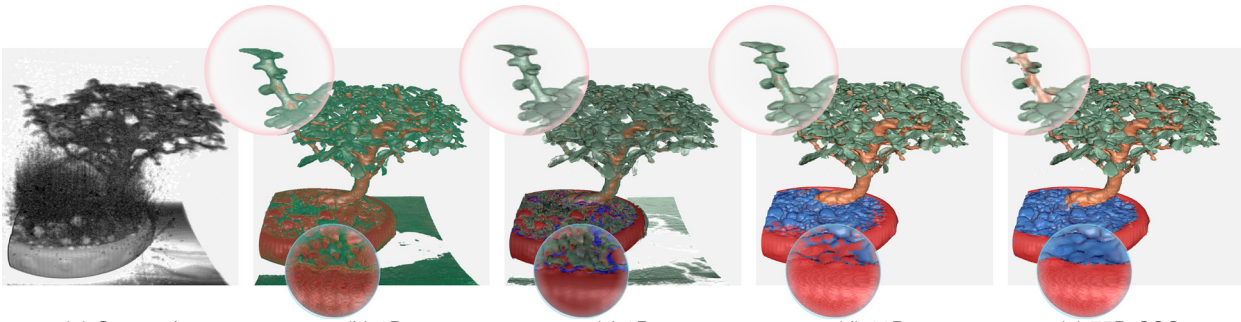
An Intelligent System Approach for Probabilistic Volume Rendering using Hierarchical 3D Convolutional Sparse Coding

Tran Minh Quan†, *Student Member, IEEE*, Junyoung Choi†, *Student Member, IEEE*, Haejin Jeong, *Student Member, IEEE*, and Won-Ki Jeong*, *Member, IEEE*



(a) Grayscale (b) 1D (c) 3D (d) 11D (e) 75D-CSC

Fig. 1: Comparison of different volume visualization results. (a) Grayscale volume rendering, (b) 1D transfer function, (c) 3D transfer function Kniss et al. [17], (d) 11D intensity-driven feature with the random forest classifier Soundararajan and Schultz [29], and (e) ours. Among all, our method results in the most accurate classification (vase, leaves, trunks, and soils are clearly separated), showing the discriminative power of high-dimensional feature vectors generated by using hierarchical 3D convolutional sparse coding. The same user scribble input is used for (d) and (e).

Abstract— In this paper, we propose a novel machine learning-based voxel classification method for highly-accurate volume rendering. Unlike conventional voxel classification methods that incorporate intensity-based features, the proposed method employs dictionary based features learned directly from the input data using hierarchical multi-scale 3D convolutional sparse coding, a novel extension of the state-of-the-art learning-based sparse feature representation method. The proposed approach automatically generates high-dimensional feature vectors in up to 75 dimensions, which are then fed into an intelligent system built on a random forest classifier for accurately classifying voxels from only a handful of selection scribbles made directly on the input data by the user. We apply the probabilistic transfer function to further customize and refine the rendered result. The proposed method is more intuitive to use and more robust to noise in comparison with conventional intensity-based classification methods. We evaluate the proposed method using several synthetic and real-world volume datasets, and demonstrate the methods usability through a user study.

Index Terms—Volume Rendering, Machine Learning, Hierarchically Convolutional Sparse Coding

1 Introduction

Direct volume rendering (DVR) [7, 11, 19] is a powerful visual tool for interpreting three-dimensional (3D) volumetric data used in numerous fields, such as medicine, natural and applied sciences, and engineering. Significant research efforts have been made to improve the speed and quality of DVR over the past few decades, including designing transfer functions [26]. The transfer function involves mapping data values to optical properties, i.e., color and opacity, and classifies which region in the data is visible. However, designing good transfer functions poses many challenges, especially to non-expert users, for a number of reasons. Most transfer function design approaches require significant manual user interactions, as the process inherently involves trial-and-error and is based on human decisions using data statistics, such as a histogram of the feature values. In addition, although multi-dimensional transfer functions have become increasingly popular due to their superior capacity for the classification of complex data, the feature spaces that can be explored in multi-dimensional transfer functions are hardly higher than three dimensions because of difficulties arising from the large perceptual gap between the transfer function domain and the spatial domain.

The other alternative to the conventional transfer function is explicit selection in the spatial domain to classify visible regions. The seminal work by Tzeng et al. [33] introduced high-dimensional probabilistic classification based on direct user selection of the input volume. The proposed method is an *intelligent system* approach, meaning that users specify the regions of interest using a paintbrush tool, whereupon the selected regions (i.e., voxels) are used to build high-dimensional features to train a machine learning classifier. In this work, intensity-based features, such as voxel intensity values, the magnitude of the gradient, and the current voxel location, are used to define 11 dimensional feature vectors. A recent work by Soundararajan and Schultz [29] compared various supervised learning algorithms for classification within the same intelligent system approach to suggest the best classifier. This alternative approach provides easy-to-use, high-level user interaction tools for leveraging high-dimensional features for interactive classification in DVR.

The main motivation for the proposed work stems from the following observations: The intelligent system based on high-dimensional features has opened the door to a new research direction for leveraging machine learning in DVR. However, the intensity-based feature model adopted in the conventional intelligent system may not work well under certain harsh conditions (e.g., data that contain noise and anisotropic shapes). For example, as shown in Figure 1, conventional methods fail

E-mails: {quantm, juny0603, goodhen2, wkjeong}@unist.ac.kr.
The authors are with Ulsan Nat'l Inst. of Science and Technology (UNIST).
†Both authors contributed equally to this work, ∗Corresponding author

Manuscript received 31 Mar. 2017; accepted 1 Aug. 2017.
Date of publication 28 Aug. 2017; date of current version 1 Oct. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2017.2744078

to clearly separate different structures or noise from the objects in the volume. This failure occurs because the intensity-based features cannot distinguish noise and voxels from the object of interest if they fall into a similar intensity and gradient range. In addition, there exist various image modalities, e.g., low-dose computer tomography, ultrasound, and optical coherence tomography, etc, that are either noisy or have image structures that cannot be easily interpreted using conventional intensity-based algorithms. Another observation is that the recent advances in machine learning and deep neural networks have significantly improved the image classification accuracy [18], but their direct adaptation in volume rendering has not been actively explored yet.

To address these problems, we introduce a novel high-dimensional feature model based on the 3D convolutional sparse coding (CSC) [15], a state-of-the-art unsupervised learning-based sparse representation algorithm, for accurate voxel classification in DVR. Inspired by deep convolutional neural networks, we propose a novel *hierarchical* CSC that successively builds up dictionaries in different scales. The proposed hierarchical CSC automatically extracts multi-scale features from the input volume, which are then used to build up high-dimensional feature vectors that encode local shapes more robustly and discriminatively compared with the conventional intensity-based features or a single-scale dictionary. This method also benefits from the characteristic of sparse representation that makes the final rendering less sensitive to noise. We also propose a robust 3D CSC method that uses a blackout approach, which is inspired by the *dropout* technique [30] in deep neural networks and autoencoders to establish a more general and compressive collection of 3D filter banks. We then employ an intelligent system built on a random forest classifier for accurate and interactive voxel classification using high-dimensional feature vectors, up to 75 dimensions, from only a handful of selection scribbles made directly on the input volume by the user. Last, we apply the probabilistic transfer function to further customize and refine the rendered result. [1]

2 Related Work

Transfer Function Design Conventional DVR [7, 19] focuses mainly on designing transfer functions that map volume data to color and opacity values. Early transfer functions are simply one-dimensional because the mapping depends only on input intensity, but more complicated classification methods are necessary for noisy and complex 3D volume data. An immediate extension of the one-dimensional transfer function is employing more dimensions for the feature space. Kindlmann and Durkin [16] suggested using first- and second-order derivatives with histogram visualization for multidimensional transfer functions. Kniss et al. [17] introduced a novel user interface (widget) for easy handling of multidimensional transfer functions. Park and Bajaj [24] proposed various data features, such as the gradient magnitude and second derivatives, with the voxels' spatial distances to the boundaries to determine the opacity characteristics. Correa and Ma [5] advocated using the scale feature to make classifications, which was constructed based on the relative size of each voxel at different volume resolutions. Although the multi-dimensional transfer function is useful, user interactions become more challenging as the number of dimensions increases. Other methods provided either a segmentation-driven analysis of the 2D histogram [13] or an abstracted attribute space between the density value versus the gradient magnitude [21]. Nevertheless, manual interaction with a multi-dimensional transfer function is generally time-consuming and difficult for non-expert users. The recent work by Guo et al. [9, 10] introduced interesting sketch-based volume rendering systems that nicely address the challenges of conventional transfer functions shown above. However, advanced voxel classification based on machine learning and data-driven techniques has not been actively adopted to advanced sketch-based approaches yet, which leaves room for improvement in the transfer function design.

Learning-based Volume Rendering The transfer function in volume rendering is, in fact, a voxel classification method. Therefore, machine learning can serve as an accurate voxel classifier. Volume rendering that employed machine-learning techniques was first introduced in the seminal work of Tzeng et al. [33]. Their approach involves directly selecting visible voxels from the data space (i.e., volume) using a paintbrush tool, rather than manipulating the transfer function to indirectly classify the visible voxels. Feature vectors, input to a learning-based classifier, are then built from the selected voxels using the current intensities and the gradient magnitude as well as those of its six-neighbors, and location information x, y, and z, which builds up to an 11-dimensional representation for each selected voxel. A neural network with a single hidden layer is leveraged to train the feature vectors, and is eventually used to classify each voxel in the volume as foreground or background material (i.e., binary classification). Most recently, Soundararajan and Schultz [29] systematically compared the effect of different classification approaches on multinomial classification tasks and concluded that no single technique is significantly better than the others but, random forests are generally useful in most scenarios. On the other hand, Johnson and Huang [14] explicitly considered the local frequency distribution of the 3D stencil in which a particular voxel is located at the center. This training query helps estimate the relational pattern between 3D stencils and is then deployed to each voxel for final classification. Cai et al. [4] recently proposed abstract volume attributes, i.e., rules of frequency distribution, that were trained using a selection-based genetic algorithm. As a result, they were able to effectively visualize the overlapping data values in 3D volumes. A more comprehensive survey of the state-of-the-art in transfer functions for DVR can be found in [20]. Although previous research efforts attempted to exploit machine learning for classification in DVR, unsupervised learning for data-driven feature vector generation, as in our proposed method, has never been examined.

Dictionary Learning and Sparse Coding Dictionary learning is a technique that helps decompose many patches of signal into a linear combination of overcomplete basis sets and that forces the associated codes to be sparse. This technique has been shown to be useful in many image-processing applications, such as image denoising [8]. A similar approach is convolutional sparse coding, in which patch extraction is replaced by a convolution operator. A solution of the energy function can be found efficiently in the frequency domain, where the convolution operator can be transformed into an element-wise multiplication. The entire solver was derived within the Alternating Direction Method of Multiplier (ADMM) framework proposed by Bristow et al. [3]; Wohlberg [35, 36] later introduced the direct inverse solution for more efficient convergence. Convolutional sparse coding can generate many compact dictionaries because of the shift-invariant nature of the filters, and the pixel-wise computation in the Fourier domain maps well to parallel architecture, such as GPUs. Hence, it has been applied to various machine vision problems, such as image inpainting [12], classification [39], segmentation [38], super-resolution [23] and reconstruction [28]. However, these advanced machine learning approaches have not yet been fully exploited in the volume rendering literature. In this paper, we propose a novel feature-based volume rendering method that leverages the compactness and efficiency of hierarchical 3D convolutional sparse coding to enrich the feature vectors.

3 Overview of Convolutional Sparse Coding

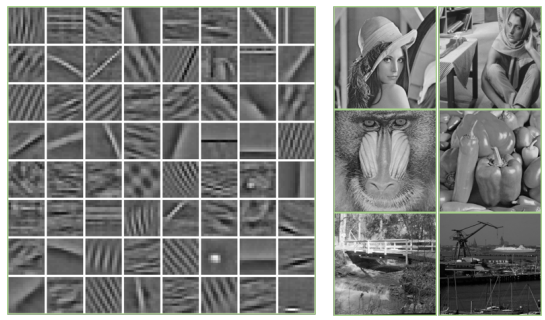


Fig. 2: A 64-atom dictionary generated from natural images.

CSC can be regarded as an alternative to dictionary learning [1], which builds the dictionary with convolution filters instead of local

[1] Source code is available at http://hvcl.unist.ac.kr/pvr/

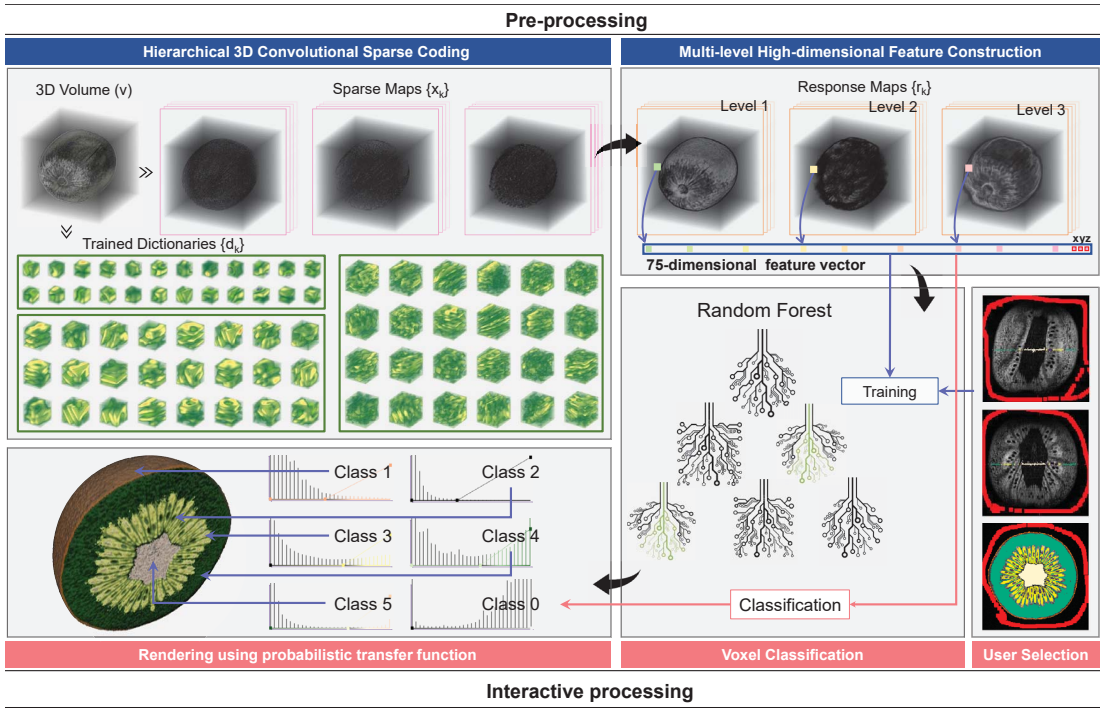


Fig. 3: Overview of the proposed intelligent volume rendering system.

patches. In this section, we present a brief background review of CSC for 2D images. For a given image s, we would like to find its best approximation from the summation of *response maps* $\sum r_k$. This reverse problem can be solved if we impose a constraint that each *response map* r_k is the result of convolution between a *filter* (or *atom*) d_k and its associated *sparse map* x_k. The non-linear ℓ_p norm (where $0 \leqslant p \leqslant 1$), which is applied to x_k (i.e., x_k is sparse), affords a feasible solution of finding such a collection of d_k and x_k. Once d_k and x_k are obtained properly, we can compute back the *response map* r_k by convolving the corresponding *filter* and *sparse map*. Mathematically, the CSC problem is equivalent to minimizing this energy function:

$$\min_{d,x} \frac{\alpha}{2} \left\| s - \sum_k d_k * x_k \right\|_2^2 + \lambda \sum_k \|x_k\|_1 \quad s.t.: \quad \|d_k\|_2^2 \leqslant 1 \qquad (1)$$

In Equation 1, the first term measures the difference between s and its sparse approximation $\sum_k d_k * x_k$, weighted by α. The second term is the sparsity regularization of x_k using an ℓ_1 norm with a weight λ instead of an ℓ_0 norm as used in [1]. The remaining constraint restricts the Frobenius norm of each atom d_k within a unit length.

If we wish to train d_k to generalize for the dictionary and to represent the features of many images obtained from a database, all the training instances can be fetched and participate in contributing to the feature extraction. For example, Figure 2 shows a 64-atom dictionary that has been trained using a collection of natural and standard images (lena, barbara, etc.). Their components in Gabor-like shapes capture directional edges that match the fundamental features in our human visual perception.

Solving Equation 1 is introduced in the seminal work of Zeiler et al. [37]. They proposed an alternating strategy in which a series of convex subproblems between d_k and x_k is solved until convergence. Because their solver is completely in the image domain, the linear complexity of convolution affects the performance of their algorithm. More efficient approaches based on the Fourier Convolution theorem are also proposed [3, 35, 36]. The filter is padded with zeros to make it the same size as the image, and the Fourier transform is applied to both the padded filter and the image so that the convolution can be computed as a pixel-wise multiplication in the Fourier domain.

4 Proposed Method

In this section, we introduce our high-dimensional feature-based intelligent volume rendering system in detail.

4.1 Overview of the Proposed System

Figure 3 depicts the overview of the proposed intelligent volume rendering system. The proposed system consists of two components – one component is the pre-processing stage to construct the hierarchical 3D dictionary from the input volume and to create a per-voxel multi-level high-dimensional feature vector (see the upper row in Figure 3). This process is required only once for each input volume. The other component is the interactive stage to accept user selections to train the random forest classifier using high-dimensional feature vectors to classify voxels and to generate a rendered image based on the user-given probabilistic transfer function for each class (see the bottom row in Figure 3).

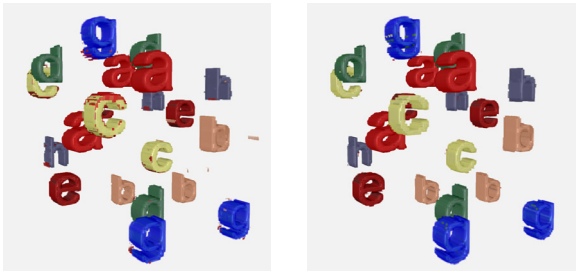
4.2 Generating 3D Dictionary using 3D CSC

4.2.1 Training 3D Atoms using 3D CSC

In this section, we extend 2D CSC formulation to 3D to apply for volume rendering. The main formulation is identical to Equation (1) except now the 2D image s is replaced with the 3D volume v. If we use a simplified notation without the index k and replace the result for the Fourier transform of a given variable by using the subscript f (for example, d_f is the simplified notation for $\mathscr{F}d$ in the 3D domain to derive the solution to Equation (1)), then the 3D CSC problem can be expressed using the auxiliary variables y and g for x and d as follows:

$$\min_{d,x,g,y} \frac{\alpha}{2} \left\| v - \sum d * x \right\|_2^2 + \lambda \|y\|_1$$
$$s.t.: \|g\|_2^2 \leqslant 1, \quad x - y = 0, \quad g = Proj(d) \qquad (2)$$

where g and d are related by a projection operator as a combination of a truncated matrix followed by a zero-padding matrix in order to make the dimension of g same as that of x. The above constraint problem can be solved using an augmented Lagrangian method [28] (more details can be found in the supplementary text).



(a) No regularizer (b) With blackout

Fig. 4: Comparison of voxel classification accuracy without (a) and with (b) blackout regularizer.

4.2.2 Improving the Dictionary using Blackout Regularizer

Table 1: Effect of using blackout regularizer.

Metrics	No regularizer	With blackouts
Dice	0.876	0.880
SSIM	0.934	0.946

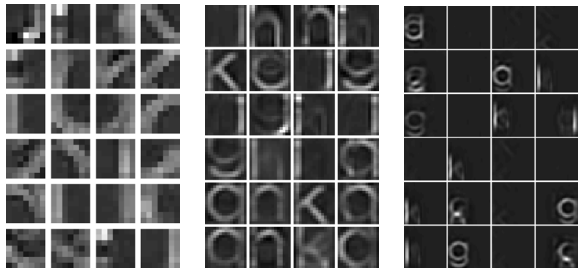
The proposed method can be considered as a learning approach in which the features are learned directly from 3D volume data. One problem commonly occurs during the learning process is that the solution may converge to a local minimum. The converged result we expect is that each atom in the dictionary is used as evenly as possible and the sparse maps are as sparse as possible. However, there might be a case when only a few atoms are used to approximate the input data. In that case, only a few sparse maps have largely non-zero values and the rest are all zero maps. To avoid such cases, we propose *blackout* regularization. The main idea is that we choose portion q of K atoms, in which $0 < q < 1$, and delete their corresponding sparse maps during the dictionary training process. This is similar to *dropout* regularizer [30] in machine learning to avoid overfitting.

For this, we introduce two strategies to give major improvements over conventional dictionary training – maximal blackout and random blackout. Maximal blackout is selecting the q highest of the K atoms in terms of the energy in the corresponding sparse map, and discard them to avoid the case that only a few atoms dominate to approximate the target data. The other strategy, random blackout, is selecting q sparse maps randomly and discard, which relies on the probabilistic regularization. We used $q = 12.5\%$ in all cases of our experiments. However, neither strategy emerged as significantly better than the other. Therefore, we combined both strategies in the dictionary training process.

Figure 4 depicts two renderings of synthetic alphabet dataset: one is of the feature enhanced volume derived from regular dictionary, the other is from the dictionary that has been applied blackout regularizers. As shown in this figure, the blackout result tends to visualize the letters better than the result without blackout because the characters “c”, “g” and “h” have misclassified their boundaries and the corresponding rendered volume has contained more noise artifacts. For quantitative comparison of Figure 4, we used two error metrics: the Dice coefficient and the SSIM of rendered results. In both metrics, the uses of blackout(s) gave higher accuracy as shown in Table 1.

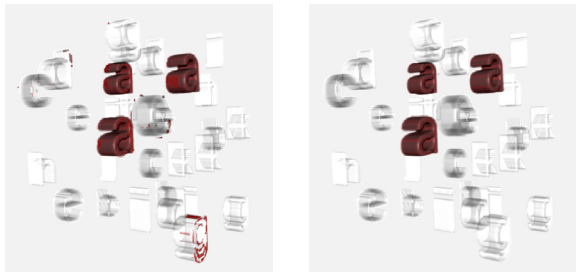
4.3 Constructing High-dimensional Feature Vectors using Hierarchical CSC

Once we have the method to solve the 3D CSC problem as introduced in Section 4.2, we extend the method to handle multi-scale dictionary, called *hierarchical CSC*. The main idea of this method is that the features learned by the CSC is defined by the size of the filter (atom). Figure 5 shows an example of multi-scale dictionaries learned from the synthetic alphabet volume data. The filter size is in three different scales ($7\times7\times7$, $15\times15\times15$, and $31\times31\times31$, respectively). As can be seen, the smaller filter captures the edge-like small features, but the larger filter captures the letter’s entire shape. Therefore, if we construct



(a) $7\times7\times7$ (b) $15\times15\times15$ (c) $31\times31\times31$

Fig. 5: Multi-scale dictionaries learned from the alphabet dataset. Note that the small filter (a) learned edge-like local features, the medium size filter (b) learned the fraction of alphabet shapes, and the large filter (c) learned the entire alphabet shapes.



(a) Single-scale rendering (b) Three-level rendering

Fig. 6: Volume rendering of the alphabet dataset using (a) single-scale dictionary, and (b) three-level multi-scale dictionary. In (a), some edges of the letters ”c” and ”g” are misclassified as ”a” because their round shape looks similar, whereas in (b) only ”a” letter is selected due to the higher accuracy of the multi-scale dictionary.

the dictionaries with the filters in different sizes, then we can encode much richer feature information in the dictionary, which will eventually affect the classification accuracy (see Figure 6 and Table 2). Another important idea is to make the dependency across filter levels. This idea is similar to the structure of the convolutional neural network (CNN) where the neuron activation from the previous level is used as the input to the convolutional layer in the following level. Table 2 shows that hierarchical dictionary is more accurate than multi-scale dictionary without hierarchical dependency.

Based on these ideas, we propose the hierarchical CSC (Figure 7) in which the dictionary and the sparse maps are successively constructed for every level using the result of the previous level. For example, the input volume to the current level $m+1$ is the approximation $a^m = \sum_k d_k^m * x_k^m$ from the previous level m. Whenever the level increases, we enlarge the filter size but keep the volume size the same so that the filters can extract features in different scales. After collecting the multi-scale dictionaries and their corresponding sparse maps, we construct a set of the response maps $\{r_k^m\}$ by convolving the filters $\{d_k^m\}$ with the sparse maps $\{s_k^m\}$. Thus, the number of generated response maps depends on the dictionary size (i.e., the number of atoms) and the number of levels. These response maps show the spatial extent of each filter contributing to the input approximation a^m. In our experiment, we used three levels ($m = \{1,2,3\}$, the filter size for each level is $7\times7\times7$, $15\times15\times15$ and $31\times31\times31$, respectively), and each level generates 24 filters and sparse maps ($k = \{1,...,24\}$), as follows:

$$v^1 \approx \sum_{k=1}^{24} d_k^1 * x_k^1 = \sum_{k=1}^{24} r_k^1 = a^1, \quad v^2 = a^1 \tag{3}$$

$$v^2 \approx \sum_{k=1}^{24} d_k^2 * x_k^2 = \sum_{k=1}^{24} r_k^2 = a^2, \quad v^3 = a^2 \tag{4}$$

$$v^3 \approx \sum_{k=1}^{24} d_k^3 * x_k^3 = \sum_{k=1}^{24} r_k^3 = a^3 \tag{5}$$

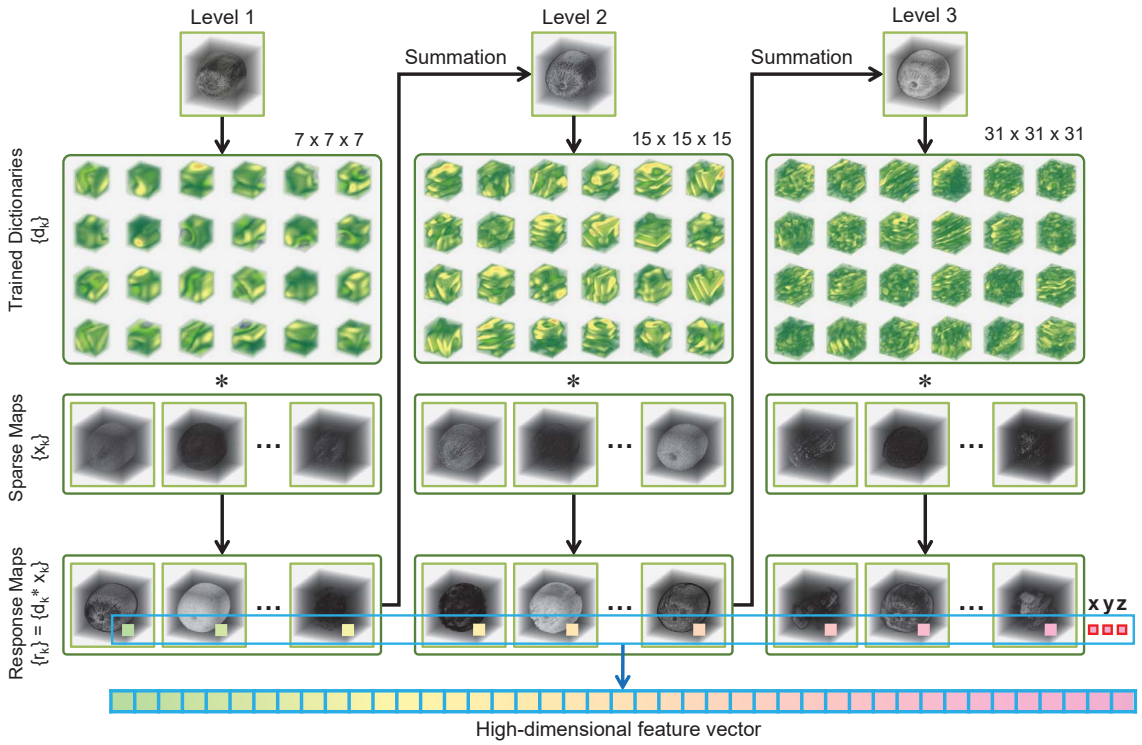


Fig. 7: Constructing high-dimensional feature vectors using hierarchical 3D convolutional sparse coding

Table 2: Accuracy comparison of dictionaries of different types.

Metrics	Single-scale	Multi-scale	Hierarchical
Dice	0.900	0.945	0.997
SSIM	0.961	0.976	0.977

Note that the choice of the number of atoms and levels is problem-specific and empirically chosen, meaning that the number of levels is chosen based on the input volume size and the number of atoms is chosen to allocate the largest number of atoms fit to the available memory. Once we generate the multi-scale hierarchical dictionary, we finally generate the high-dimensional feature vector per voxel which will be used as the input for classification. The high-dimensional feature vector is the collection of the response map value at a given voxel location. Since we have three levels in which each level consists of 24 response maps, we have total 72 response values per voxel. The final per-voxel feature vector becomes 75 dimensions by appending the spatial location (x, y, z indices of each voxel location). Note that, unlike the CNN that reduces the image size by half using max-pooling between levels, the volume size in the hierarchical CSC must be the same across all levels because we collect the response values (i.e., $d_k^m * x_k^m$ for all k and m) for every voxel location to construct the feature vector.

4.4 Voxel Classification using High-dimensional Feature Vectors

Regression: Our volume renderer allows users to draw strokes and scribbles directly in the volume in order to select the structures or data classes to visualize. Figure 8 depicts the interface of our painting application. Pixels can be selected using a region-growing method, similar to magic wand tools in current commercial design software. Once the painting procedure is completed on N classes of the desired materials, the application collects the feature vectors that correspond to the manually labeled voxels from the user's scribbles and feeds them through a classifier for the on-the-fly regression training.

Many types of classifiers can be considered, as discussed in [29].

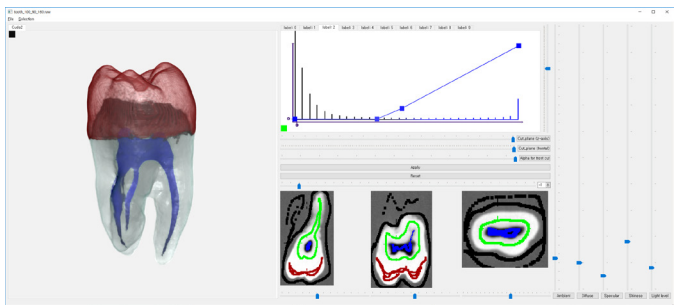


Fig. 8: Volume rendering and user interface window. The user can use a paint-brush tool to select voxels on the x-y-z plane view (bottom center, three black windows with selections). Each label is assigned with a one-dimensional transfer function (top center).

We empirically observed that the random forest algorithm [2], an ensemble machine learning approach that leverages multiple decision trees, outperformed the multi-layer perceptron neural network, which is in agreement with Soundararajan and Schultz [29]. In addition, its training time was also short compared to other learning-based classifiers. Therefore, we chose the random forest for our volume rendering system.

Classification: Once the classifier, that is, the random forest, has been trained appropriately, it can be used to predict the multinomial classification of the entire volume. The score of this one-time predictive procedure is used to assign labels that have the highest probability for all the volume data. The input in the random forest model are the full 75-dimensional feature vectors that have been generated before. The results will be the expected class of the individual voxels with the highest probability among the materials of interest. This enables us to compose the volume rendering by building up multiple per-class one-dimensional transfer functions and applying the ray casting algorithm thereafter. The running time of random forest is acceptable for reasonably sized

volumes, as shown in Table 3. We expect further speed up by leveraging the GPU for training the random forest classifier, which is left for the future work.

Table 3: Running time of the voxel classification method.

Data	Number of voxel selection	Number of labels	Total time (s) (Train+Deploy)
Spiral 128×128×128	9378 (1 slice)	2	0.420
	100775 (6 slices)	2	4.690
CT-Tooth 100×90×168	19623 (2 slices)	4	0.698
	57900 (6 slices)	4	1.739
MRI-Kiwi 256×256×256	63340 (1 slice)	6	16.400
	167973 (3 slices)	6	19.174

4.5 Rendering Multi-labeled Volume using Probabilistic Transfer Function

To obtain the final rendering, the ray casting algorithm is employed to project the entire view frustum on to the two-dimensional screen. Along each ray, color and opacity values are sampled and combined using front-to-back composition as follows:

$$Color = Color + v.Color \times v.Alpha \times (1 - Alpha) \quad (6)$$

$$Alpha = Alpha + v.Alpha \times (1 - Alpha) \quad (7)$$

where $v.Color$ and $v.Alpha$ are the color (possibly shaded using Phong shading [27]) and opacity at the sampling location v. This composition continues until the ray exits the volume or the accumulated alpha value becomes close to zero. Each $v.Color$ and $v.Alpha$ are defined by the probabilistic transfer function as follows:

$$v.Color = \sum_n TableColor(n, v.Prob_n) \times v.Prob_n \quad (8)$$

$$v.Alpha = \sum_n TableAlpha(n, v.Prob_n) \times v.Prob_n \quad (9)$$

where n is the label index, $v.Prob_n$ is the probability of the label n at the location v, and $\sum_n v.Prob_n = 1$. $TableColor(n,x)$ or $TableAlpha(n,x)$ is the user-defined 1D transfer function for the label n that maps the probability value x (between 0 to 1) to a color or an alpha value, respectively. Note that conventional methods [29] use single color and alpha value per label for simplicity, e.g., $TableColor(n,x) = LabelColor(n)$ for all x, and we can employ a similar approach by using a pre-defined linear transfer function by default. However, our probabilistic 1D transfer function allows more flexible, fine-tune control of intra-label color-alpha value assignment. In addition, since we assign a curve in the 1D transfer function that smoothly interpolates the probability values, partial volume effects (or blocky artifacts) can be effectively reduced.

5 Results

The most time-consuming process in the proposed system is solving the 3D CSC problem. To accelerate this process, we used a multi-GPU computing server equipped with NVIDIA Titan X GPUs. The interactive visualization component is implemented and executed on a PC equipped with an Intel i7 CPU with a 12 GB main memory and an NVIDIA GTX Geforce 980 Ti GPU, offering a paintbrush user interface, on-the-fly random forest training and deployment for classification, and ray-casting volume rendering. For implementation of the random forest, we used the open-source Python library [25].

5.1 Visualization of 3D Dictionaries

We extracted features from several public domain 3D volume datasets: MRI-Kiwi, MRT-Aneurysm, CT-Bonsai, low-dose CT-Chest (LdCT-Chest), CT-Tooth, and EM-Mouse. As shown in Figure 9, our method can generate 3D dictionaries from the input volume where each atom in the dictionary represents local feature in the data. For example, the atoms in the dictionary of the LdCT-Chest data represent fine-level details related to bones and surrounding muscle tissues. Similar results can be found in other datasets as well. They generate the data-dependent learned basis, which differs significantly from the universal dictionary in the Fourier, cosine and wavelet transforms.

5.2 Comparison with Other Methods

In this section, we compare our method with Kniss et al. [17] and Soundararajan and Schultz [29]. We implemented and extended their method to multi-material classification. Moreover, we used the same setup for random forest parameters and user scribble selection to make fair comparisons.

Synthetic volume data: We assess the robustness of our algorithm by comparing the rendering on the noisy synthetic dataset (Figure 10). We created a phantom volume of a spiral structure with random noise. As shown in this figure, our method is robust to noise and renders the entire spiral shape faithfully while the other methods incompletely render the structure of spiral shape or pick up the noise artifacts in the background. This is because our method uses feature vectors from learned dictionary which is less susceptible to noise.

Real volume data: The top row of Figure 11 presents the rendering results of the MRI-Kiwi dataset in the comparison of Kniss et al. [17] and Soundararajan and Schultz [29], and ours, respectively. We drew scribbles on the x-y plane to collect the training data for multiple classes: the background, the carpel parts (locules filled with juice), the central placenta vascular, the lateral branch, the placental vascular bundles, and the seeds. Kniss et al. [17] captured many false positives surrounding the seed regions and improperly recognized the septum vascular bundles. Although Soundararajan and Schultz [29] showed generally good separation of the carpel region, the peel classification was not satisfactory. Our rendering, on the other hand, offers the best result, as it can clearly separate each structure with minimal errors. Similar advantages of our method are also achieved with the other datasets (see the rest of Figure 11) with which the multi-class material separation was performed. In the MRI-Aneurysm dataset, we intentionally applied two labels, one is on the thicker trunk and the other is on the thinner tubes. Kniss et al. [17] failed to separate the structure, while Soundararajan and Schultz [29] had some errors in thinner tubes near the main trunk (see the zoom-in circle). On the other hand, our method was able to separate two regions cleanly, which is due to the multi-scale dictionary that learns features in different sizes. The bottom of Figure 11 presents the results of the EM-mouse dataset with many artifacts and complicated structures. As can be seen, our method is highly robust to noise, whereas the others incorrectly picked up noise outside. In addition, thanks to the learned features, the myelinated axons are well-separated from the regular axons because different membrane thickness can be recognized and classified differently. In summary, our method is more robust to noise and is superior in classifying structures based on their shapes as well as voxel intensity.

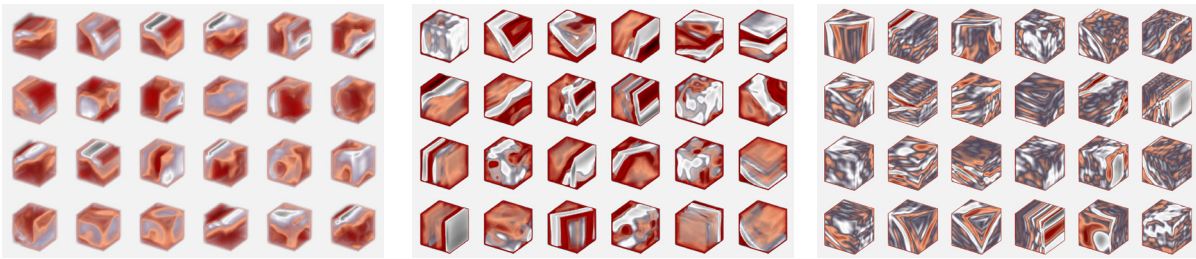
6 User Evaluation

We conducted a user study to evaluate the usability and the rendering quality of the proposed volume rendering method. Specifically, we compared our method with a multi-dimensional transfer function approach [17] and a painting-based intelligent system [29] to demonstrate how the proposed method performs better than conventional volume rendering methods.

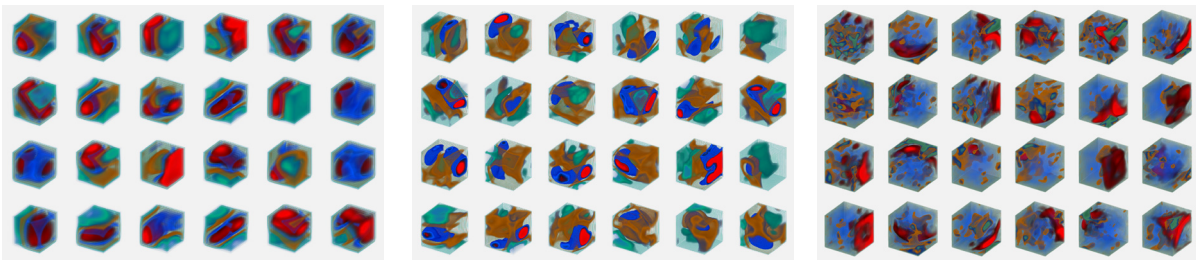
6.1 Experiment Setup

Through a preliminary questionnaire, we recruited 30 participants who had never used volume rendering software but were familiar with computers. Participants were between the ages of 19 and 25, and the number of male and female participants was the same.

In this study, we assessed the usability of the volume rendering methods and the accuracy of the rendered results. For the usability evaluation, we implemented a a 3D transfer function-based user interface [17] and a painting-based user interface employed in both [29] and our method. To evaluate both methods under the same conditions, we used the same ray casting (alpha compositing) and shading algorithms to produce identical rendering effects, and also used the same



(a) Dictionaries of LdCT-Chest dataset



(b) Dictionaries of CT-Bonsai dataset

Fig. 9: Volume rendering of hierarchical multi-scale 3D dictionaries. From left to right: three resolutions of 24 atoms (7^3, 15^3 and 31^3).

Fig. 10: Rendering of the noisy spiral dataset. From left to right: Kniss et al. [17], Soundararajan and Schultz [29], and ours. Our method is robust to noise due to the nature of learned dictionary.

window size for the volume renderer and the transfer function editor for measuring the number of mouse interactions.

We used a Windows PC equipped with an Intel i7-6700 CPU, 64GB of main memory, and an NVIDIA GTX Geforce 1080 GPU for the experiment. Two 27-inch monitors were used to display the volume rendering user interface, the user study manual, and the model answer (the rendered image that the participants were asked to generate). The resolution of each monitor was 1920×1080 running at a 60Hz refresh rate, and the model of the monitor was AOC 2769. The participants used a Microsoft wired desktop 400 keyboard and mouse for the experiment.

6.2 Experiment Procedure

The experiment consisted of three steps: 1) the training step to learn how to use the software, 2) the main experiment step to perform the given tasks, and 3) the post-survey step to evaluate the software subjectively.

Training step. The participants were first asked to watch the instruction video to learn how to use the two volume rendering systems (i.e., the 3D transfer function-based and painting-based systems). The instruction video provided basic knowledge about the software system required to perform the tasks. We chose video-guided training over instructor-guided training because we wanted to provide identical training to the participants without bias. After the training step, the participants were aware that mouse interactions, experiment time, and accuracy are automatically recorded.

Main experiment step. The participants were asked to perform three tasks. Tasks 1 and 2 were designed to assess the usability of the two user interfaces used in 3D transfer function-based volume rendering and painting-based intelligent volume rendering. Task 3 was designed to assess the accuracy of the voxel classification results from three methods: Kniss et al. [17], Soundararajan and Schultz [29], and ours. The input data and their model answer for each task are shown in Figure 12. More details about the tasks are as follows:

- **Task 1.** In the given synthetic data with two separated 3D alphabet letters (Figure 12 (a)), the participants were asked to visualize the letter "a" only (Figure 12 (d)). The total user time spent to make the rendered result close to the model answer of 99% or higher (similarity measured with a Dice coefficient between the ground-truth and classification results).
- **Task 2.** This is similar to Task 1 but more challenging because two letters overlapped (Figure 12 (b) and (e)). We expected that additional user interactions would be required for this task.
- **Task 3.** The participants were asked to separate three structures in the CT-Tooth dataset (Figure 12 (c) and (f)) as close to the model answer as possible. This task had a time limit (10 minutes), and the similarity between the user result and the ground-truth was measured using a Dice coefficient.

6.3 User Evaluation Result

We analyzed the collected experimental data (i.e., time, mouse interactions, and accuracy) using statistical models, such as Fitts' law [22] and the one-way ANOVA [31].

6.3.1 Experiment Time Analysis

Figure 13 (a) shows the running time of the 3D transfer function(3D TF)-based method and our paint-based method for Tasks 1 and 2 using a boxplot. We designed Task 2 more difficult by overlapping two objects, and the 3D TF method shows significant difference between Task 1 and 2 as expected. However, in our method, the running time of Task 2 is within the range of that of Task 1. This is because the users can easily generate fairly accurate results using only a few user-scribbles in our method, so after Task 1 the users become more familiar with the user interface of our method and the user time is rather reduced.

We further analyzed this result using Fitts' law, which shows the relationship between the user time and the usability of the system [22]. Fitts' law is a linear regression model for the user movement time (MT) and the index of difficulty (ID) as follows:

$$MT = a + b * ID \tag{10}$$

where a and b are device-dependent coefficients. Since b affects the user time when the difficulty of the task changes, our strategy derives b

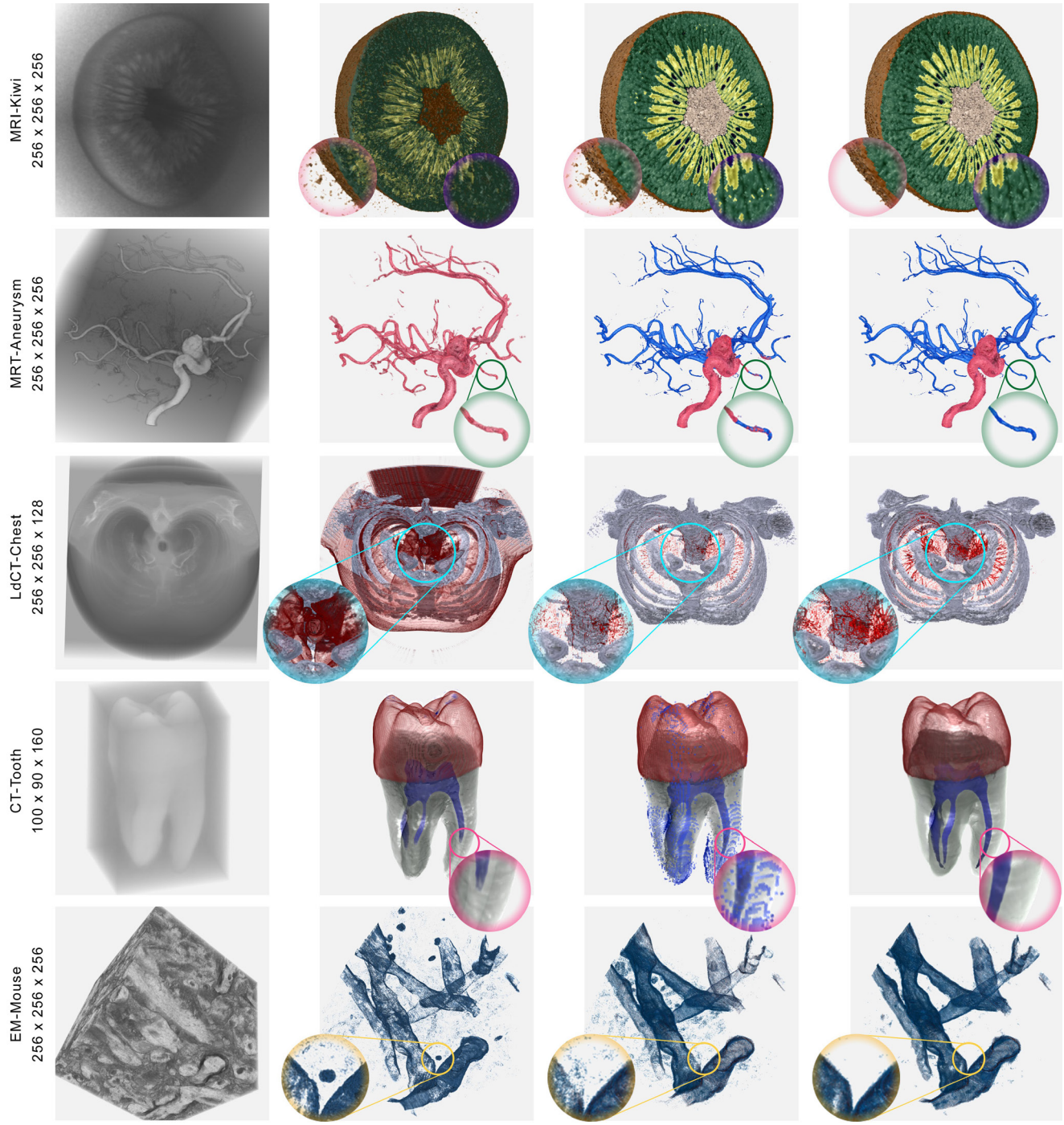
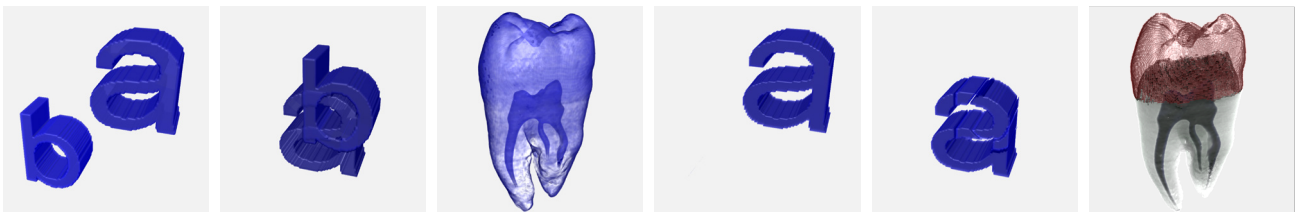
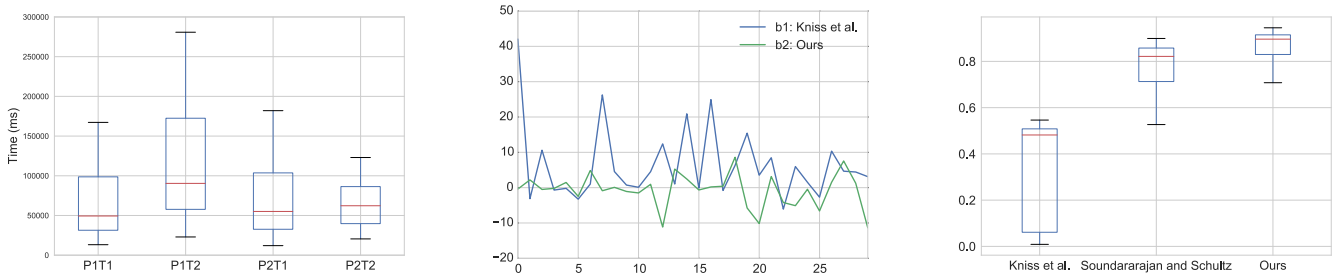


Fig. 11: Comparison on various datasets. From left to right: Grayscale input volumes, Kniss et al. [17], Soundararajan and Schultz [29] and ours.



(a) Task 1 input (b) Task 2 input (c) Task 3 input (d) Task 1 model answer (e) Task 2 model answer (f) Task 3 model answer

Fig. 12: Input data (a – c) and their model answers (d – f) for each task.



(a) (b) (c)

Fig. 13: User evaluation results. (a) : Experiment time of Kniss et al.(P1) and our method (P2) on Task 1 (T1) and Task 2 (T2), (b): Comparison of b values from Fitts' law analysis on Task 1 and 2. b1: Kniss et al., b2: ours, and (c): Accuracy result of Task 3.

by fixing the method and performing two tasks with different difficulty levels, and then analyzing whether there is a statistically meaningful difference between the methods. For this, let us assume the ID and the MT of Tasks 1 and 2 are ID1, ID2, MT1, and MT2, respectively. Then, we can derive b from Equation 10 as follows:

$$b = (MT2 - MT1)/(ID2 - ID1) \tag{11}$$

Figure 13 (b) shows the b values of 3D TF and our method. The average b value for the 3D TF is 6.53041, and that of our method is -0.75062. Our method shows negative b value because the level of difficulties of Task 1 and 2 are not significantly different for our method. Therefore, contrary to our assumption that Task 2 is more difficult than Task 1 ($ID2 > ID1$), the learning effect during the study actually reduced the experiment time of Task 2 (i.e., $MT2 < MT1$).

To show statistically significant differences between two average b values, we further applied one-way ANOVA [31]. The p-value was 0.00102, which was significantly smaller than the significance level of 0.05, and the average b value for our method was significantly smaller than that for the 3D TF. Therefore, we conclude that the usability of our painting-based user interface is much higher compared to that of the conventional transfer function-based user interface.

6.3.2 Accuracy Analysis

Figure 13 (c) shows the accuracy result for Task 3 in which the average accuracy of Kniss et al. [17], Soundararajan and Schultz [29], and our method was 0.45044, 0.73717, and 0.84109, respectively. We used a Dice coefficient [6] to compare the similarity between the ground truth and the user-generated result. The p-value from one-way ANOVA was closed to zero ($p < 0.00001$), which is smaller than the significance level of 0.05; therefore, we confirmed that at least two methods have statistically significant differences. To confirm the difference between our method and the others, we further conducted post-hoc analysis using Fisher's least significant difference (LSD) [34]. Using Fisher's LSD, the significance value of our method over Kniss et al. [17] is closed to zero ($p < 0.00001$) and over Soundararajan and Schultz [29] is 0.04, both are less than 0.05; thus, our method has statistically significant differences. In addition, our approach has the highest average accuracy; thus, we confirmed that our method is the most accurate method among the three.

7 Discussion

Connection to Deep Learning. The proposed hierarchical CSC is designed to mimic multi-scale feature learning in a deep convolutional neural network without the burden of the extensive training usually required in deep supervised learning. In addition, 3D convolutional networks are even more difficult to implement and to train than the proposed 3D CSC due to the increasing size of the network. We believe the proposed hierarchical 3D CSC, alongside with other unsupervised learning techniques, such as convolutional autoencoder, can be an appropriate and practical solution for data-driven interactive volume rendering problems. The proposed hierarchical CSC is also in line with the interesting recent research direction about no-gradient learning [32], which replaces the gradient descent in deep learning with the global energy minimization problem.

Limitations. One limitation of the proposed method is the running time for the dictionary learning and high-dimensional feature construction. Although this task needs to be performed only once during pre-processing (i.e., there is no need to recalculate the dictionary and the response maps when the user changes scribble selections), this step requires approximately 30 minutes to complete on a single CPU core and 10 minutes with GPU acceleration. In addition, increasing the number of filters increases the memory consumption, as the filters and sparse maps are all in 3D. The most memory-consuming process is CSC computation, where all of the intermediate variables and their corresponding forms in Fourier domain as complex numbers (see Supplements) should be in-memory. This consumes roughly $16\times$(# of atoms)$\times$ of the input volume size, for example, the preprocessing of a 256^3 with a 24-atom dictionary would need $16\times24\times256^3\times8$(bytes), or equivalently, 48GB of memory. Once the preprocessing is done, we need (# of atoms $\times$ # of scales)$\times$ of the input volume size to store feature volumes (in unsigned integer data type), which is around 3.375GB for a 256^3 volume. Training and deploying a random forest classifier (with 20000 selected voxels) on a 256^3 (bytes) volume requires about 1.5GB of memory. Due to such memory requirement, the dimension of the largest volume data we used in our experiment is 256^3. Although the time and memory problems are not the limitation of the method but the implementation issues (ADMM-based numerical solvers [28] for Equation 2 are highly parallelizable, see [32]), we plan to address those in the future work.

8 Conclusion

In this paper, we introduced a novel interactive volume rendering framework that leverages machine learning to accurately classify voxels. Inherited from the intelligent system approach in which users simply need to specify materials of interest using rough scribbles, we improved the classification quality by employing a novel high-dimensional feature generated by using a hierarchical 3D CSC. We presented for the first time 3D volume rendering of learning-based multi-scale features that do not rely only on voxel intensity; thus, the proposed method is more robust to noise and is highly discriminative for fuzzy boundaries that results in superior rendering quality.

In the future, we plan to extend the proposed method to large-scale parallel computing systems to handle a much larger 3D volume data and dictionary. We also intend to explore various computer graphics applications that can benefit from the proposed hierarchical CSC, such as mesh segmentation and surface reconstruction. In-depth study of the connections between hierarchical CSC and a deep convolutional neural network would be another interesting future research direction. We believe that the proposed intelligent volume rendering will open up new research opportunities related to deep learning in scientific visualization and computer graphics.

Acknowledgments

This work was partially supported by the National Research Foundation of Korea (NRF) funded by the Korea government (MOE) (NRF-2017R1D1A1A09000841, Basic Science Research Program) and by the Korea government (MSIT) (NRF-2017M3C7A1047904, Brain Research Program).

REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, Nov. 2006.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] H. Bristow, A. Eriksson, and S. Lucey. Fast Convolutional Sparse Coding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 391–398, June 2013.

[4] L.-L. Cai, B. P. Nguyen, C.-K. Chui, and S.-H. Ong. Rule-Enhanced Transfer Function Generation for Medical Volume Visualization. *Computer Graphics Forum*, 34(3):121–130, June 2015.

[5] 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, Nov. 2008.

[6] L. R. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[7] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH, pp. 65–74, 1988.

[8] M. Elad and M. Aharon. Image Denoising Via Learned Dictionaries and Sparse representation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 895–900, June 2006.

[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, Dec. 2011.

[10] H. Guo and X. Yuan. Local WYSIWYG volume visualization. In *Proceedings of IEEE Pacific Visualization Symposium (PacificVis)*, pp. 65–72, Feb. 2013.

[11] M. Hadwiger, J. M. Kniss, C. Rezk-salama, D. Weiskopf, and K. Engel. *Real-time Volume Graphics*. 2006.

[12] F. Heide, W. Heidrich, and G. Wetzstein. Fast and flexible convolutional sparse coding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5135–5143, June 2015.

[13] 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, Dec. 2012.

[14] C. R. Johnson and J. Huang. Distribution-Driven Visualization of Volume Data. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):734–746, Sept. 2009.

[15] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierarchies for visual recognition. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1090–1098, 2010.

[16] G. Kindlmann and J. W. Durkin. Semi-automatic Generation of Transfer Functions for Direct Volume Rendering. In *Proceedings of the IEEE Symposium on Volume Visualization*, VVS '98, pp. 79–86, 1998.

[17] 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, July 2002.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105. 2012.

[19] M. Levoy. Display of Surfaces from Volume Data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, May 1988.

[20] P. Ljung, J. Krger, E. Groller, M. Hadwiger, C. D. Hansen, and A. Ynnerman. State of the Art in Transfer Functions for Direct Volume Rendering. *Computer Graphics Forum*, 35(3):669–691, June 2016.

[21] R. Maciejewski, Y. Jang, I. Woo, H. Jnicke, K. P. Gaither, and D. S. Ebert. Abstracting Attribute Space for Transfer Function Exploration and Design. *IEEE Transactions on Visualization and Computer Graphics*, 19(1):94–107, Jan. 2013.

[22] I. S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Hum.-Comput. Interact.*, 7(1):91–139, Mar. 1992.

[23] C. Osendorfer, H. Soyer, and P. v. d. Smagt. Image Super-Resolution with Fast Approximate Convolutional Sparse Coding. In *Neural Information Processing*, number 8836, pp. 250–257. Nov. 2014.

[24] S. Park and C. L. Bajaj. Multi-dimensional transfer function design for scientific visualization. In *Proceedings of the Fourth Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP)*, pp. 290–295, 2004.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] H. Pfister, W. E. Lorensen, C. L. Bajaj, G. L. Kindlmann, W. J. Schroeder, L. S. Avila, K. Martin, R. Machiraju, and J. Lee. The transfer function bake-off. *IEEE Computer Graphics and Applications*, 21(3):16–22, 2001.

[27] B. T. Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, June 1975.

[28] T. M. Quan and W.-K. Jeong. Compressed sensing dynamic MRI reconstruction using GPU-accelerated 3D convolutional sparse coding. In *Proceedings of Medical image computing and computer-assisted intervention (MICCAI)*, pp. 484–492, 2016.

[29] K. P. Soundararajan and T. Schultz. Learning Probabilistic Transfer Functions: A Comparative Study of Classifiers. *Computer Graphics Forum*, 34(3):111–120, June 2015.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[31] B. G. Tabachnick and L. S. Fidell. *Using Multivariate Statistics (5th Edition)*. 2006.

[32] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein. Training neural networks without gradients: A scalable ADMM approach. In *Proceedings of the International Conference on Machine Learning, ICML 2016*, pp. 2722–2731, 2016.

[33] F. Y. Tzeng, E. B. Lum, and K. L. Ma. An intelligent system approach to higher-dimensional classification of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):273–284, May 2005.

[34] L. J. Williams and H. Abdi. Fishers least significant difference (LSD) test. *Encyclopedia of research design*, pp. 1–5, 2010.

[35] B. Wohlberg. Efficient convolutional sparse coding. In *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7173–7177, May 2014.

[36] B. Wohlberg. Efficient algorithms for convolutional sparse representations. *IEEE Transactions on Image Processing*, 25(1):301–315, Jan 2016.

[37] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus. Deconvolutional networks. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2528–2535, June 2010.

[38] Y. Zhou, H. Chang, K. Barner, and B. Parvin. Nuclei segmentation via sparsity constrained convolutional regression. In *Proceedings of IEEE International Symposium on Biomedical Imaging (ISBI)*, pp. 1284–1287, Apr. 2015.

[39] Y. Zhou, H. Chang, K. Barner, P. Spellman, and B. Parvin. Classification of Histology Sections via Multispectral Convolutional Sparse Coding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3081–3088, June 2014.