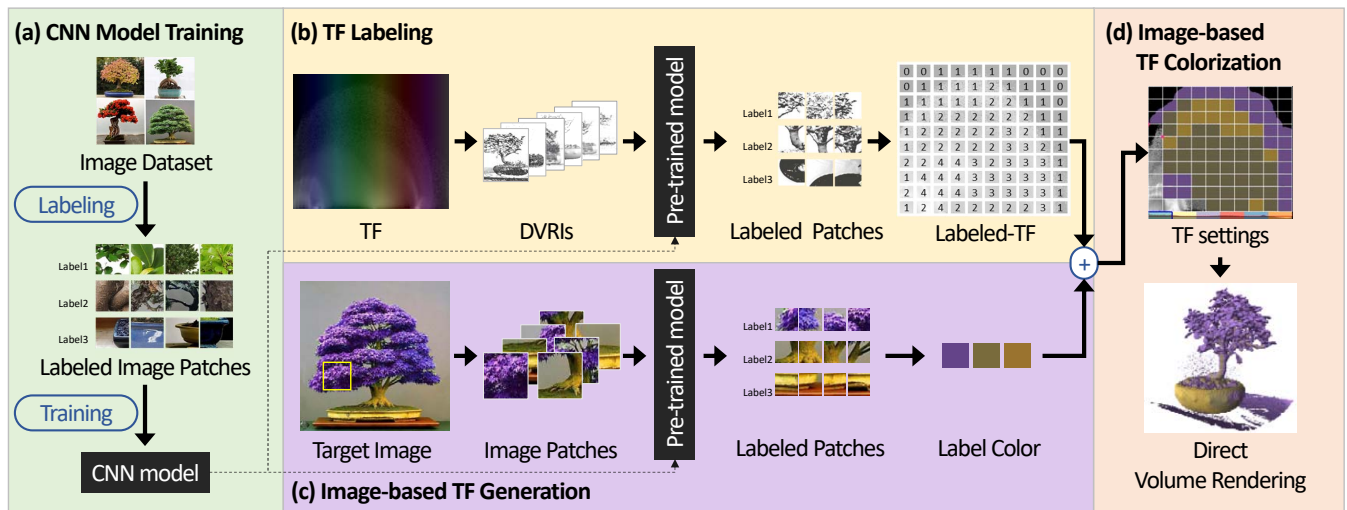


FIGURE 1. The image-based TF colorization pipeline. Our system incorporates a CNN model training, TF labeling, image-based TF generation, and volume rendering. We classify training datasets from raw images and train our proposed network (CNN model) in (a). (b) shows how we generate the Labeled-TF with the CNN model, and (c) presents the way we extract label colors from the target image with the CNN model. The TF is created by mapping the label color to the Labeled-TF in (d). We use the color-mapped TF to produce a volume rendering image similar to the target image. Note that the Pre-trained model for the TF labeling and TF generation is the CNN model trained in (a).

the obtained DVRI with CNN.

The up-to-date deep learning techniques have been applied to many research fields as a means of generating a model by combining neural network architectures with data. Several volume rendering techniques employ the deep learning techniques, including volume segmentation [4], [5], viewpoint estimation [6], transfer function [7], lighting [8], and quality improvement [9]. These studies utilize Convolution Neural Network (CNN) and Generative Adversarial Net (GAN). However, the techniques using CNN are limited only to indirect volume rendering (IVR), which are difficult to be extended to DVR, and the studies using GAN require long training time and data distortion might happen.

In this work, we propose an image-based TF using CNN to create a DVRI similar to the target image. Figure 1 illustrates our approach. We employ the target image as an approach to overcome the difficulty in the TF manipulation in the DVR. The target image refers to the image associated with the volume dataset that we want to produce and includes not only the DVRI but also the actual images such as photos. The user who is not familiar with volume rendering first reviews the name of the volume dataset and infers the data shape before starting the volume rendering. Then, the user observes the rough shape of the volume while adjusting the TF within the volume renderer. Therefore, the user sets the target image, which can be obtained from anywhere with keywords guessed from the volume dataset name and the rough shape of the volume.

The CNN model training in Figure 1 (a) is a preparation step for applying CNN to the volume rendering, which classifies training datasets and trains our proposed network. We gather necessary image datasets by crawling on the internet according to the domain of volume data and then label the image datasets. We train the CNN network with the labeled

training image datasets. In the TF labeling step, as shown in Figure 1 (b), the 2D transfer function consisting of the intensity value and gradient magnitude (I-GM TF) is divided into grids, and DVRI corresponding to each grid is rendered. We input the rendered DVRI into the pre-trained model that is the CNN model trained in (a) to determine the label of the DVRI and assign the label to the TF grid label. To coordinate the colors of our final DVRI, we extract the label colors from the target image, which is mapped to the Labeled-TF to build a TF as presented in Figure 1 (c) and (d). The TF automatically visualizes the DVRI matching the target image. Note that we utilize only one CNN model in this study, and the Pre-trained models in (b) and (c) are the CNN model trained in (a).

The main contributions of this study are as follows.

- We propose an image-based volume rendering technique that imitates the target image.
- We apply a CNN model to determine the labels for TF grids.
- We extract the primary colors from the target image with the CNN model.

II. RELATED WORK

Direct volume rendering (DVR) is a rendering technique by specifying the color and opacity of each voxel in the transfer function (TF). Since setting appropriate TF in DVR increases understanding of volume data and improves visual cue, many TF techniques have been studied to set proper TFs in the volume data domain. Drebin et al. [10] have proposed a direct volume rendering (DVR) with a 1D histogram TF. He et al. [11] have used the 1D TF which classifies voxels based on their scalar intensity values. and Li et al. [12] have introduced the classification of volume rendering by applying a 1D TF to an automobile CT image. The 1D TF

is the most common transfer function technique employed in various systems [13]–[15]. However, when trying to separate voxels with different features, there are many cases where the intensity ranges of voxels overlap. Therefore, it is limited to distinguish and identify features with only 1D TF [16].

For the overcome of limitations by the 1D TF, 2D or higher dimensional TFs have been proposed by employing additional parameters including scalar value and gradient magnitude [17], color distance gradient [18], curvature [19]–[21], spatial distances to boundaries [22], occlusion spectrum [23], and view-dependent occlusion [24]. The high dimensional TFs allow us to distinguish the volume features more efficiently and achieve better volume rendering images compared to the 1D TFs. As more parameters are added to various TFs, it becomes possible to refine the volume data features. Typical multi-dimensional transfer functions (MDTFs) include the predefined probability distributions [25], LH domain for material boundary identification [26], peak values using RBF network [27], non-parametric kernel density estimation [28], statistical TF [29], attribute space [3], and stochastic neighbor embedding-based dimension reduction [30]. In these studies, multi-dimensional domain properties are employed to the MDTFs, which increases the likelihood of separating features within the volume data and enhances the visual cue in the rendering image. However, the MDTFs require much effort to designate suitable TFs embedding domain characteristics because of the complexity of user interactions as the number of attributes increases.

Unlike the techniques that directly increase the dimension of TF, some studies improve TF manipulation or introduce new TF manipulation techniques. Castro *et al.* [31] have presented a TF classification technique by coupling metadata with 1D TF to select high-level components, including bones, brains, and muscles. Kniss *et al.* [17] have introduced a set of widgets and interaction tools for the MDTF. Moreover, various interaction techniques have been proposed. Especially, parameterized primitive sets [32], sorted histogram [33], fusing multiple features [34], textural metrics [35], anatomical structure [36], feature size-based TF [37], parallel coordinates plot [38]–[40], WYSIWYG [41], hierarchical modified dendrogram [42], attribute space exploration [3], and spatial connectivity of boundaries [43] have been presented.

Instead of manipulating TFs, TFs are also automatically generated in DVR. Fujishiro *et al.* [44] have proposed an automated TF utilizing the Reeb graph topology analysis. Sereda *et al.* [45] have introduced an automatic classification model by computing the similarity between the LH space and volume. Pfaffelmoser *et al.* [46] have visualized the uncertainty of the volume surface with the spatial deviation of the surface in the automatically selected TF. Bramon *et al.* [47] have presented an information weight model to which the information theory is applied to define multiple TFs automatically. Ma and Entezari [48] have shown a technique to identify FOI (Feature of Intensity) utilizing the cell-based feature similarity and automatically generate the TF based on

the selected feature.

Some studies enhance the performance of data analysis by employing machine learning techniques for the transfer function. De Moura Pinto and Freitas [49] have proposed an MDTF design by reducing the dimension of TF space with the self-organizing map. Soundararajan and Schultz [50] have compared five supervised classification techniques, including Gaussian Naive Bayes, k Nearest Neighbor, Support Vector Machines, Neural Networks, and Random Forests for the stochastic TF design. Quan *et al.* [51] have introduced a high-dimensional feature model using hierarchical multi-scale 3D convolutional sparse coding as a voxel classification technique.

Neural networks facilitate the retrieval of solutions to complex problems by repeatedly fitting the networks to data through multiple neural network structures. Convolutional Neural Network (CNN) [52] and Generative Adversarial Nets (GAN) [53] have been employed to enhance the TF manipulations. Cheng *et al.* [4] have used CNN for the volume visualization of complex high-dimensional information as the TF study, where they have divided the data into 65x65 patches and trained the CNN with the ADADELTA solver [54]. 200-dimensional features are extracted, and the volume is visualized with the conventional marching cube algorithm for specific high-dimensional feature values. However, since this approach is limited to indirect volume rendering by marching cube, it is not easy to extend this technique to the transfer function in the DVR. Berger *et al.* [7] have introduced a volume rendering technique utilizing GAN to compute a model from a large set of volume rendering images at specific viewpoints and certain transfer functions for the opacity and color. Hong *et al.* [55] have trained a deep neural network combined with GAN and CNN using volume rendering images. Their system synthesizes the volume rendering images with goal-effect rendering. This technique enables volume data browsing without an explicit transfer function.

Although there have been many machine learning techniques for the volume rendering, we still find difficulty in producing the rendering images from volume data automatically according to the rendering preference. Our work in this paper combines the deep learning technique, TF manipulation, and user preference setting for DVR as an automatic DVR pipeline.

III. CNN MODEL TRAINING

CNN model training is a foundation step for applying CNN to the volume rendering. In this step, we label training datasets and train the CNN network. Figure 2 presents the training process for CNN. We build image datasets by crawling images on the web that are similar in the shape and context to the volume data. We divide the image datasets into smaller image patches and label them to generate labeled image patches. The system then trains our CNN network with the generated labeled image patches. The trained CNN is used to classify direct volume rendering images.



FIGURE 2. The training process for CNN. We split crawled images into smaller patch images and label them to generate training datasets. We train the CNN with the generated training datasets.

A. IMAGE DATA COLLECTING

In general, labeled data are required to complete supervised learning. In this section, we introduce how we build labeled image datasets representing the shape and context of the volume data. When rendering a volume dataset, we usually render the overall shape of the data and manipulate the TF to color the voxels. During this procedure, we deduce keywords related to the shape and name of the volume data. We also determine labels with the keywords that represent prominent features in the volume rendering images. The labels are fictitious names assigned only by the overall shapes of the volume rendering image. For the bonsai dataset, for example, we begin with *leaf*, *stem*, *soil*, and *pot* as the labels. Therefore, we create the keywords as *bonsai*, *bonsai leaf*, *bonsai stem*, *bonsai soil*, and *bonsai flowerpot* and crawl the image datasets on the internet with these keywords. However, the crawled image datasets include images that are inappropriate for the training, such as graphic edited images, illustrated images, images with watermarks, and images irrelevant to the context. Therefore, we create a set of raw images after removing images that are inappropriate for the training. The raw images contain various objects together. Therefore, in order to classify the raw images into the labels, we divide the raw images into 32×32 and 64×64 patches and classify the patches into the labels. If the patch size is smaller than 32×32 , there are too few features, which may interfere with training. If it is larger than 64×64 , the patch may contain multiple features.

B. LABELED DATASET GENERATION FOR CNN

For the CNN model training, we generate the training datasets. The objective of this paper is to label the transfer function (TF) using CNN to generate rendering images similar to the target image colors. Although the main goal of this paper is not the image labeling, the labeled dataset for the training is essential in the deep learning research. Many public datasets are intended to provide labels as creatures or objects, such as a human, dog, cat, or car. However, for the volume rendering, it is necessary to label the data from a more anatomical point of view. For example, it is necessary to label a plant image by dividing it into leaves, stems, and pots in the image, not the plant itself. Since it is not easy to obtain labeled volume image data, we divide the images into smaller patches and classify them manually to evaluate our study. In order to generate a labeled dataset, we develop a labeling system, as shown in Figure 3. We have

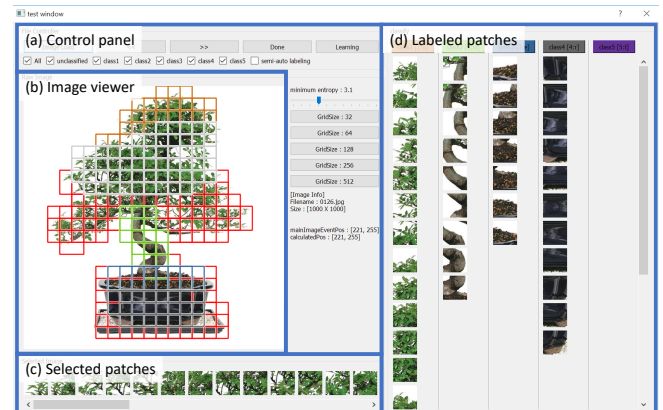


FIGURE 3. Our Image labeling system. (a) is a control panel that enables us to modify parameters for labeling. (b) is an image viewer that shows image patches and colored label boxes overlapped on the patches. (c) displays patches for labeling selected in (b), and (d) shows labeled patches listed by labels.

built the labeling system with Python, PyQt5, and Keras, which divides the meaningful parts of an image into patches and enables us to assign labels to the patches simply by mouse clicking & dragging. Figure 3 (a) is a control panel that enables us to adjust parameters for labeling, such as an image file, patch size, and entropy threshold. (b) is an image viewer that shows image patches and the labels of the patches in a box. (c) shows the patches requiring labeling selected in (b). Classified patches are arranged by label in (d). The user loads an image and adjusts the parameters in the control panel. Then the user selects the patches that are considered to correspond to a specific label in the image viewer by clicking & dragging. Selected patches are displayed in (c). The user explores the selected patches to see if the selection is correct and removes mistakenly selected patches. Then, when the user assigns a label to the selected patches, the patches are listed in (d). The user can examine patches vertically in (d) and remove the incorrectly labeled patches. The final labeled patches are stored in the computer and used for the CNN training. Labeling a single image takes about 15 seconds on average.

C. CNN ARCHITECTURE AND TRAINING

Convolutional neural network (CNN) is a class of deep learning mainly utilized to recognize images. As a CNN architecture, many well-designed networks such as *AlexNet* [56], *VGGNet* [57], *GoogLeNet* [58], and *ResNet* [59] have been studied. In particular, the *VGGNet* is applied to the volume rendering by Shi and Tao [6] to demonstrate its superiority when measuring the image recognition accuracy. The *VGGNet* architecture increases the layer depth by overlapping the 3×3 convolution layer multiple times. Since it has fewer parameters than large-sized filter like 5×5 or 7×7 , the *VGGNet* increases computational efficiency and accuracy. In this work, we design the CNN model by adjusting the layer size by referring to the VGG architecture.

Figure 4 illustrates the CNN architecture applied in this

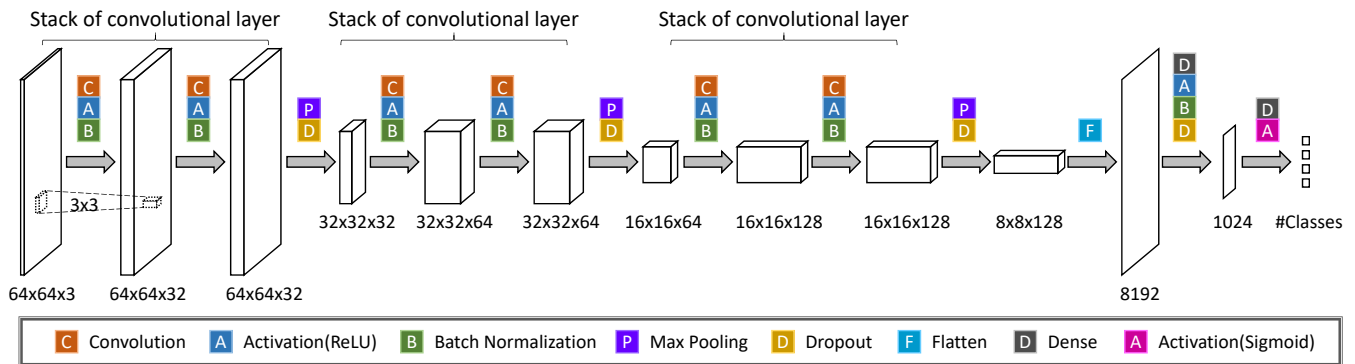


FIGURE 4. The CNN architecture. We modify the *VGGNet* architecture and apply the optimized structure for the 64×64 patch images to the CNN model. The CNN architecture trains data features through 3 stacks of convolutional layers.

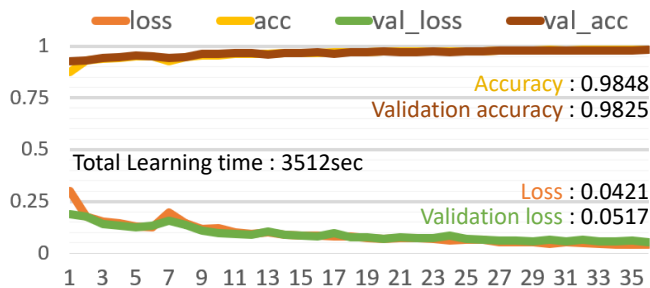


FIGURE 5. The performance of CNN model. As each epoch progresses, accuracy and validation accuracy increase, and loss and validation loss decrease. The validation accuracy of our model shows 0.9825.

study. We modify the architecture and parameters of the *VGGNet*. We utilize the previously labeled 32×32 or 64×64 patch images for the input to the CNN. The 32×32 patch is resized to a 64×64 patch to apply the same architecture. The *ReLU* is used as an activation function, and the batch normalization is performed after every convolution and activation to avoid the data skewness. Moreover, the system gradually reduces the neurons with the 2×2 maxPooling after each stack of convolutional layer is performed. Finally, the neurons are flattened in one dimension and train the network according to the number of labels in the dense layer and the sigmoid activation layer. The trained network labels the input image patches.

We train our deep learning network using labeled patches. However, if the number of images is different for each label, the deep learning network tends to be biased. Therefore, we randomly sample the labeled patches with the same number to generate a representative set of each label. Then, the patches of each label are randomly divided into a training set and a test set to train the deep learning network. Note that 70% and 30% of the patches are used as the training set and the test set, respectively.

Figure 5 presents the performance of our CNN model. We check the performance of training a CNN model with the labeled data generated from 500 bonsai images. Figure 5 shows the loss, accuracy, validation loss, and validation accuracy during 35 epochs. The training time is 3,512 seconds and the

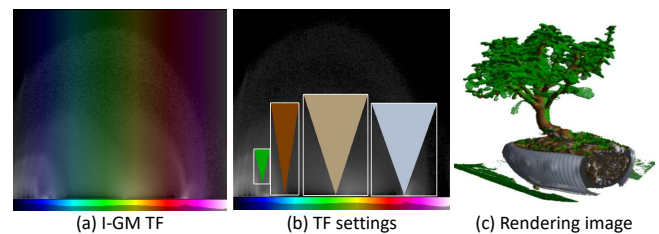


FIGURE 6. The 2D transfer function based on the intensity value and gradient magnitude (I-GM TF). (a) presents I-GM TF of the bonsai dataset. The user adjusts the TF colors in (b), and renders the data with the colors. (c) displays the direct volume rendering image (DVRI) with the TF in (b).

accuracy is 98.48 %.

IV. VOLUME RENDERING TECHNIQUES TO REFLECT TARGET IMAGE'S COLOR

In volume rendering, the transfer function setting affects the rendering quality, such as the visibility of valuable information within the volume data. In general, to acquire direct volume-rendered images (DVRI) from discrete volume data, a camera is placed in a space where 3D volume data exists, and sets of color and opacity are assigned to all voxels. Then, the renderer visualizes the 3D volume data with the camera and voxel information by applying direct volume rendering techniques such as ray casting. In this process, a transfer function (TF) is manipulated to determine the optical characteristics of voxels within the volume renderer. The user then interacts with the TF to adjust the color and transparency of a voxel. Properly manipulated TF can efficiently convey data type and domain characteristics and help the user explore the area of interest in the volume data. Figure 6 (a) presents the 2D transfer function based on the intensity value and gradient magnitude (I-GM TF). In order to visualize the volume data, a user adjusts the TF colors, as shown in Figure 6 (b), and render the data with the colors. Figure 6 (c) displays the direct volume rendering image (DVRI) with the TF in (b). Although the proper TF allows the user to achieve the desired visualization, the user still needs to adjust the TF repeatedly to explore the volume data until the best TF is discovered. In this study, we propose a novel volume rendering system

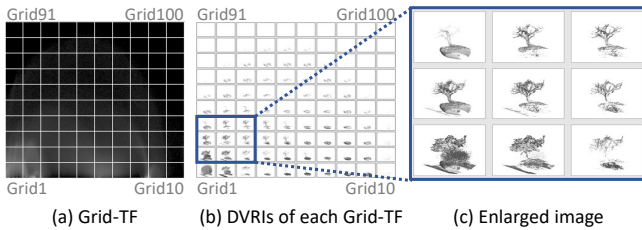


FIGURE 7. Grid-TF and rendering images. (a) presents the 10×10 Grid-TF of the bonsai volume dataset. (b) shows DVRI of each grid of the Grid-TF, which we call G-DVRIs in this paper. (c) is an enlarged view of G-DVRIs generated with Grid11, 12, 13, 21, 22, 23, 31, 32, and 33. Note that our approach supports an " $N \times N$ " grid, not limited to 10×10 .

resembling the target image to shorten the learning curve for the TF setting.

A. DVRI DEFINITION FOR TF LABELING

In order to set the TF automatically, we explore and label the TF features. For the TF labeling, we first subdivide the TF to see which volume region each sub-TF represents. As the most straightforward approach, the I-GM TF is divided into $N \times N$ grids to build the Grid-TF, and we examine the DVRI of each grid in the Grid-TF. Figure 7 (a), as an example, presents the 10×10 Grid-TF of bonsai volume dataset. Figure 7 (b) shows DVRI of each grid of the Grid-TF, which we call G-DVRIs. Figure 7 (c) is an enlarged view of DVRI of grids generated with the Grid 11, 12, 13, 21, 22, 23, 31, 32, and 33. Since the color information in the rendered image might influence the classification in the CNN, we assign only the white to each grid in the Grid-TF and apply the Phong shading for the rendering. As presented in Figure 7 (b) and (c), it is seen that each grid of the Grid-TF represents a different volume having a unique material or feature in the volume data. For the automatic classification of the G-DVRIs, we label the G-DVRIs with the trained CNN model. However, the CNN model classifies most of G-DVRIs as either unlabeled or incorrectly labeled. As observed in Figure 7 (c), the voxel sparsity of the DVRI hinders the classifications because the sparse image feature points in the DVRI are not well recognized with the convolutional feature of CNN model. In order to prevent a sparse DVRI, we may reduce the number of grids, employ image processing techniques such as blurring or utilize a clustered TF by volume features. Reducing the number of grids does not fit our approach because it is not the goal of subdividing the TF. Also, the blurring process may cause data loss or distortion. Therefore, we employ the TF [3] in the attribute space for the clustered TF, as a way to resolve this challenge.

Figure 8 (a) presents a TF classified into six abstract space classes, which we call AS-TF. In the figure, we set Grid1, Grid2, and Grid11 transparent to remove apparent noise in the data. Figure 8 (b) shows DVRI for all AS-TF classes, which we call AS-DVRIs. Since AS-DVRIs have smoother volume surfaces than G-DVRIs, AS-DVRIs tend to preserve the features of volume objects better than G-DVRIs. Therefore, the AS-DVRI is more suitable for labeling

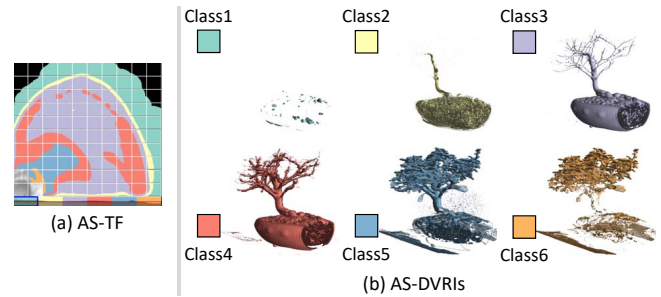


FIGURE 8. AS-TF and rendering images. (a) presents the TF classified into six abstract space classes, which we call AS-TF. (b) shows DVRI for all six AS-TF classes, which we call AS-DVRIs.

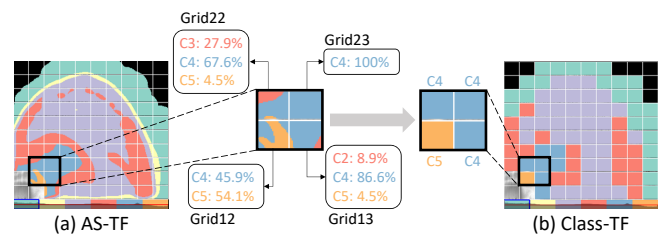


FIGURE 9. The process of converting AS-TF to Class-TF. (a) is the AS-TF of the bonsai dataset, and (b) shows the Class-TF. In Grid22, for example, Class3, Class4, and Class5 occupy 27.9 %, 67.6 %, and 4.5 %, respectively. We set the representative class of Grid22 as Class4 that occupies the most area.

with the CNN model. However, there might be cases where multiple AS-TF classes exist simultaneously in a specific grid. In this multi-class grid, it is difficult to specify the class of the grid only with the AS-DVRI. Since the our model performs based on an image, it is possible to extract and label patches of DVRI, but it is difficult to trace the labeled patches back to the TF space.

To overcome the difficulty, we propose a Class-TF, as shown in Figure 9, to suitably label each grid in the Grid-TF by considering the characteristics of the AS-TF. Figure 9 illustrates how to build the Class-TF from the AS-TF. Figure 9 (a) is the AS-TF of the bonsai dataset, and Figure 9 (b) shows the Class-TF. We determine the class that occupies the most area among AS-TF classes in a grid as the representative class of the grid. In Grid22, for example, class3, class4, and class5 occupy 27.9 %, 67.6 %, and 4.5 %, respectively. We,

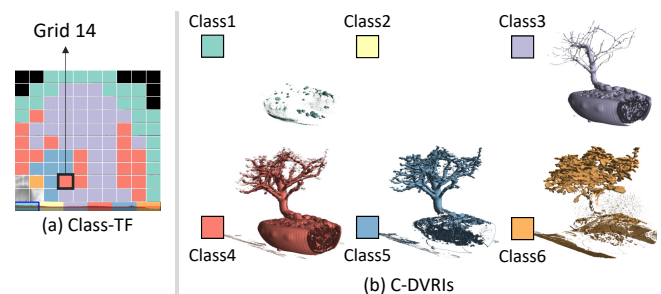


FIGURE 10. Class-TF and rendering images. The Class-TF is presented in (a), and the DVRI for classes are shown in (b), which we call C-DVRIs.

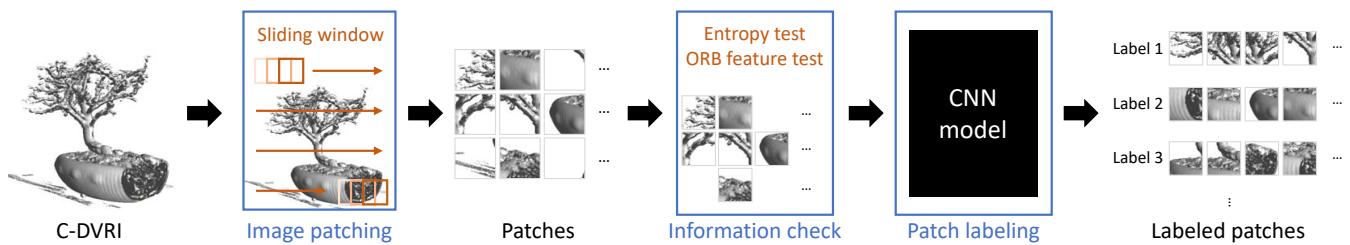


FIGURE 11. The procedure to obtain labeled patches from the C-DVRI. We divide C-DVRI into patches, perform information check, and label patches with CNN model.

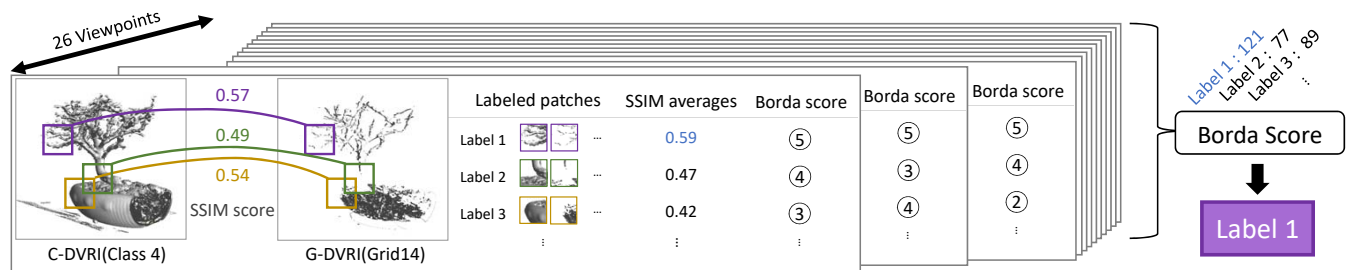


FIGURE 12. The procedure of TF labeling. We compare the SSIMs of C-DVRI and G-DVRI and perform the voting with Borda count scores to assign the label to the grid in the Grid-TF.

therefore, set the representative class of Grid22 as class4. After finding all the representative classes in all grids, the Class-TF presented in Figure 9 (b) is obtained. Figure 10 presents the Class-TF and DVRI for all classes, which we call C-DVRIs. If we compare the AS-DVRIs in Figure 8 (b) and the C-DVRIs in Figure 10 (b), it is observed that the image for the class 2 is not produced, which can be interpreted as a missing class because the class ratio within the Grid-TF is too small to become a representative class. Compared to the AS-TFs separated by features in the attribute space, the Class-TF may suffer the loss of attribute space classes or the loss of the fine separations. However, most C-DVRIs and AS-DVRIs are similar, and the Class-TF at least avoids the sparsity seen in the G-DVRIs. Besides, each class in the Class-TF includes multiple grids in the Grid-TF, which allows us to label each grid in the Grid-TF. In Figure 10 (a), for example, Grid3, 13, 22, 23, 24, 33, and 34 in the Grid-TF are included in Class5 in the Class-TF.

The C-DVRIs generated by the Class-TF are divided into small patches, and the CNN model labels each patch. Figure 11 describes the procedure to obtain labeled patches from the C-DVRI. Our CNN model is optimized for 64×64 images. We apply the 64×64 sliding window to the C-DVRI to extract smaller image patches. However, the extracted image patches may have too much blank space. By removing such image patches in advance, the unnecessary computation can be reduced. During the patch extraction, our system performs the information check and eliminates unnecessary patches. We check the amount of information with Oriented FAST and rotated BRIEF (ORB) and Information entropy. ORB is a technique that is frequently employed to detect the feature points of an image in the image processing, and it can be interpreted that an image without a feature

point has no distinct pattern in the image. Also, information entropy is used to examine the consistency and distribution of information. Low information entropy in an image indicates little change in the information. When labeling with CNN, images with little information or few feature points are likely to be assigned wrong labels. To minimize false labeling, we remove patches without feature points and patches with information entropy less than 2 bits from the patch list. The patches after the information check are labeled with the CNN model. The labeled patches in Figure 11 indicate classified patches according to the feature points of the patch images.

B. TF LABELING FROM LABELED PATCHES

In this section, we use the C-DVRIs and G-DVRI described in the previous section for the TF labeling. We label the Grid-TF utilizing the patch positions in the C-DVRI, where labeled patches are extracted. As an example, the patches in Figure 12 are ones of the C-DVRI generated with the Class-TF corresponding to the Class4, which contains 20 grids in the Grid-TF. Note that the grids corresponding to the Class4 are represented with the pink squares, as seen in Figure 10 (a). In this example, the Grid14 is one of the grids for the Class4. Therefore, the volume contained in the Grid14 must be observed in the C-DVRI for Class4, which represents stem, soil, and pot. There are three labeled patches in the purple, green, and yellow boxes in C-DVRI (Class4), as presented in Figure 12. We compare the structural similarity (SSIM) [60] of images in the same image location to examine the structural relationship between C-DVRI (Class4) and G-DVRI (Grid14). The purple, green, and yellow boxes in G-DVRI (Grid14) are located at the same positions in C-DVRI (Class4). We calculate the SSIM between each corresponding patches. In Figure 12, the SSIM

scores are 0.57, 0.49, and 0.54 for the purple, green, and yellow patches, respectively. In the same way, SSIMs are calculated for all labeled patches in C-DVRI (Class4), and the average SSIM is assigned to each label. However, the viewpoint in the 3D rendering is an important factor in featuring the volume objects. Even the DVRI's generated with the same TF are often seen as different representations of the volume objects according to the viewpoints. Figure 13 (a) illustrates the 26 viewpoints. We render the DVRI's by placing the camera at the 26 viewpoints in all directions with 45° intervals about the volume center. (d) and (e) in Figure 13 are the C-DVRI's (Class4) rendered at the two viewpoints in (b) and (c). (d) contains mostly the bonsai stems, while (c) reveals mostly the bonsai pot. Therefore, when labeling the TF with the C-DVRI's, we need to examine all DVRI's from various viewpoints to judge the label. In Figure 12, the SSIM averages for all labels from a viewpoint are presented. As mentioned above, since the data shapes seen in DVRI's vary depending on viewpoints, we create images in multiple directions when producing C-DVRI's and G-DVRI's and calculate SSIM averages for each viewpoint. Therefore, we should integrate all SSIMs of C-DVRI's and G-DVRI's at all viewpoints.

In our study, instead of specifying the G-DVRI label as the largest number of the first ranking labels, we apply the Borda count, which is a preferential, or ranked voting system [61], [62]. Although the higher-ranked candidates receive more points, the Borda count has the disadvantage of being vulnerable to tactical voting. However, in our system, since all processes are executed by computing algorithms, tactical voting cannot occur. Therefore, we estimate the ranks of the SSIM values obtained from the G-DVRI for each viewpoint and perform the Borda counts to designate the Grid-TF label as the label that receives the most points. Also, when labeling with one G-DVRI, a specific label may appear dominant. However, our approach allows us to evaluate all label SSIM values and assists in assigning the label with the highest SSIM regardless of viewpoint changes. In Figure 12, for example, we set Borda counts based on the SSIM average of C-DVRI (Class4) and G-DVRI (Grid14). Our system adds the Borda scores from all viewpoints to determine the grid label of Grid-TF (Grid 14). Since Label 1 has the highest Borda score, we finalize the representative grid label of Grid-TF (Grid14) as Label 1. In this way, we obtain all grid labels of 10×10 Grid-TF.

C. IMAGE-BASED TF GENERATION USING TARGET IMAGE COLORS

We extract colors from the target image and assign them to the image-based TF, which allows us to obtain the DVRI similar to the target image. Figure 14 shows the procedure of creating the image-based TF. (a) is the target image set by the user. (b) shows the process of the patch labeling of the target image. We label the patches in the same way as Figure 11. We divide the target image into patches using a 64×64 sliding window and label them with the SDDVR-

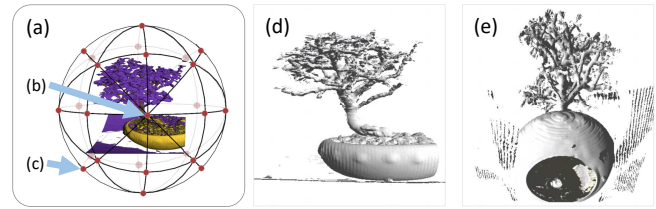


FIGURE 13. The viewpoint settings. (a) illustrates the 26 viewpoints. (d) and (e) are the C-DVRI's rendered at the two viewpoints in (b) and (c). (d) reveals mostly the bonsai stems, while (c) shows mostly the bonsai pot.

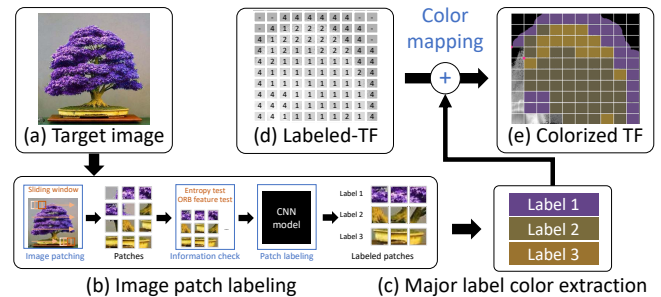


FIGURE 14. The procedure to obtain the image-based TF. We divide the target image into patches, label the patches with CNN model, and extract the label colors from the labeled patches. We then obtain the image-based TF by mapping the obtained major colors to the Labeled-TF.

CNN. The patches are then sorted according to the label. The most dominant color in the patches for a label is assigned to the label color in the image-based TF. However, most labeled patches have a dominant color and background color together. We separate these two colors by performing a *k-mean* algorithm consisting of two clusters in the 3D space with RGB axes. The larger of these two clusters is likely to contain the dominant color of the patches. Therefore, we extract the RGB value corresponding to the centroid of the larger cluster as the dominant color of each label. Figure 14 (c) shows the extracted major colors. We then achieve the image-based TF by mapping the major colors to the Labeled-TF. (d) presents the Labeled-TF and (e) shows the final image-based TF. Note that the Labeled-TF is simply virtual TF stored in an array format, but it is represented as a TF to enhance understanding. The image-based TF allows us to generate DVRI's similar to the target image.

For the TF labeling, we classify the C-DVRI's obtained by the Class-TF applying CNN model and finalize the TF labeling with the voting system. During this process, the C-DVRI's are automatically labeled by the CNN features. Figure 15 presents labeled patches acquired from a C-DVRI. (a) is the C-DVRI and (b)-(d) show the labeled patches according to the patch sizes, 128×128 , 64×64 , and 32×32 , respectively. The purple, green, and orange boxes are patches of leaf, stem, and pot, respectively. Most patches are properly labeled depending on the characteristics of the leaf, stem, and pot with CNN model. However, there are cases where the erroneous labels are assigned, as observed within the red ellipses in Figure 15 (d). When the patch size is small, the patch does not have many feature points, which means that

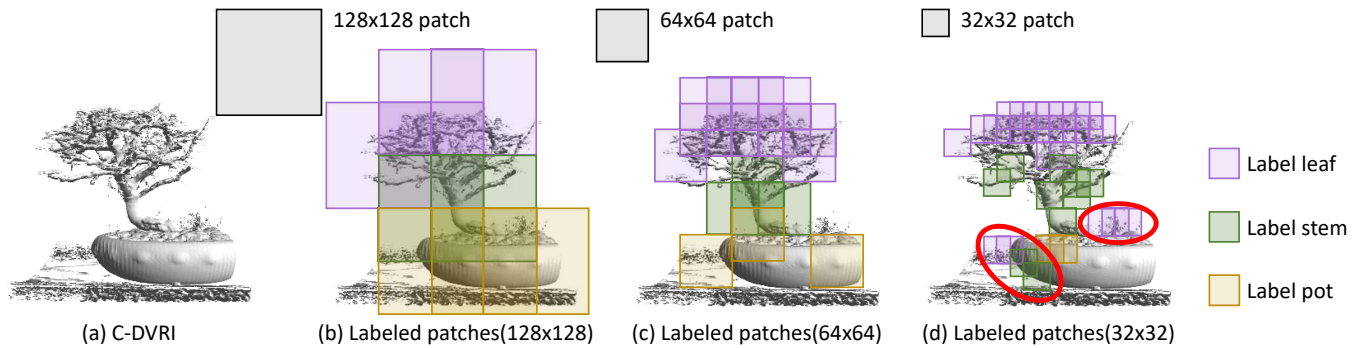


FIGURE 15. Patch size comparison. (a) is the C-DVRI and (b)-(d) show the labeled patches according to the patch sizes, 128×128 , 64×64 , and 32×32 , respectively. The purple, green, and yellow boxes are patches of leaf, stem, and pot, respectively. The red ellipses in (d) are cases where the incorrect labels are assigned.

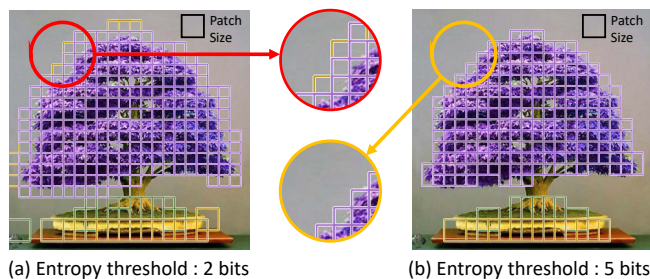


FIGURE 16. The comparison of labeling with different entropy threshold. (a) and (b) show the labeled patches whose entropies are higher than 2 bits and 5 bits, respectively. Mixed colors with the background are removed in (b).

the patch cannot be suitably labeled by the features trained in the CNN. Moreover, since the CNN model is trained with the images from the internet rather than actual DVRI, the features trained in CNN model may be different from ones in DVRI. We introduce the voting system for the TF labeling with Borda count in Section IV-B to prevent the system from partially mislabeling by searching for the best label from labels in 26 different viewpoints. Since our technique filters small errors with the voting system, the wrong labeled patches, seen within the red ellipses in Figure 15 (d), are corrected to the right labels.

We label the target image patches with the CNN model and extract the dominant color for each label as the major color of the label. Figure 16 shows the major colors automatically extracted by auto labeling with the CNN model. During the major color extraction, each image patch is assigned a probability of a label by the CNN model. However, the color of a patch with a low probability is not representative because a patch with a low probability includes several features together. Therefore, we extract colors only from the patches with the label probabilities of 90% or higher. Figure 16 shows the labeling results with 64×64 patches. The purple, green, and yellow boxes indicate the labels for the leaf, stem, and pot, respectively. Although we apply the label probability rule, as seen in the red circle in (a), the leaves overlap with the background within the patches, which leads the system to assign the patch colors as the pot

color. Therefore, we improve the color extraction scheme by adjusting the entropy threshold of the patch. As mentioned in Section IV-A, we utilize the 2-bit entropy threshold to filter out non-informatic patches, which is presented in (a). However, the 2-bit entropy threshold is not sufficient for color extraction. Therefore, we apply the 5-bit entropy threshold to determine the major colors. (b) shows only patches with the entropy threshold of 5 bits or higher. We can see that the patches associated with the background are removed within the orange circle. Although increasing the entropy threshold reduces the number of patches for the color extraction, it enhances to capture more reliable target image colors.

V. RESULT AND DISCUSSION

Our system is built on Intel i7-10700 CPU with 32GB RAM, and Nvidia RTX 2080 Super with 8 GB VRAM. We implement the deep learning module using Python and Keras and integrate it with the volume renderer. The volume renderer is built using Qt and OpenGL with shaders.

In order to evaluate the usability of the proposed image-based TF colorization with CNN, we recruited five novice participants for an experiment, including four students from computer science, and one from the medical field. Since all participants do not know about the volume rendering, we first gave them necessary information about the direct volume rendering and taught them how to use our volume renderer. We provided the participants with the bonsai image as a target image, as seen in Figure 17, and measured the time until they have obtained the volume rendering image similar to the target image. The participants explored the bonsai volume data while looking at the target image. They adjusted the TF and assigned the colors in the rendered image. We requested that the participant informed us when further manipulation was considered pointless as of completion. Figure 17 (a) displays five DVRI manually generated by the participants with bonsai volume dataset. Although all participants selected the purple to render purple leaves, we can see that the colors recognized by all participants are different. User1 rendered the stem, soil, and moss by mixing colors, and User5 gave up the pot color to better represent the moss.







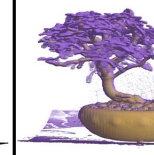

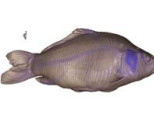







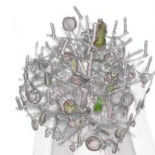
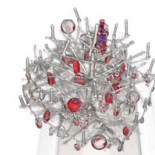

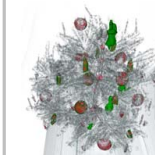
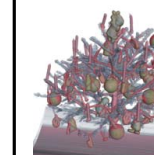
	Target Image	User1	User2	User3	User4	User5	Ours
(a)							
	Rendering time	19min 15sec	24min 23sec	23min 35sec	10min 34sec	47min 46sec	3min 41sec
(b)							
	Rendering time	35min 25sec	12min 14sec	27min 51sec	26min 20sec	17min 1sec	3min 35sec
(c)							
	Rendering time	22min 31sec	15min 21sec	38min 46sec	16min 45sec	23min 6sec	2min 56sec

FIGURE 17. The comparison of volume rendering styles. Five rendering images by users and our rendering image are presented. Five novice users manually produced images with the same target image, whereas our model generated the image automatically resembling the target image.



FIGURE 18. The target image based rendered images for the bonsai data with three different target images.

Although all participants rendered bonsai data using the same target image, all participants assigned different colors to the data. The rendering time varies from 10 minutes 24 seconds to 47 minutes 46 seconds, and the average rendering time is about 25 minutes 7 seconds. We interviewed all the participants about the difficulties in the volume rendering after the experiment. User 1 stated, "After dividing the TF into multiple rendering areas, I tried to assign colors. But I could not remember which TF area indicates which part of the rendered image, hence it took a long time to check." User 2, 3, 5 mentioned, "I wanted to divide the TF into smaller ones as I want, but many volume parts were sharing the same TF area, so I had to compromise." User 2, 4, 5 talked about the difficulty in selecting colors and said, "It is difficult to choose colors with proper saturation and contrast. It seems to require aesthetic sense for the color selection."

Figure 17 (b) shows the DVRIs for the carp dataset.

Different rendering results were produced from the target images because each participant estimated the color values subjectively and had a different region of interest in the volume. After rendering the carp dataset, all participants said, "It was obvious to distinguish carp bones, but it was tough to distinguish between body and bladder." User1 and User4 said, "The body and bladder seem to be located at the same position in the I-GM TF, and I gave up after inspecting the TF by 1 pixel." User2 mentioned that "I checked the shape of each TF area and assigned a color to each area, but when all the selected TF colors were applied, the desired color was not well reproduced because other voxels obstructed the voxels that I want to represent." User3 remarked, "After coloring the bladder, I rotated the volume and found that different part of the body was being rendered with the bladder color at the same time," and stated his experience with perception distortion caused by projecting a 3D space into a 2D image.

Figure 17 (c) shows the DVRIs of the X-mas tree dataset. The participants rendered the X-mas tree dataset with the snow-covered pine tree, red ball, and green bottle as the target images. Since all parts of the X-mas tree are made of similar materials, separating each shape with TF alone is not straightforward. All participants pointed out that the decoration objects are indistinguishable and said, "The tree and decorations are separable, but it is difficult to distinguish different decorations such as bottles and balls." User2 revealed the difficulty of distinguishing object boundaries and three-dimensional recognition, saying, "After applying the color to the red ball, the red voxels were hidden and disappeared when a color was applied to the tree." User4

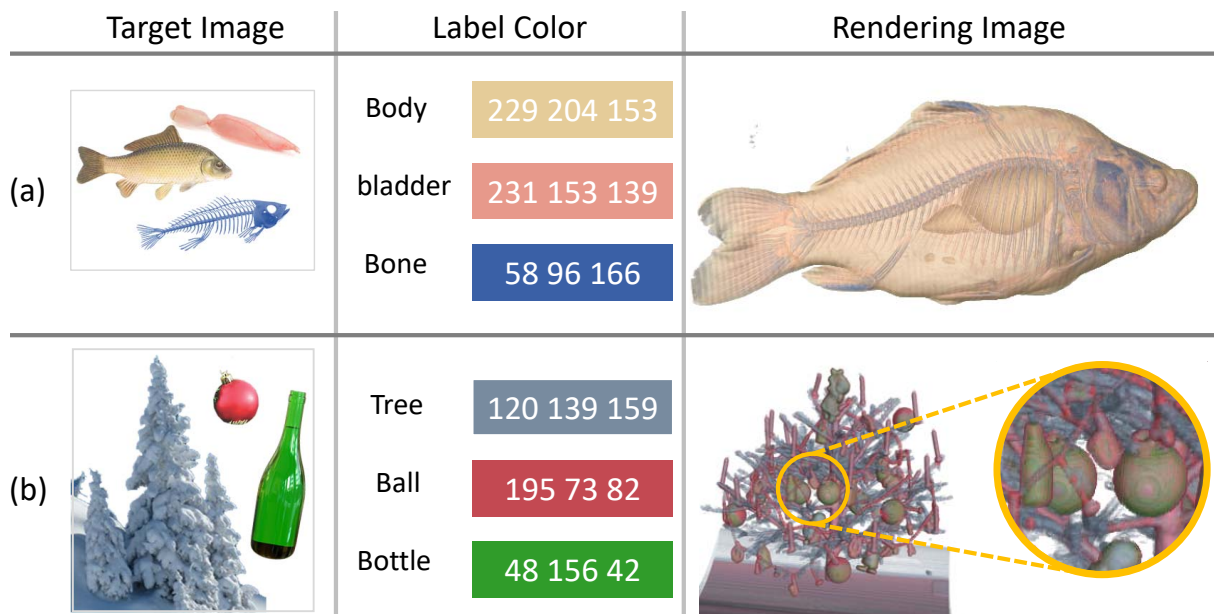


FIGURE 19. The target image-based colorized volume rendering images with the carp and X-mas tree datasets. The carp data is rendered in (a) with three target images, including the bone, air bladder, and carp shape images. The X-mas tree data is rendered with three target images in (b).

talked about the sensitivity of TF manipulation, saying, "I changed the TF only a little bit, but the colors of all voxels changed."

Our approach took less than 4 minutes to render the images. The color scheme of our model is not inferior to the manual rendering images at all, and it was possible to render the volume data faster than the manual rendering by the participants. However, there were some differences in the case of the x-mas tree. Users explored the entire volume despite the difficulty of separating decorations from each other and forcibly divided them into green and red by personal criteria. In contrast, our approach did not distinguish them explicitly and mingled colors in the rendering. In addition, the participants represented the tree in white color, but our approach rendered the tree in a light blue color. It is interpreted that when people perceive colors, the general knowledge they learned beforehand influenced color perception. On the other hand, since there is no information about "snow is white" in our approach, the rendering color is determined by calculating the color values on the target image.

We applied various target images to the bonsai volume data to see how differently rendering images are generated according to different target images. Figure 18 shows the colorized volume rendering images generated with three different target images. (a), (b), and (c) show the rendered images with purple maple, orange maple, and green pine tree as the target images, respectively. We classified the target image features into leaf, stem, and pot to render the bonsai data, and our system determined the feature colors. As seen in the figure, the colors in the rendered images are well-matched with the feature colors of the target images.

We applied the CNN model to the carp data for the

transparent rendering and multiple target images. We first trained CNN by labeling the carp with the body, bone, and air bladder features. During the bonsai data rendering, since different features were found within one target image, it was possible to render the data with only one target image. However, it was difficult to obtain the target images that contain both internal structures and an external shape at the same time. Therefore, we selected different feature images as the target images. Figure 19 (a) presents three target images to render the carp data, including the bone, air bladder, and carp shape images. The CNN model determined each feature color, which is called the label colors in the figure. Once our system automatically generated the rendered image with the colorized TF, we adjusted the global opacity separately to show the inside and outside of the volume simultaneously. As seen in the image, we can observe that the hard object, which is the bone, was explicitly rendered. However, the soft objects were not well separated, which indicates that the bladder and body share similar data values. We will study how to separate these features in the future. Moreover, since the global opacity setting is operated manually, we will investigate optimal opacity values for features by synthesizing the information of the target images as future work.

Figure 19 (b) shows the target images, label colors, and rendering images for the automatic X-mas tree rendering. We trained CNN by labeling the X-mas tree with the tree, ball ornament, and bottle ornament features. We applied three target images. In the our approach, we can see that the tree is rendered with the snow color automatically assigned by the CNN model, but the ornaments have mixed colors of all three label colors. Our deep learning model determines the colors by matching the object shapes of the data with the object shapes in the target image. However, the rendering

TABLE 1. Computing time of image-based TF colorization

Dataset name	Volume data Resolution	Number of Images for label training	Model training(M)		TF generation(TF)		Rendering time
			$M_{labeling}$	$M_{training}$	$TF_{extract}$	$TF_{coloring}$	
Bonsai	$256 \times 256 \times 256$	2,764	1h 46m	2h 19m 8s	2m 5s	1m 36s	<1ms
Carp	$256 \times 256 \times 512$	1,360	20m	15m 44s	2m 11s	1m 24s	
X-mas tree	$256 \times 249 \times 256$	1,663	32m	23m 3s	2m 7s	49s	

TABLE 2. Performance comparison of VGGNet, ResNet, LeNet, and AlexNet architectures.

Dataset Name	Architecture	Accuracy	Validation Accuracy	Loss	Validation Loss	Training Time
Bonsai	VGGNet	0.9982	0.9903	0.0053	0.0049	2h 19m 8s
	ResNet	0.9986	0.9834	0.0040	0.1159	2h 30m 36s
	AlexNet	0.9992	0.9841	0.0023	0.0971	2h 22m 1s
	LeNet	0.9278	0.8983	0.1761	0.0242	42m 54s
Carp	VGGNet	0.9979	0.9843	0.0061	0.0007	15m 44s
	ResNet	0.9977	0.9691	0.0082	0.0973	15m 56s
	AlexNet	0.9986	0.9631	0.0052	0.2757	16m 23s
	LeNet	0.9654	0.9004	0.0902	0.2814	5m 23s
X-mas tree	VGGNet	0.9982	0.9962	0.0058	0.0666	23m 3s
	ResNet	0.9979	0.9695	0.0071	0.1928	24m 20s
	AlexNet	0.9973	0.9884	0.0112	0.0607	23m 20s
	LeNet	0.9845	0.9368	0.0427	0.2718	8m 52s

system separates the material features and renders the data accordingly. The ball and bottle ornaments of the X-mas tree data cannot be separated by our TF alone. Therefore, our approach has a limitation in that the colors may not be distinguished if the target image is selected based on the shapes without considering the material features. This problem is also the limitation of the I-GM TF. Datasets that are difficult to classify by the I-GM TFs can be classified with multidimensional TFs or classifying voxels in volume space rather than projected space. However, this paper aims to lower the hurdles for rendering for beginners, so we avoided either applying multidimensional TFs that required complex interactions or approaches that compromised the underlying volume pipeline. We are planning a study using multidimensional TFs as a future work.

Table 1 presents the computing time of the CNN model and rendering. We performed the target image-based colorized volume renderings for the bonsai, carp, and X-mas tree datasets. The computing times were measured separately for the CNN model training time, M , and the TF labeling and TF generation time, TF . The CNN model training time includes labeling time, $M_{labeling}$, and training time, $M_{training}$. The $M_{labeling}$ is the time it takes to create labeled patch images from the crawled images introduced in Section III-B. The $M_{training}$ is the time for training the CNN model with the labeled patch images. Note that M is not necessary if CNN model is already trained within the system. TF is composed of $TF_{extract}$ and $TF_{coloring}$. $TF_{extract}$ is the time to extract major colors from target images. $TF_{coloring}$ is the time to label TF and map major colors. As seen from TF in Table 1, our system performs the image-based TF colorization within less than 4 minutes. Since the TF is pre-processing operations, it does not affect the performance of direct volume rendering.

Table 2 shows the performance of deep models, including

VGGNet, ResNet, AlexNet, and LeNet, on the bonsai, carp, and x-mas tree datasets, respectively. VGGNet, ResNet, and AlexNet have accuracy above 0.99 and loss below 0.015. However, the validation loss varies for deep learning models and datasets. VGGNet produces consistently good performance for all datasets. Also, VGGNet has short computation times compared to ResNet and AlexNet. However, since the experiment was evaluated with specific datasets, an experiment with more data is necessary. Therefore, we plan to compare deep learning architectures with more datasets in the future. Also, to improve the model performance, we will investigate how the depth of deep learning architecture affects the performance.

In our work, it is possible to reuse the pre-trained CNN model for domains similar to the previously trained domains. However, the CNN approach for different domains demands to retrain the CNN model with new labeled images according to new data domains. To overcome this limitation in labeled data acquisition, we developed a collaborative data labeling system and a semi-automatic labeling procedure as a learning process for a new domain. For the training data labeling, as mentioned in Section III-B, it took about 15 seconds per single image with our in-house software. We found 2,764 bonsai images obtained on the Internet, and the time it took for one person to label all images manually was about 12 hours. The collaborative labeling system took an hour and 46 minutes when ten people labeled the bonsai images together.

Although the collaborative system was efficient in reducing the time for the data labeling, the manual data labeling is still cumbersome. Different approaches to avoid manual labeling could be semi-automatic learning and few-shot learning. For the semi-automatic labeling approach, 100 images are labeled manually, and then the labeled images are used to train any CNN model. Afterward, new images are labeled with the CNN model, and a human just reviews

the labels and corrects only the wrong labels. Although the initial performance of semi-automatic learning is somewhat poor, if it is combined with the collaborative system, it is possible to obtain a stable model that gradually improves performance. As another approach, we plan future work for a meta-learning-based method that learns metadata constructed from only one image and a method that expands the learning data using a GAN that has learned a target image.

VI. CONCLUSION AND FUTURE WORKS

In this paper, we proposed an image-based TF colorization with CNN to automatically generate a direct volume rendering image (DVRT) similar to a target image color. We created labeled datasets from internet images and trained the CNN model. We defined various TFs, including Grid-TF, AS-TF, and Class-TF, to label the TF using information entropy and a voting technique with Borda count scores. We also extracted label colors from the target image with the CNN model to create a colorized TF for the final volume rendering. For the usability evaluation, we performed an experiment with novice participants and compared manual volume rendering by the participants with our automatic rendering image. We used the bonsai, carp, and X-mas tree volume datasets to evaluate our rendering pipeline and system. However, our approach employs supervised learning, which has the disadvantage that the labeling time might be significantly longer. Besides, there is a limitation in that it is difficult to apply different feature to objects that are not distinguishable in the I-GM TF space. Therefore, we plan to multidimensional TF approach and apply the automatic labeling technique or few-shot learning techniques as future work. Also, we are planning semi-supervised learning-based volume rendering in which GAN and CNN are associated as a technique of training a single target image without using multiple labeled datasets in the future.

REFERENCES

- [1] S. Arens and G. Domik, "A survey of transfer functions suitable for volume rendering," in Proceedings of the 8th IEEE/EG international conference on Volume Graphics, pp. 77–83, Eurographics Association, 2010.
- [2] 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," Computer Graphics Forum, vol. 35, no. 3, pp. 669–691, 2016.
- [3] R. Maciejewski, Y. Jang, I. Woo, H. Jänicke, K. P. Gaither, and D. S. Ebert, "Abstracting attribute space for transfer function exploration and design," IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 1, pp. 94–107, 2013.
- [4] H.-C. Cheng, A. Cardone, S. Jain, E. Krokos, K. Narayan, S. Subramaniam, and A. Varshney, "Deep-learning-assisted volume visualization," IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 2, pp. 1378–1391, 2018.
- [5] L. Fang, J. Liu, J. Liu, and R. Mao, "Automatic segmentation and 3d reconstruction of spine based on fcn and marching cubes in ct volumes," in 2018 10th International Conference on Modelling, Identification and Control (ICMIC), pp. 1–5, IEEE, 2018.
- [6] N. Shi and Y. Tao, "Cnns based viewpoint estimation for volume visualization," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 10, no. 3, pp. 1–22, 2019.
- [7] M. Berger, J. Li, and J. A. Levine, "A generative model for volume rendering," IEEE Transactions on Visualization and Computer Graphics, vol. 25, no. 4, pp. 1636–1650, 2018.
- [8] S. Martin, S. Bruton, D. Ganter, and M. Mancke, "Using a depth heuristic for light field volume rendering," in Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP, pp. 134–144, 2019.
- [9] S. Weiss, M. Chu, N. Thuerey, and R. Westermann, "Volumetric isosurface rendering with deep learning-based super-resolution," Preprint on IEEE Transactions on Visualization and Computer Graphics, 2019.
- [10] R. A. Drebin, L. Carpenter, and P. Hanrahan, "Volume rendering," in Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '88, (New York, NY, USA), pp. 65–74, ACM, 1988.
- [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, IEEE, 1996.
- [12] J. Li, L. Zhou, H. Yu, H. Liang, and L. Wang, "Classification for volume rendering of industrial ct based on moment of histogram," in 2007 2nd IEEE Conference on Industrial Electronics and Applications, pp. 913–918, 2007.
- [13] S. Arens and G. Domik, "A Survey of Transfer Functions Suitable for Volume Rendering," in IEEE/EG Symposium on Volume Graphics (R. Westermann and G. Kindlmann, eds.), The Eurographics Association, 2010.
- [14] T. Fogal and J. Krüger, "Tuvok, an Architecture for Large Scale Volume Rendering," in Proceedings of the 15th International Workshop on Vision, Modeling, and Visualization, November 2010.
- [15] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, G. H. Weber, H. Krishnan, et al., "Visit: An end-user tool for visualizing and analyzing very large data," High performance visualization—enabling extreme-scale scientific insight, pp. 357–372, 2012.
- [16] 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, vol. 12, no. 6, pp. 1570–1579, 2006.
- [17] J. Kniss, G. Kindlmann, and C. Hansen, "Multidimensional transfer functions for interactive volume rendering," IEEE Transactions on Visualization and Computer Graphics, vol. 8, no. 3, pp. 270–285, 2002.
- [18] C. J. Morris and D. Ebert, "Direct Volume Rendering of Photographic Volumes Using Multi-Dimensional Color-Based Transfer Functions," in Eurographics / IEEE VGTC Symposium on Visualization (D. Ebert, P. Brunet, and I. Navazo, eds.), The Eurographics Association, 2002.
- [19] G. Kindlmann and J. W. Durkin, "Semi-automatic generation of transfer functions for direct volume rendering," in Volume Visualization, 1998. IEEE Symposium on, pp. 79–86, IEEE, 1998.
- [20] J. Hladuvka, A. König, and E. Gröller, "Curvature-based transfer functions for direct volume rendering," in Proceedings of Spring Conference on Computer Graphics and its Applications, pp. 58–65, 2000.
- [21] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller, "Curvature-based transfer functions for direct volume rendering: Methods and applications," in Visualization, 2003. VIS 2003. IEEE, pp. 513–520, IEEE, 2003.
- [22] S. Roettger, M. Bauer, and M. Stamminger, "Spatialized transfer functions," in EuroVis, pp. 271–278, 2005.
- [23] C. Correa and K.-L. Ma, "The occlusion spectrum for volume classification and visualization," IEEE Transactions on Visualization and Computer Graphics, vol. 15, no. 6, pp. 1465–1472, 2009.
- [24] C. D. Correa and K.-L. Ma, "Visibility histograms and visibility-driven transfer functions," IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 2, pp. 192–204, 2011.
- [25] J. M. Kniss, R. V. Uiter, A. Stephens, G. Li, T. Tasdizen, and C. Hansen, "Statistically quantitative volume visualization," in VIS 05. IEEE Visualization, 2005., pp. 287–294, 2005.
- [26] P. Sereida, A. V. Bartoli, I. W. Serlie, and F. A. Gerritsen, "Visualization of boundaries in volumetric data sets using 1h histograms," IEEE Transactions on Visualization and Computer Graphics, vol. 12, no. 2, pp. 208–218, 2006.
- [27] M. A. Selver and C. Guzelis, "Semiautomatic transfer function initialization for abdominal visualization using self-generating hierarchical radial basis function networks," IEEE Transactions on Visualization and Computer Graphics, vol. 15, no. 3, pp. 395–409, 2009.
- [28] 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, vol. 15, no. 6, pp. 1473–1480, 2009.
- [29] M. Haidacher, D. Patel, S. Bruckner, A. Kanitsar, and M. E. Gröller, “Volume visualization based on statistical transfer-function spaces,” in IEEE Pacific Visualization Symposium (PacificVis), pp. 17–24, 2010.
- [30] W. Serna-Serna, A. M. Álvarez-Meza, and Á.-Á. Orozco-Gutierrez, “Volume rendering by stochastic neighbor embedding-based 2d transfer function building,” in Iberoamerican Congress on Pattern Recognition, pp. 618–626, Springer, 2017.
- [31] S. Castro, A. König, H. Löffelmann, and E. Gröller, “Transfer function specification for the visualization of medical data,” Vienne University of Technology, 1998.
- [32] 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, vol. 12, no. 5, 2006.
- [33] C. Lundström, P. Ljung, and A. Ynnerman, “Multi-Dimensional Transfer Function Design Using Sorted Histograms,” in Volume Graphics (R. Machiraju and T. Moeller, eds.), The Eurographics Association, 2006.
- [34] Y. Wu and H. Qu, “Interactive transfer function design based on editing direct volume rendered images,” IEEE Transactions on Visualization and Computer Graphics, vol. 13, no. 5, pp. 1027–1040, 2007.
- [35] J. J. Caban and P. Rheingans, “Texture-based transfer functions for direct volume rendering,” IEEE Transactions on Visualization and Computer Graphics, vol. 14, no. 6, pp. 1364–1371, 2008.
- [36] S. Wesarg, M. Kirschner, and M. F. Khan, “2d histogram based volume visualization: combining intensity and size of anatomical structures,” International journal of computer assisted radiology and surgery, vol. 5, no. 6, pp. 655–666, 2010.
- [37] C. Correa and K.-L. Ma, “Size-based transfer functions: A new volume exploration technique,” IEEE Transactions on Visualization and Computer Graphics, vol. 14, no. 6, pp. 1380–1387, 2008.
- [38] H. Akibay and K.-L. May, “A tri-space visualization interface for analyzing time-varying multivariate volume data,” in Proceedings of the 9th Joint Eurographics/IEEE VGTC conference on Visualization, pp. 115–122, Eurographics Association, 2007.
- [39] H. Guo, H. Xiao, and X. Yuan, “Multi-dimensional transfer function design based on flexible dimension projection embedded in parallel coordinates,” in IEEE Pacific Visualization Symposium (PacificVis), pp. 19–26, 2011.
- [40] H. Guo, H. Xiao, and X. Yuan, “Scalable multivariate volume visualization and analysis based on dimension projection and parallel coordinates,” IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 9, pp. 1397–1410, 2012.
- [41] H. Guo, N. Mao, and X. Yuan, “Wysiwyg (what you see is what you get) volume visualization,” IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 12, pp. 2106–2114, 2011.
- [42] L. Wang, X. Zhao, and A. E. Kaufman, “Modified dendrogram of attribute space for multidimensional transfer function design,” IEEE Transactions on Visualization and Computer Graphics, vol. 18, no. 1, pp. 121–131, 2012.
- [43] S. Lan, L. Wang, Y. Song, Y.-p. Wang, L. Yao, K. Sun, B. Xia, and Z. Xu, “Improving separability of structures with similar attributes in 2d transfer function design,” IEEE Transactions on Visualization and Computer Graphics, vol. 23, no. 5, pp. 1546–1560, 2017.
- [44] I. Fujishiro, T. Azuma, and Y. Takeshima, “Automating transfer function design for comprehensible volume rendering based on 3d field topology analysis,” in Visualization’99. Proceedings, pp. 467–563, IEEE, 1999.
- [45] P. Sereda, A. Vilanova, and F. A. Gerritsen, “Automating transfer function design for volume rendering using hierarchical clustering of material boundaries,” in EuroVis, pp. 243–250, 2006.
- [46] T. Pfaffelmoser, M. Reitingner, and R. Westermann, “Visualizing the positional and geometrical variability of isosurfaces in uncertain scalar fields,” Computer Graphics Forum, vol. 30, no. 3, pp. 951–960, 2011.
- [47] R. Bramon, M. Ruiz, A. Bardera, I. Boada, M. Feixas, and M. Sbert, “Information theory-based automatic multimodal transfer function design,” IEEE journal of biomedical and health informatics, vol. 17, no. 4, pp. 870–880, 2013.
- [48] B. Ma and A. Entezari, “Volumetric feature-based classification and visibility analysis for transfer function design,” IEEE Transactions on Visualization and Computer Graphics, vol. 24, no. 12, pp. 3253–3267, 2018.
- [49] 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, Eurographics Association, 2007.
- [50] K. P. Soundararajan and T. Schultz, “Learning probabilistic transfer functions: A comparative study of classifiers,” Computer Graphics Forum, vol. 34, no. 3, pp. 111–120, 2015.
- [51] T. M. Quan, J. Choi, H. Jeong, and W.-K. Jeong, “An intelligent system approach for probabilistic volume rendering using hierarchical 3d convolutional sparse coding,” IEEE Transactions on Visualization and Computer Graphics, vol. 24, no. 1, pp. 964–973, 2018.
- [52] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, “Face recognition: A convolutional neural-network approach,” IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 98–113, 1997.
- [53] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial nets,” in Advances in neural information processing systems, pp. 2672–2680, 2014.
- [54] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” CoRR, vol. abs/1212.5701, 2012.
- [55] 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, IEEE, 2019.
- [56] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems, pp. 1097–1105, 2012.
- [57] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in 3rd International Conference on Learning Representations, ICLR 2015, Conference Track Proceedings (Y. Bengio and Y. LeCun, eds.), 2015.
- [58] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9, 2015.
- [59] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.
- [60] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, 2004.
- [61] J. d. Borda, “Mémoire sur les élections au scrutin,” Histoire de l’Académie Royale des Sciences pour 1781 (Paris, 1784), 1784.
- [62] P. Emerson, “The original borda count and partial voting,” Social Choice and Welfare, vol. 40, no. 2, pp. 353–358, 2013.