

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/322612433>

# Deep-Learning-Assisted Volume Visualization

Article in IEEE Transactions on Visualization and Computer Graphics · January 2018

DOI: 10.1109/TVCG.2018.2796085

---

CITATIONS

46

READS

821

7 authors, including:



Hsueh YA Chien

beautynology

15 PUBLICATIONS 342 CITATIONS

[SEE PROFILE](#)



Kedar Narayan

NCI-Frederick

152 PUBLICATIONS 2,218 CITATIONS

[SEE PROFILE](#)



Sriram Subramaniam

University of British Columbia

325 PUBLICATIONS 16,772 CITATIONS

[SEE PROFILE](#)

# Deep-learning-assisted Volume Visualization

Hsueh-Chien Cheng, Antonio Cardone, Somay Jain, Eric Krokos, Kedar Narayan, Sriram Subramaniam,  
Amitabh Varshney *Fellow, IEEE,*

**Abstract**—Designing volume visualizations showing various structures of interest is critical to the exploratory analysis of volumetric data. The last few years have witnessed dramatic advances in the use of convolutional neural networks for identification of objects in large image collections. Whereas such machine learning methods have shown superior performance in a number of applications, their direct use in volume visualization has not yet been explored. In this paper, we present a deep-learning-assisted approach that uses a convolutional neural network to generate high-level, data-driven representations to distinguish various complex structures, which are challenging for conventional approaches based on handcrafted features calculated locally. A significant challenge in designing volume visualizations based on the derived high-dimensional deep features lies in efficiently handling the immense amount of information that deep learning methods provide. To this end, this paper presents two contributions: 1) a technique that uses spectral methods to organize the high-dimensional features into an accessible form that facilitates users' interactions with the visualization through a simple design widget, and 2) a semi-automatic hierarchical exploration of volumetric data. We have validated our approach on two electron microscopy volumes and one magnetic resonance imaging dataset.

**Index Terms**—Volume visualization, convolutional neural networks

## 1 INTRODUCTION

DEEP learning and, more specifically, convolutional neural networks (CNNs) have received much attention in various fields including image classification [1], face recognition [2], [3], and segmentation [4], [5]. Given raw images as input, a CNN progressively builds its final outcome by sequentially propagating responses from one neural network layer to the next. The hierarchical architecture of CNNs mimics the multiscale process of human visual system that aggregates lower-level signals into higher-level concepts. A CNN can therefore learn, by iteratively fitting the network to the data, useful application-dependent representations [2], [6] that facilitate the search for solutions to complex problems.

The success of volume visualization relies heavily on appropriate voxel representations. Creating volume visualizations in general requires three steps, each involving various degrees of human intervention. First, users define the criteria (e.g. intensity and texture) that distinguish the structures of interest (e.g. soft tissue and bone). Second, based on these user-defined criteria, users choose a suitable voxel representation that forms the feature space to search for solutions (i.e. visualizations). Finally, users interact with the visualization tool to create and modify the solutions. In the past, handcrafted features that correspond to specific user-defined criteria (e.g. size [7], [8], texture [9], and visibility [10]) have been successfully applied as the voxel representation in the second step. Nevertheless, as the complexity of criteria grows, finding features that precisely describe the characteristics of target structures is increasingly challenging. Conventional handcrafted features no longer suffice because they are defined locally without

addressing the relationships among voxels in a larger context, which is crucial to characterizing complex structures.

Although the high structural complexity significantly hinders the manual search for suitable feature spaces in the second step of the current visualization workflow, users can still provide valuable domain knowledge in a different way. From a machine learning point of view, the criteria that distinguish different structures can be implicitly defined by a large number of examples that include as many structural variants as possible. In this work we train a CNN as if we were solving a classification problem based on the given examples. After training, the CNN automatically derives a feasible high-level data representation, thus creating a feasible feature space for the visualization of complex structures. Despite improvements in user interfaces [11], [12], [13] and semi-automatic techniques [14], [15], current methods proceed with the assumption that a suitable feature space is explicitly defined, not automatically learned. In contrast, the proposed deep-learning-assisted approach automatically creates a feasible high-dimensional feature space in a data-driven way without explicit specifications from users.

In the derived feature space, voxels with similar characteristics are in close proximity. These similar voxels can therefore be described and selected for visualization with a representative characteristic feature vector, which essentially defines a point in the high-dimensional space. Nevertheless, the characteristic feature vector will soon become too complicated to maneuver as its dimensionality grows. Previous work alleviates this problem by designing visualizations in a reduced space with manageable complexity [12], [16]. In this work we present two techniques for this usability problem. The first one reorders the features with respect to their similarity. With this improved, similarity-aware feature layout, the conventional design widget can select groups of similar voxels in the original space without reduction, thus providing a complementary strength to the dimension reduction-based techniques. The second one is a semi-automatic technique that generates a hierarchy of volume visualizations for users to choose from, thus providing another layer of abstraction on top of

- H.-C. Cheng, S. Jain, E. Krokos, and A. Varshney are with the Department of Computer Science, University of Maryland, College Park, MD, 20742. *E-mail:* {cheng, somay, ekrokos, varshney}@cs.umd.edu
- A. Cardone and A. Varshney are with the University of Maryland Institute for Advanced Computer Studies, College Park, MD, 20742. *E-mail:* {cardone, varshney}@umiacs.umd.edu
- K. Narayan and S. Subramaniam are with the Center for Molecular Microscopy, NCI, NIH, Bethesda, MD, 20850. *E-mail:* {narayank, subramas}@mail.nih.gov

the high-dimensional feature space.

In addition to the aforementioned usability issue, we follow the indirect volume rendering framework and design a method to extract surfaces from voxels defined in a high-dimensional feature space. We first convert the feature space to a binary scalar field and then extract a triangular mesh from that binary scalar field using the conventional marching cubes algorithm [17]. Similar to the conventional method, the binary scalar field is obtained by thresholding an intermediate scalar field, which is controlled by users through the design widget. This conversion can be done efficiently in a GPU with little computational overhead.

Although modern rendering techniques and hardware can now render volumetric data interactively, we still need a suitable feature space that differentiates target structures and an easy way for users to interactively build the solutions. Whereas there will always be a need for users to adjust the visualization results for visual appeal manually, this paper presents our initial explorations in using deep learning methods to assist the design of volume visualizations. We expect that the techniques proposed in this work will be useful in many visual computing tools in creating informative visualizations.

In summary, the contributions of this work are:

- A deep-learning-assisted approach to automatically derive high-level features based on information extracted from a larger context than conventional local features.
- Based on the high-dimensional abstract feature space, a feature reordering technique that groups similar features together to facilitate manual visualization design and a semi-automatic technique to allow hierarchical exploration.
- A method that converts a high-dimensional feature space to a binary scalar field that can be used as the input to the conventional marching cubes algorithm.

In this paper, we first review the existing work on CNNs and volume visualization design in Section 2. In Section 3 we introduce major components in our visualization tool, starting from extracting useful high-level features with a trained CNN to the marching cubes-based rendering, feature reordering, and semi-automatic exploration. In Section 4 we present the results of applying our method to three volumetric datasets. Finally, we summarize our approach and provide future research directions in Section 5.

## 2 RELATED WORK

### 2.1 Convolutional neural networks (CNNs)

The basic components of a conventional feedforward CNN include convolutional layers, spatial pooling layers, and fully-connected layers. A convolutional layer filters an input tensor with trainable kernels, which combine neurons in a fixed receptive field. The size of receptive fields directly affects the number of trainable parameters. A spatial pooling layer outputs an aggregation of nearby neurons with a predefined operator. For example, a max-pooling layer outputs the maximum value within a small neighborhood. A fully-connected layer connects every input neuron with the output neurons, disregarding their spatial proximity. The raw output values of neurons in convolution layers and fully-connected layers are usually transformed by an activation function (usually non-linearly) into the result signals, representing the activation level of neurons.

AlexNet [1] is one of the most recognized contributions in image classification and has become the standard benchmark algorithm. AlexNet contains five convolutional layers followed by two fully-connected layers, and three max-pooling layers mixed in between. The last fully-connected layer then connects to the output consisting of 1000 neurons, one for each target class.

Most ordinary CNN architectures more or less resemble the basic construct of AlexNet but differ in depth. For example, GoogLeNet [18] has 22 trainable layers, much deeper than AlexNet, which has seven trainable layers. At this scale of depth, kernel sizes have to be kept small to maintain a manageable model complexity. This limitation in kernel size does not restrict the power of CNNs. In fact, deliberately stacking multiple convolutional layers with small kernels leads to a large receptive field, which is comparable to that of a large kernel but with fewer parameters [19]. Although research in complex problems tends to create deeper networks, simply stacking convolutional layers in a CNN does not always lead to better performance because of the increased model complexity. Addressing the deficiency of conventional CNNs, a 152-layer residual network [20] shows significant improvement over the state-of-the-art architectures.

Given a sequence of training examples, the training procedure starts with a forward pass through the network, calculates a loss value with respect to certain criteria, and then adjusts the weights associated with the neurons in the network accordingly by the backpropagation algorithm. Training a CNN appropriately can be challenging because of problems such as overfitting and premature convergence, which deteriorate performance significantly. Dropping connections between random neurons (i.e. removing them from the network temporarily) avoids co-adaptations among them [21], thereby reducing the chance of overfitting but also slowing down convergence. The slow convergence of CNNs is partly due to the frequent changes in the distribution of a layer’s input during training. Batch normalization [22] addresses this problem by normalizing each feature independently to have a zero sum and unit variance distribution.

### 2.2 Volume visualization

Typical volume rendering techniques can be grouped into two categories: direct and indirect volume rendering. Direct volume rendering (DVR) techniques, such as volume ray-casting [23], compose a result image by aggregating the colors and opacities of relevant voxels calculated using a user-defined transfer function. Indirect volume rendering (IVR) techniques first generate geometric primitives as an intermediate representation and then render those primitives using conventional 3D computer graphics. For example, the marching cubes algorithm [17] extracts a triangular mesh that represents an isosurface in a structured grid.

Despite different ways in composing result images, DVR and IVR techniques all require a feature space suitable for the visualization criteria. Simple structures that are distinguishable by intensity values can be extracted and visualized as corresponding isosurfaces [17] following the IVR framework. Nevertheless, in many applications the complex structures and their surroundings are indistinguishable based on a single isovalue. In these cases the target structures may be better extracted as subvolumes composed of voxels with intensity values contained in an interval [24], [25].

Whereas typical IVR techniques extract geometric primitives from a scalar field formed by intensity values, DVR techniques rely on a transfer function, sometimes defined in an alternative

feature space derived from intensity, to assign distinct colors and opacities to different structures. Common features such as gradient magnitude [11] and curvature [26] allow distinguishing regions with significant local changes in intensity. Other specialized features are used for structures with specific characteristics. For example, twenty texture features have been used to identify voxels with texture differences [9]. Evaluating the size of structures in scale-space at each voxel creates a scale field that can be used to highlight structures of different sizes [7]. An alternative way to assess structure sizes is by searching in all directions for neighboring voxels with similar intensities [8].

Whereas most volume visualizations are done manually, semi-automatic techniques automate specific time-consuming procedures to accelerate the design. Candidate isosurfaces that correspond to meaningful structure boundaries can be selected automatically based on gradient magnitudes [27]. Exploiting the proximity of voxels in the feature space, many semi-automatic techniques use clustering algorithms to group similar voxels, thus reducing the complexity of visualization design [14], [15].

A prerequisite of successful volume visualizations, including those created by semi-automatic techniques, is that the selected feature space must differentiate the target structures. Finding such a feature space is therefore critical for volume visualization design. Nevertheless, existing feature spaces, including those formed by texture features, are too local to reliably differentiate complex structures when given only limited context around each voxel. In this work, we address this limitation by forming a feature space based on high-level features extracted automatically from a trained CNN. Based on the derived feature space, we then create visualizations following the IVR framework.

### 3 METHODOLOGY

#### 3.1 Motivation

Visualization is crucial to the analysis of volumetric data yet its design remains labor-intensive. Conventional design workflow assumes a feasible feature space with which users, after some trial and error, will eventually find configurations suitable for the target structures. Nevertheless, so far the feature space is always hand-crafted with respect to specific application-dependent criteria. This manual procedure limits the feature space to incorporate only local features because of the limited capability of human to comprehend and integrate non-local properties defined in high-dimensional spaces. As structural complexity grows, the assumption of having a feasible feature space is increasingly unrealistic. When the visualization result does not suffice, users are faced with the non-trivial choice to modify the feature space, the configuration, or both in the worst case (Figure 1a).

This observation motivates our deep-learning-assisted approach that learns a feasible feature space automatically when provided enough examples of the target structures. This approach adopts a workflow that effectively replaces the manual search of a feasible feature space by the concrete labeling of samples followed by deep learning (Figure 1b). In contrast to crafting features in a high-dimensional abstract space, the labeling task is much easier for typical users, who are domain experts familiar with the data but have limited knowledge about computational techniques.

##### 3.1.1 Conventional volume visualization design

With imaging modalities such as X-Ray and computed tomography, structures of different intrinsic material densities (e.g. soft

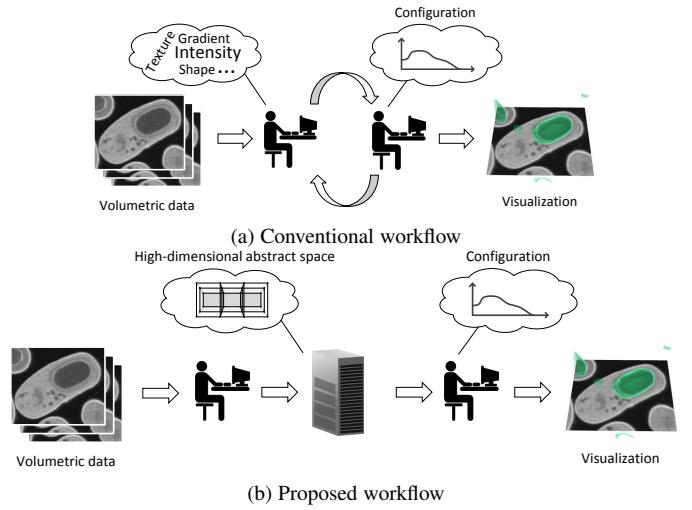


Figure 1. (a) Conventional workflow requires users to adjust both the feature space and the configuration. (b) Our deep-learning-assisted approach derives from labeled examples a feasible feature space automatically, effectively removing the need for an user-defined feature space.

tissue and bone) are distinguishable by their distinct intensity values. This characteristic allows conventional intensity-based feature spaces to separate these structures effectively. For example, the boundaries of bone may correspond well to a certain isosurface. With other modalities, such as electron microscopy, the type of structures to which a voxel belongs is seldom decided solely by the voxel’s intensity value without consulting the arrangement of the neighboring voxels. High intra-class variations further complicate the decision. In addition, gradient only describes local changes in intensity values and lacks the capability in depicting the boundary of complex structures. As a result, intensity and gradient do not provide enough information for designing informative visualizations for complex structures.

For example in Figure 2 we compare the meshes formed by the groundtruth labels of mitochondria (Figure 2b), generated using the conventional marching cubes algorithm (Figure 2c), and generated using our deep-learning-assisted approach (Figure 2d) for the hippocampus dataset (c.f. Section 4.2). Although the mitochondria are typically low in intensity, many non-mitochondria voxels, for example those belonging to the membranes, are also low in intensity (Figure 2a). As a result the isosurface of isovalue 112, which is close to the average intensity value of mitochondria, does not map precisely to boundaries of mitochondria, thus creating an incomprehensible visualization with severe occlusions of various structures. In contrast, our derived feature space leads to a visualization of mitochondria comparable to the groundtruth labels.

The previous example shows the deficiency of conventional feature spaces in depicting complex structures. In fact, a significant overlap of the two types of voxels (i.e. mitochondria and non-mitochondria) in the intensity histogram confirms that we cannot obtain clear boundaries using a different isosurface (Figure 3a). Adding gradient magnitude as a second dimension does not improve separation in the density plot either (Figure 3b). Besides intensity and gradient magnitude, Haralick features [28], a set of texture features calculated from a small local neighborhood, also lack the power in depicting mitochondria although they have been successful for other structures in computed tomography

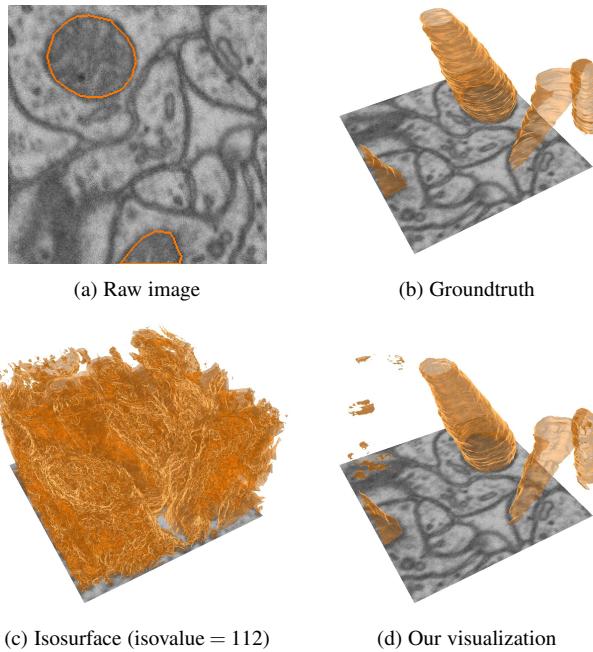


Figure 2. (a) In the hippocampus dataset, we find it challenging to differentiate mitochondrial regions (orange) from non-mitochondrial regions using conventional intensity-based feature spaces. (b) The groundtruth mitochondria. (c) The isosurface of isovalue 112, which is the average intensity value of mitochondria, does not correspond well to the boundary of mitochondria. (d) Our visualization result is comparable to the groundtruth because it is based on a feasible feature space derived automatically.

and magnetic resonance images [9]. Here we calculate seven Haralick features (i.e. energy, inertia, inverse difference moment, entropy, correlation, contrast, and sum of entropy) with the same configuration used in [9]. After projecting the voxels onto 2D using principal component analysis, the large overlapping near the center shows that the selected Haralick features do not separate the mitochondria from the rest of the image (Figure 3c).

### 3.1.2 Deep-learning-assisted visualization design

Previous studies on the hippocampus dataset have shown that features depicting both the complex shape of the mitochondria and the contextual information around voxels perform better in segmentation [29]. If we were to apply the conventional design workflow to create visualizations that show mitochondria and other complex structures presented in this paper, users would need to define such sophisticated features themselves. Although there may exist a combination of novel and established features properly configured for those specific targets, searching manually for that combination is time-consuming and challenging. On the contrary, our proposed approach uses a CNN to perform that search effectively and automatically. Users can therefore focus on creating useful volume visualizations based on the derived feature space.

Our deep-learning-assisted approach extracts high-level features (i.e. deep features) from a trained CNN (top of Figure 4). We first train the proposed CNN as if we were solving a voxel-wise classification problem. Because CNNs are powerful in finding suitable application-dependent features, we use those features to distinguish voxels that belong to various complex structures

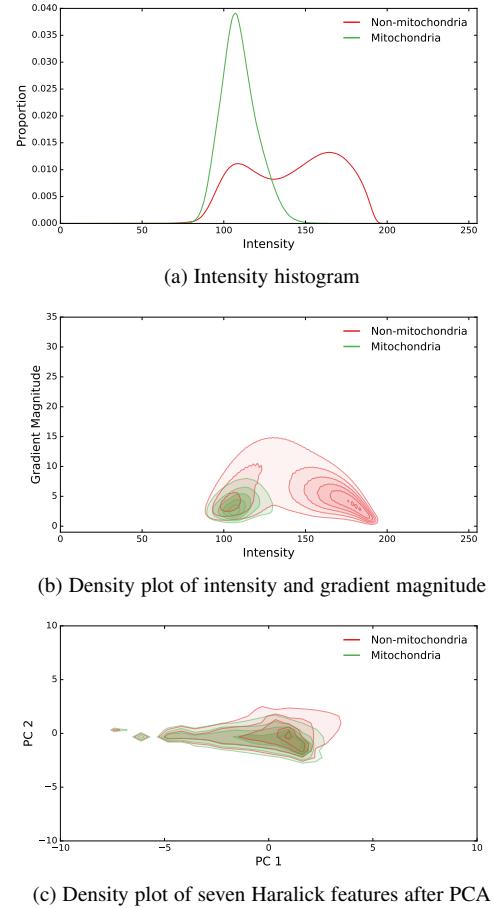


Figure 3. (a) The overlap in the intensity histogram shows that we cannot extract the mitochondria precisely by using a different isosurface. The density plots, in which higher saturation indicates higher density, show that (b) intensity and gradient, and (c) seven selected Haralick features do not help extract the mitochondria either. The seven Haralick features are projected onto the first two principal components, “PC 1” and “PC 2”, obtained using principal component analysis.

during rendering. The extracted deep features correspond to much higher level concept when compared with local features such as gradient magnitude or textures, thus improving their capability in discerning complex structures.

Based on the derived feature space, we address practical issues when creating volume visualizations using deep features (bottom of Figure 4). Even for moderate-sized volumetric data, the high dimensionality of features could be overwhelming because of the limited memory available to a GPU. Therefore, we use vector quantization to compress the features to a manageable size. Furthermore, because the features that form the high-dimensional feature space are not independent, we apply a spectral method to reorder them. After feature reordering users can easily change the visualization by modifying a characteristic feature vector through a simple design widget. We also use a semi-automatic method to accelerate the design of volume visualization by pre-generating a tree of visualizations, with which users can explore the volumetric data hierarchically and interactively.

## 3.2 Learning high-level features

The high-level features are extracted from a CNN, which contains three groups of convolutional layers (Table 1). In each group, we

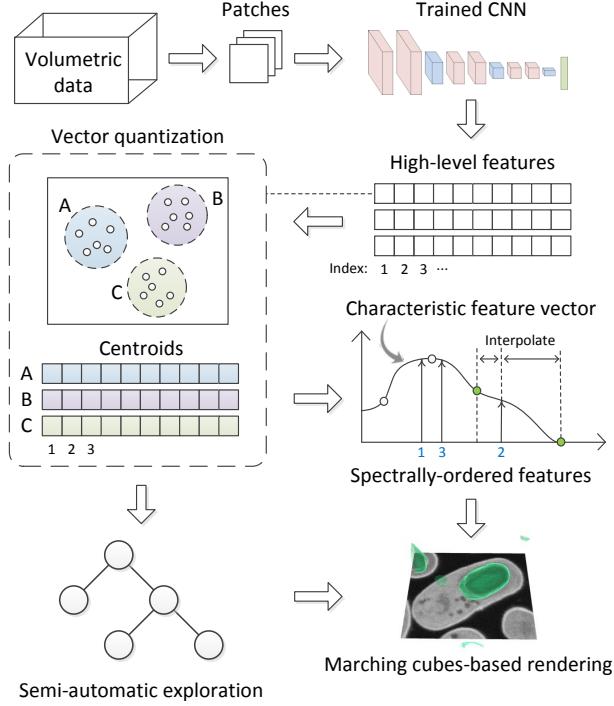


Figure 4. Our proposed deep-learning-assisted approach first extracts high-level features from a trained CNN. We then use vector quantization to encode the extracted high-dimensional features of each voxel by the nearest centroid found using  $k$ -means clustering. Users modify the visualization result, which is generated by a marching cubes-based rendering, either by editing the characteristic feature vector of a target subvolume or exploring a pre-generated subvolume hierarchy semi-automatically.

Table 1  
The architecture of our proposed CNN

| Layer           | #channels | #neurons       | Kernel       | stride |
|-----------------|-----------|----------------|--------------|--------|
| Convolutional   | 64        | $32 \times 32$ | $3 \times 3$ | 2      |
| Convolutional   | 64        | $32 \times 32$ | $3 \times 3$ | 1      |
| Max-pooling     | 64        | $16 \times 16$ | $2 \times 2$ | 2      |
| Convolutional   | 64        | $16 \times 16$ | $3 \times 3$ | 1      |
| Convolutional   | 64        | $16 \times 16$ | $3 \times 3$ | 1      |
| Max-pooling     | 64        | $8 \times 8$   | $2 \times 2$ | 2      |
| Convolutional   | 64        | $8 \times 8$   | $3 \times 3$ | 1      |
| Convolutional   | 64        | $8 \times 8$   | $3 \times 3$ | 1      |
| Max-pooling     | 64        | $4 \times 4$   | $2 \times 2$ | 2      |
| Fully-connected |           | 200            | $1 \times 1$ |        |
| Fully-connected |           | #classes       | $1 \times 1$ |        |

stack two convolutional layers with  $3 \times 3$  kernels to resemble the reception field of a  $5 \times 5$  kernel but with fewer trainable parameters. We perform batch normalization for all the convolutional layers. Also, we use dropout [21] with a probability of 0.5 for the first fully-connected layer. The first fully-connected layer contains 200 neurons, which connect to neurons in the second fully-connected layer. Each neuron in the second fully-connected layer corresponds to an individual class in the data. We chose the values of parameters based on the results of our preliminary study conducted using the target datasets. For other datasets, we may need to adjust these parameters depending on the complexity of target classes for better performance.

### 3.2.1 Convolutional layers

A convolutional layer slides  $k$  trainable kernels  $W_{m,k}$  over the input 3D tensor  $U$  of size  $x \times y \times m$ , where  $x$  and  $y$  correspond to the spatial dimensions, and  $m$  denotes the number of input feature maps (or, channels). The layer outputs a 3D tensor  $V$ , which is connected to the next layer. The  $i$ -th output feature map  $V_i$  is calculated as  $V_i = \sum_m W_{m,i} * U_m + b_i$ , where  $b_i$  denotes the associated bias term. The convolution can be selectively applied to neurons separated by a predefined distance referred to as stride. Here the first convolution uses a stride of two, resulting in a reduction in spatial dimension by half. The output of a convolutional layer is always subjected to an activation function, which we will introduce next.

### 3.2.2 Activation function

An activation function transforms the values in the  $i$ -th feature map  $V_i$  into corresponding activation signals. This bio-inspired function simulates the activation of neurons by electron signals. For all convolutional layers and the first fully-connected layer, we use Parametric Rectified Linear Unit (PReLU) [30], which learns for the  $i$ -th feature map a coefficient  $a_i$  and scales the value by  $a_i$  if the value is negative. When  $a_i = 0$ , PReLU reduces to the conventional rectified linear unit, which sets all the negative values to zero. For the second fully-connected layer, we apply the soft-max function to calculate the predicted probability of voxel classes.

### 3.2.3 Max-pooling layers

Max-pooling layers output the maximum of nearby neurons by scanning the input. This layer is not trainable and is used to reduce the number of parameters in subsequent layers by setting a stride larger than one, thereby reducing the spatial dimension. Here we use max-pooling with a  $2 \times 2$  window and a stride of two to reduce the spatial dimension by half after each group of two convolutions.

### 3.2.4 Training

The training data to the CNN is a  $65 \times 65$  patch centered at each voxel sample selected at random. We draw 400,000 voxels divided equally among the classes to avoid the majority class from dominating other classes during training. We use a batch size of 200 and train a total of 60,000 iterations. After 2,000 iterations the training set is resampled to learn from diverse examples. We use ADADELTA solver [31] with a momentum of 0.9 and a decay of 0.0005 during training.

The input volume is padded with reflection about the edges before generating patches to accommodate for boundary cases where the sampled voxel is near an edge. Because the structures of interest in the target datasets are rotation-invariant, such padding affects minimally the performance of the CNN.

### 3.2.5 Class imbalance

The CNN trained with balanced classes tends to overestimate the probability of the minority class in the testing stage because of the significant disparity in abundance between the training data, which contain equal amounts of samples among voxel classes, and the testing data, which would have the actual class distribution. Such a class imbalance is increasingly problematic as the disparity grows.

Because the degree of class imbalance in the datasets we used is moderate, a simple way to address this problem is by multiplying the predicted probability of each class by the corresponding prior probability, thereby scaling down the predicted

probability of the minority class. We have also implemented the postprocessing technique used in [4], which finds a monotonic cubic polynomial to match the predicted probability and the prior probability. Instead of having a constant scaling factor (i.e. the class prior), this technique transforms the predicted probability in a data-driven way that allows the setting of variable scaling factors with respect to the probability values.

### 3.3 Marching cubes-based rendering

We adapt the conventional marching cubes algorithm [17], which extracts isosurfaces from a scalar field (e.g. intensity), to instead extract surfaces based on a high-dimensional feature space. Given an isovalue, the conventional method converts the input scalar field into a binary one using the isovalue as a threshold, which essentially defines the cut-off value of two target structures. The marching cubes algorithm then iterates over the set of cubes, each represented by the values of its eight bounding voxels in the binary scalar field, and accumulates the corresponding triangles based on the eight bits representation. Because in this work the input is a multiscalar field formed by the high-dimensional feature space, we describe in the following a method that converts the input multiscalar field to a binary scalar field with which the marching cubes algorithm can extract surfaces as usual (Figure 5).

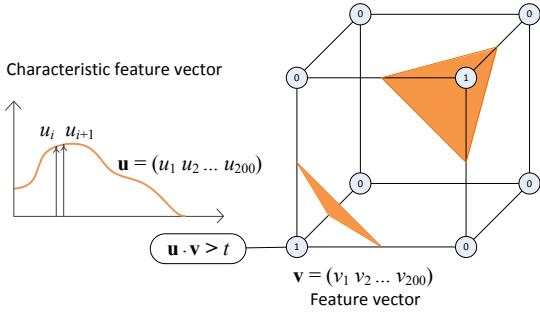


Figure 5. Because the high dimensionality of feature space, we generate a binary scalar field by comparing the dot product of the characteristic feature vector  $\mathbf{u}$  and the feature vector  $\mathbf{v}$  of each voxel with a threshold  $t$ . After that we apply the conventional marching cubes algorithm on the binarized volume.

Assuming that each voxel is represented by a 200-dimensional feature vector  $\mathbf{v} = (v_1 v_2 \dots v_{200})$ , we can decide whether two voxels belong to the same structure by calculating their dot product and compare it with a threshold  $t$ . Following the same intuition, we can also decide whether a voxel belongs to a user-defined structure, which is specified by a characteristic feature vector  $\mathbf{u} = (u_1 u_2 \dots u_{200})$ . The value of a voxel  $\mathbf{v}$  in the binary scalar field, used as the input for the marching cubes algorithm, is calculated as:

$$f(\mathbf{u}, \mathbf{v}) = \begin{cases} 1, & \text{if } \mathbf{u} \cdot \mathbf{v} \leq t \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

The extracted surface changes with the (binary) value of  $f(\mathbf{u}, \mathbf{v})$ , which is controlled by the characteristic feature vector  $\mathbf{u}$  and the value of threshold  $t$ . Users can interactively configure  $\mathbf{u}$  and  $t$  as well as the color and opacity of the corresponding surface (used for blending multiple semi-transparent surfaces) using the design widget.

The marching cubes algorithm is highly parallelizable because each cube can be processed independently. Our GPU-based parallel implementation uses the histogram pyramid data structure [32]

to allow the query of a triangle's location in the volume, and the query of the triangle indices (if any) inside of a specific cube. In a histogram pyramid, each space-partitioning square cell stores the number of triangles in the corresponding partition. The levels from the bottom to the top of the pyramid represent the volumetric data at increasingly lower spatial resolutions. When constructing the pyramid, a cell stores the sum of eight corresponding cells (i.e.  $2 \times 2 \times 2$ ) in the previous (bottom) level. Both queries require a top-down traversal of the pyramid, which only takes a constant time that is logarithmic to the size of spatial dimension (or linear to the height of pyramid). For visual appeal we smooth the extracted triangular mesh by moving each vertex to the average location of its adjacent vertices in the mesh. During mesh smoothing the query of adjacent vertices is done by first fetching a list of triangles in the current cube and its neighboring cubes (through the second type of queries) and then inspecting each edge of the fetched triangles. We use two vertex buffers to store vertex positions before and after the smoothing. After each smoothing iteration, we swap the two vertex buffers to avoid expensive data movement.

### 3.4 Reordering high-dimensional features

The order of features in conventional volume visualization designs is usually straightforward and less of a concern because of the low dimensionality of feature spaces. Nevertheless, the high-level features extracted from the CNN are 200-dimensional. Further, some, but not all, of the extracted features could be closely related to each other. If two unrelated features (e.g.  $i$ - and  $j$ -th feature) are in close proximity along the  $x$ -axis of the design widget (left of Figure 5), assigning different weights to them in the characteristic feature vector (e.g.  $u_i$  and  $u_j$ ) may be difficult. In the worst case, such overhead could even overshadow the benefits of applying the deep-learning-assisted approach.

We address the relationships among features by reorganizing them using spectral ordering, which sorts the features by the eigenvector of the second smallest eigenvalue of a graph Laplacian [33]. We start with a feature-by-voxel matrix  $M$  and compute a similarity (adjacency) matrix  $A$  by multiplying the matrix by its transpose, resulting in a feature-to-feature matrix:  $A = MM^\top$ .

Next, a diagonal matrix  $D$  is created by calculating the sum of the  $i$ -th row in the similarity matrix  $A$  and placing it at  $D(i, i)$ . The Laplacian matrix  $L$  is then calculated as  $L = D - A$ . Finally, the normalized Laplacian matrix  $\mathcal{L}$ , which is more robust to non-regular graphs [34], is calculated as  $\mathcal{L} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ .

Based on the normalized Laplacian matrix  $\mathcal{L}$ , we calculate the eigenvector associated with the second smallest non-negative eigenvalue (the Fiedler vector). After calculating the Fiedler vector, we sort the original ordering of features in the input matrix  $M$  using the values in the Fiedler vector. The result is an ordering of features where neighboring features are similar. The Fiedler vector and other eigenvectors associated with small eigenvalues also form the basis of spectral clustering [35].

We use the following example to show the effect of organizing deep features into the spectral order (Figure 6). A bright pixel in the  $200 \times 200$  similarity matrix marks a pair of highly-correlated features (Figure 6a). Many pairs of the 200 features are indeed highly correlated because many pixels are bright. Nevertheless, an arbitrary order of features does not take advantage of such correlations, resulting in a disorganized similarity matrix (left of Figure 6a). The spectrally ordered similarity matrix puts similar features closer together, resulting in large bright blocks of various

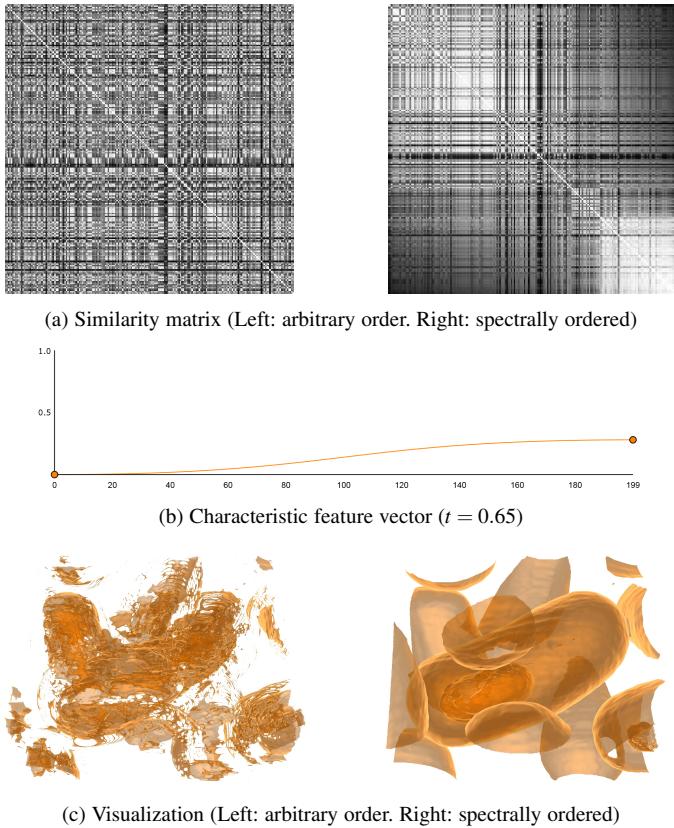


Figure 6. (a) Before spectral ordering, the similarity matrix, in which a bright pixel represents a pair of highly-correlated features, does not show apparent pattern. After spectral ordering, highly-correlated features are closer to each other. (b) A simple characteristic feature vector. (c) The arbitrary order of features does not exploit the correlation among features, thus leading to broken surfaces. The spectrally ordered features allow creating a visualization that reveals interesting structures using the same characteristic feature vector.

sizes along the diagonal (right of Figure 6a). An accessible feature order therefore allows selecting similar voxels using fewer control points in the design widget.

Reordering features into this accessible form is a crucial step. In this example, we use the same characteristic feature vector, which simply specifies an increasing weight from the first (leftmost) to the last (rightmost) feature (Figure 6b), and compare the visualization results obtained before and after reordering features. An arbitrary feature order does not exploit the similarity among features and creates surfaces that are broken into small pieces, which do not correspond well to specific structures (left of Figure 6c). In contrast, the spectrally ordered features are highly structured such that even a simple characteristic feature vector can reveal some interesting structure (right of Figure 6c).

### 3.5 Feature compression

Interactive rendering of volumetric data based on high-dimensional voxel representations can be prohibitive because the inflated data size may exceed the memory size on modern GPUs. Although directly using 20-dimensional texture features has been shown to be effective previously on small datasets (less than 8.4 million voxels) [9], in this paper we use 200-dimensional features and validate our technique on much larger datasets (over one hundred million voxels). Suppose each feature is stored as a four-byte floating point number, loading all 200 features would require

80 gigabytes of memory per hundred million voxels. The high storage demand far exceeds the capability of modern consumer-level graphic cards.

Possible solutions to this problem are out-of-core rendering and data compression. For simplicity, we use vector quantization (VQ) to compress the high-dimensional feature vector  $\mathbf{v}$ . We use an incremental  $k$ -means algorithm [36] to find  $N_c$  voxel clusters (i.e. code words in the codebook). Each voxel is encoded by the index of the nearest cluster centroid according to the  $L^2$  distance. During rendering, voxels are decoded by referencing the codebook.

Reconstruction using VQ, which is a lossy compression technique, inevitably introduces information loss. The codebook size  $N_c$  is closely related to the reconstruction error of VQ. Choosing a  $N_c$  too small will lead to excessive information loss such that the reconstructed feature vectors (i.e. the cluster centroids found by  $k$ -means) do not well approximate the actual feature vectors. On the other hand, choosing a  $N_c$  too large will inflate the size of codebook and require much more computation. A similar information loss introduced by dimension reduction techniques has been shown to be tolerable when simplifying the feature space of volume rendering [12], [16]. In the following we will evaluate how information loss affects the quality of visualizations created using our deep-learning-assisted approach.

In Figure 7 we visually evaluate the quality of the visualization results with various codebook sizes  $N_c$ . In this example we use a subvolume containing a sporulating bacterium of the bacteria dataset (Figure 7a). By comparing the results shown in Figure 7b–7e, the only visually apparent difference is near the center of the spore, where a small region is excluded from the spore, when  $N_c = 64$  (black box in Figure 7b); the other three results are comparable despite the difference in  $N_c$  (Figure 7c–7e). Throughout this work, we choose  $N_c = 256$  such that the code can be stored in eight bits, exactly the same size as the intensity values.

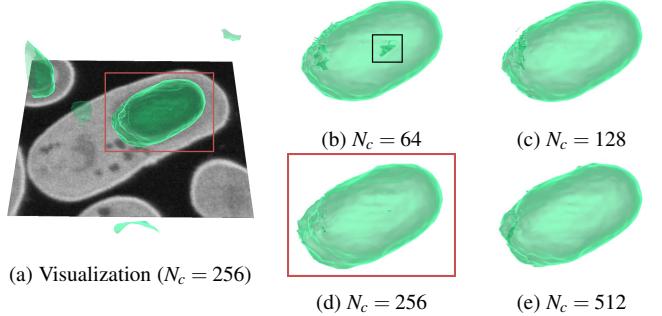


Figure 7. The size of codebook  $N_c$  in VQ controls the amount of information loss. The zoom-in views shown in (b–e) are generated from the same subvolume (red box in (a)) in the bacteria dataset with  $N_c$  ranges from 64 to 512. When  $N_c = 64$ , the view shows a noticeable disparity near the center of the spore (black box in (b)). The results in (c–e) are comparable without significant visual differences.

In some cases where an outlier voxel is replaced by the centroid of the assigned cluster, calculated as the representative for the majority voxels in that cluster, the error leads to an effect similar to a low-pass filter. The errors introduced in such cases are troublesome if those outliers, possibly anomalies or infrequent cases in the data, should be presented (instead of filtered out) in the visualization. In our applications that focus on creating a general view of the data, the low-pass filtering effect does not obstruct the understanding and interpretation of the visualization results.

### 3.6 Semi-automatic exploration

The extraction of surfaces based on a user-defined characteristic feature vector is flexible in identifying various groups of voxels. Nevertheless, even after reordering features, configuring the characteristic feature vector can still be tedious. In the past, we have used recursive segmentation of intensity-gradient 2D histogram to group similar voxels into a hierarchy [15]. Here we develop a similar semi-automatic method to explore volumetric data by exploiting voxel similarities in high-dimensional feature spaces. This method allows users to efficiently visualize a small set of surfaces in a pre-calculated hierarchy.

Given the result of VQ, we perform binary partition recursively to the adjacency graph based on the  $L^2$  distance of the  $N_c$  centroids using spectral clustering [35]. The first binary partition creates two sets of centroids, each corresponds to a group of voxels. Subsequent partitions further subdivide a target set of centroids into two disjoint sets. We organize the partition results into a binary tree, in which a parent node represents the union of the two corresponding groups of voxels represented by the two child nodes. The group of voxels represented by a node defines a surface that can be extracted by the marching cubes algorithm. Based on the binary tree, users can select any nodes to interactively and hierarchically explore the volumetric data. For example, selecting a child node of a parent node shows the subdivision of the surface that corresponds to the parent node.

## 4 EXPERIMENTS

### 4.1 Implementation

We implemented the CNN using the Caffe framework [37]. The GPU-based parallel implementation of the marching cubes-based volume rendering is implemented using OpenGL and OpenCL<sup>1</sup>. The feature reordering and compression are implemented in MATLAB and Python, respectively. The training time reported in the following is based on a computer with an Intel Xeon 2.6 GHz CPU and an NVIDIA GTX 970 GPU. The training of the CNN took about 2.5 hours; the training needs to be done only once for the training dataset. After training the CNN, the extraction of deep features given new unseen data (i.e. forward pass of a trained CNN) took 3.4 minutes per million voxels.

### 4.2 Results

Modern electron microscopy now captures images at high resolution in the scale of few nanometers to a couple of angstroms [38]. High-resolution 3D imaging has become an important tool to study minute changes in morphology and geometry. Focus ion-beam scanning electron microscopy (FIB-SEM) is a high resolution 3D imaging technique that has led to innovations in many biological systems [39]. Besides electron microscopy, magnetic resonance imaging (MRI) is another important modality, which allows precise diagnosis of brain tumor and traumatic brain injury.

Given large high-resolution volumes, computational techniques are essential to create visualizations of complex structures that were previously inaccessible by conventional methods. In the following, we present results with two FIB-SEM images and one MRI dataset of human brain. Besides visualization results, we also evaluate the descriptive power of the CNN-derived features by using the trained CNN to perform volume segmentation through voxel-wise classification.

1. <https://www.khronos.org/opencl/>

### 4.2.1 Segmentation performance measures

The segmentation performance is evaluated by three established performance measures, namely the VOC score [40], the Rand index [41] and the adjusted Rand index [42]. We also compare the effectiveness of two postprocessing techniques using these performance measures.

The VOC score is calculated as  $\text{VOC} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}$ , where TP, FP, and FN denote the number of true positives, false positives, and false negatives, respectively. The VOC score penalizes classifiers with high FP rates, therefore distinguishes reasonable classifiers from classifiers that predict the same class all the time. We calculate class-wise VOC to evaluate the segmentation quality for each class individually.

The Rand index (RI) and adjusted Rand index (ARI) assess segmentation quality based on the portion of agreements among all pairs of  $n$  voxels. The RI is calculated as  $\text{RI} = \frac{a+b}{\binom{n}{2}}$ , where  $a$  and  $b$  denote the number of voxel pairs that belong to the same and different segments in the groundtruth and the generated segmentation. The ARI measures to what extent the segmentation is better than a random segmentation in terms of RI. We calculate both the RI and ARI to evaluate the overall segmentation quality, complementary to the class-wise VOC score.

### 4.2.2 Dataset 1: Bacteria

The first dataset contains one volume of size  $1150 \times 450 \times 400$  at resolution  $12\text{nm} \times 12\text{nm} \times 12\text{nm}$ . The original anisotropic volume resolution is  $3\text{nm} \times 3\text{nm} \times 6\text{nm}$  [43]. With this dataset researchers study the sporulation of *B. subtilis* [44], a common rod-shaped gram positive bacterium.

Sporulation (i.e. formation of an endospore) is a unique survival mechanism triggered in certain bacteria in response to environmental stressors such as nutrient depletion. Once initiated, the process of sporulation includes a series of events that are tightly regulated both genetically and temporally. This temporal process results in a sequence of well-defined and replicable morphological states [44], commencing with the asymmetric division of the mother cell, and culminating in the release of a mature spore. The course of cell development is thus of great interest to biologists, and is often studied in the model system of *B. subtilis*.

This dataset is manually labeled with four classes: Resin, cell wall, spore, and others.

- 1) **Resin:** The dark region in the background.
- 2) **Cell wall:** The bright thin layer separating the interior of a bacterium and the surrounding resin.
- 3) **Spore:** The large dark oval-shaped structure inside of a sporulating bacterium. The intensity value and shape of a spore change over time, depending on the life cycle stage of the bacterium. In this dataset, only about 20% of the bacteria are sporulating.
- 4) **Others:** All the other unlabeled subcellular features (e.g. vesicles and cytoplasm).

Figure 8 shows an example of two bacterial cells with the aforementioned structures of interest. Nevertheless, because the bacteria are oriented differently, the appearance of these structures when projecting onto a 2D plane can be significantly different. We divide the original image stack into two sets of equal size  $575 \times 450 \times 400$ , one for training and validation, and the other for testing.

**Table 2**  
The quality measures of the segmentation results for the bacteria and hippocampus datasets. Numbers separated by slashes correspond to different postprocessing techniques (none / class prior / fitted transformation).

|             | Class            | Precision             | Recall                | VOC                   | Rand index (RI)       | Adjusted RI (ARI)     |
|-------------|------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| Bacteria    | Resin            | 0.996 / 0.995 / 0.995 | 0.992 / 0.993 / 0.993 | 0.988 / 0.989 / 0.989 |                       |                       |
|             | Cell wall        | 0.931 / 0.963 / 0.943 | 0.956 / 0.918 / 0.946 | 0.893 / 0.887 / 0.894 |                       |                       |
|             | Spore            | 0.813 / 0.947 / 0.860 | 0.881 / 0.718 / 0.842 | 0.733 / 0.690 / 0.741 |                       |                       |
|             | others           | 0.964 / 0.941 / 0.959 | 0.953 / 0.979 / 0.963 | 0.921 / 0.922 / 0.925 |                       |                       |
| average/all |                  | 0.970 / 0.970 / 0.971 | 0.969 / 0.969 / 0.971 | 0.943 / 0.941 / 0.945 | 0.969 / 0.969 / 0.971 | 0.934 / 0.934 / 0.937 |
| Hippo.      | Non-mitochondria | 0.999 / 0.994 / 0.991 | 0.959 / 0.989 / 0.993 | 0.958 / 0.983 / 0.985 |                       |                       |
|             | Mitochondria     | 0.571 / 0.820 / 0.873 | 0.984 / 0.900 / 0.847 | 0.566 / 0.752 / 0.754 |                       |                       |
|             | average/all      | 0.976 / 0.985 / 0.985 | 0.960 / 0.984 / 0.985 | 0.937 / 0.971 / 0.972 | 0.923 / 0.969 / 0.971 | 0.670 / 0.835 / 0.838 |

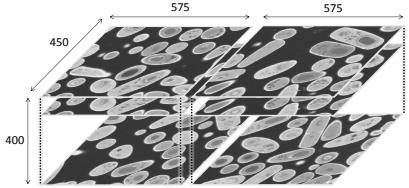


Figure 8. (left) The bacteria dataset is divided into the left and right halves for training and testing. (right) Both the spores (green) and the vesicles (yellow arrow) in the two sporulating bacteria are low in intensity. Because the difference in intensity between a spore and other structures (e.g. vesicle and cytoplasm) can be small, conventional intensity-based feature spaces will not differentiate them well.

In the following we evaluate the quality of the segmentation results and compare the effectiveness of the two postprocessing techniques that address the class imbalance problem (c.f. Section 3.2.5). In the training volume of the bacteria dataset, the proportion of (resin, cell wall, spore, others) is (0.472, 0.119, 0.034, 0.374), respectively. Because of the scarcity of the spores (0.034), we expect an overestimation of the abundance of spores in the testing stage. Interestingly the result shows that simple postprocessing using class prior can actually deteriorate segmentation performance. Table 2 compares the performance measures obtained with and without postprocessing. For the bacteria dataset, using class prior as the scaling factor of predicted probability overestimated the effect of class imbalance and scaled down the predicted probability of minority classes (i.e. spore) too much, resulted in a significant drop in recall for the spore class (from 0.881 to 0.718).

Figure 9 shows a concrete example comparing the segmentation results (Figure 9c–9e) of the input image (Figure 9a) with the groundtruth (Figure 9b). Applying postprocessing using class prior resulted in a significant increase in false negatives (Figure 9d) than before postprocessing (Figure 9c), therefore lowered the VOC score of the spore class from 0.793 to 0.690. In contrast, the fitted transformation postprocessing is more robust in handling various classes because it derives suitable scaling factors in a data-driven way (Figure 9e). Given this flexibility, the fitted transformation increased the VOC scores of all four classes as well as the RI and ARI even though the scores are only marginally better than those without postprocessing.

Having validated the segmentation results, we next present our results on volume visualization using the extracted deep features. We have shown in Section 3.1.1 that conventional feature spaces, though useful in many cases, are ineffective for complex structures. We will now show our deep-learning-assisted approach

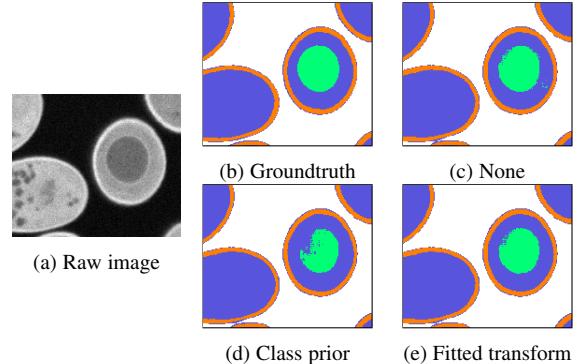


Figure 9. This example shows the spore (green) in (b) the groundtruth labels and the segmentation results (c) generated without postprocessing, and postprocessed with (d) class prior and (e) fitted transform for the raw image in (a). Simply scaling the predicted probability by class prior resulted in many false negatives for the spore class. In contrast, the data-driven fitted transformation of probability is more robust and generates a result comparable to the groundtruth.

is effective for complex structures.

For example, the two characteristic feature vectors shown in Figure 10e correspond the surfaces extracted for the spores (Figure 10a and Figure 10b) and the cell walls (Figure 10c), respectively. Combining the two extracted surfaces leads to a composite visualization showing both structures (Figure 10d). The close relationships between the semantics and the derived deep features are attributed to the semantic information, in the form of the voxel classes, users provide to the CNN in the training stage. Grouping semantically-related features together in the design widget with spectral ordering facilitates the editing of characteristic feature vectors to target voxels with specific semantics. Further studies are needed to examine the constituents of these semantically-coherent features and whether they can be reused for building visualizations for different target structures.

#### 4.2.3 Dataset 2: Hippocampus

The second dataset<sup>2</sup> [29], [45] contains two volumes of hippocampus, each of size  $1024 \times 768 \times 165$  and resolution  $5\text{nm} \times 5\text{nm} \times 5\text{nm}$ . Abnormal changes in morphology and spatial distribution of mitochondria are known to be related to cancer and neurodegenerative diseases such as Parkinson’s disease [46]. Better visualization and segmentation of mitochondria are therefore crucial to the study of these diseases. Both volumes in this dataset

2. <http://cvlab.epfl.ch/data/em>

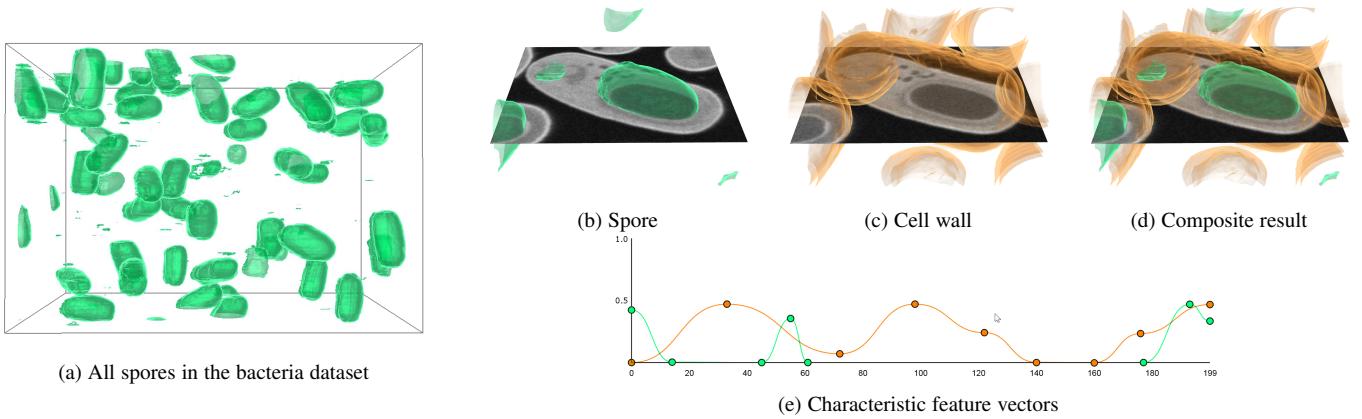


Figure 10. (a) The visualization of the spores in the bacteria dataset generated using the deep-learning-assisted approach. We show in the next two figures (b) the spore and (c) the cell wall of a bacterial cell. By rendering both semi-transparent surfaces at a time, (d) the composite result shows both the spore and the cell wall. (e) The characteristic feature vectors used to generate the green ( $t = 0.36$ ) and orange ( $t = 0.46$ ) surfaces.

contain manually-labeled mitochondria (Figure 11). All the other structures are considered as non-mitochondria.

Previous studies have shown that contextual features improve segmentation quality over standard features including intensity histogram, gradient magnitude, and texture-related features calculated locally [29]. Similar to the high-level features derived by CNNs, the contextual features describe relationships among voxels in a large neighborhood, thereby allowing them to depict more precisely the structures of mitochondria. Nevertheless, the contextual features are handcrafted instead of automatically learned.

We use the first volume for training and validation, and the second volume for testing. The first volume is divided into the training and the validation sets of size  $768 \times 768 \times 165$  and  $256 \times 768 \times 165$ , respectively.

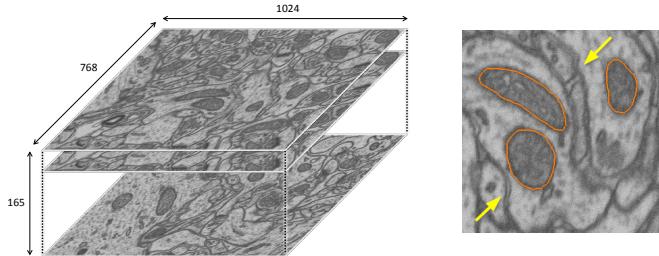


Figure 11. (left) The hippocampus dataset consists of two volumes for training and testing (not shown here), both of the same size. (right) Both the mitochondria (orange) and membranes (yellow arrow) are low in intensity, thus making them inseparable using intensity-based features.

The second part of Table 2 shows that the postprocessings are much more influential in addressing the class imbalance problem in the hippocampus dataset, where the mitochondria occupy about four percent of the voxels in the training volume. After applying the two postprocessing techniques, the precision for mitochondria increased significantly whereas the recall decreased only moderately. As a result, the VOC scores of mitochondria increased from 0.566 to 0.752 (class prior) and 0.754 (fitted transformation). Both postprocessing techniques generated VOC scores that are higher than 0.741, which is the state-of-the-art previously reported in [29]. Overall, the RI and ARI both improved substantially after postprocessing.

Compared with the bacteria dataset, the postprocessing has a larger impact on performance in the hippocampus dataset than that in the bacteria dataset. Such a difference occurred because more

voxels have similar predicted probabilities for both classes in the hippocampus dataset. After adjusting the predicted probabilities, more voxels in the hippocampus dataset change their predicted class from mitochondria (minority class) to non-mitochondria (majority class).

We have shown that the visualization results of the bacteria dataset (c.f. Figure 10) indicate that the derived deep features are semantically meaningful. In that example the features align closely with the classes assigned by users. For a voxel class with large intra-class variations, the derived deep features may each detect only a subset of the variants in the same class. For solving a classification problem, the CNN aggregates these features in the last fully-connected layer to detect that specific class as a whole. For creating interesting visualizations, we can use these features targeting different sets of voxels to show subclasses within a user-assigned class.

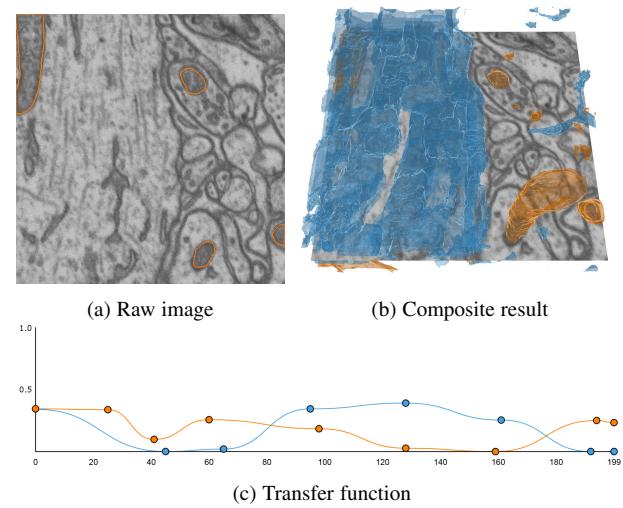


Figure 12. (a) Within the non-mitochondria region (outside of the orange regions), subregions with noticeably different characteristics can be identified. (b) The extracted deep features can be used to show the non-mitochondria regions without membranes (blue subvolumes in the left). (c) The two characteristic feature vectors correspond to the mitochondria (orange,  $t = 63$ ), and the non-mitochondria regions without membranes (blue,  $t = 49$ ) in the composite visualization.

For example in Figure 12b, the mitochondria (orange) and the non-mitochondria regions in the left (blue) correspond to the

orange and blue characteristic feature vectors (Figure 12c). Although the same (non-mitochondria) class is assigned to both non-mitochondria regions in the left and right during training, the CNN derived different features to detect them because of their distinct appearance. In the left the single largest non-mitochondria region is bright and uniform, whereas the non-mitochondria regions in the right consist of small bright regions separated by dark membranes. As a result the blue characteristic feature vector generates only the surface in the left.

#### 4.2.4 Dataset 3: Multimodal Brain Tumor Image Segmentation Benchmark (BRATS)

The Multimodal Brain Tumor Image Segmentation Benchmark (BRATS) [47] contains MRI images taken with glioma patients. Here we report results generated with the dataset released with the MICCAI 2013 data challenge<sup>3</sup>. In this dataset, each sample volume consists of four channels, namely the T1, T1c, T2, and Flair channels. All samples in the training set are manually labeled. In clinical practice, physicians decide the tissue types based on the results of multiple channels to diagnose more precisely. The structures of interest in this dataset are the edema, necrosis, non-enhancing and enhancing tumor. The boundaries between abnormal tissues and normal tissues (e.g. the gray matter, the white matter and the cerebrospinal fluid) are usually ambiguous. Figure 13 shows an example of the four channels and the four types of abnormal tissues. Typical volumes are cube-shaped, composed of about six to eight million voxels, resampled and registered into 1mm isotropic resolution.

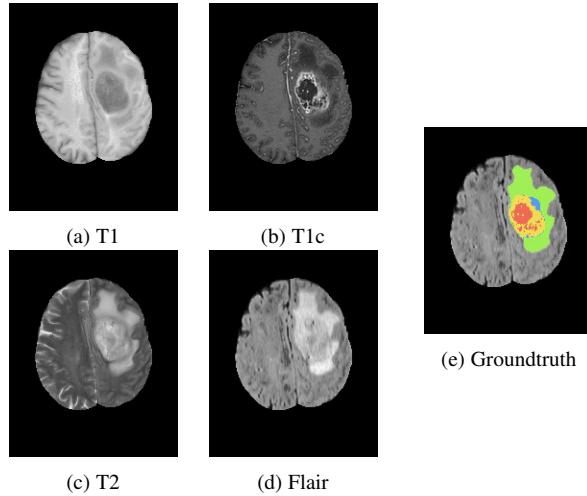


Figure 13. A slice in the volume for the (a) T1, (b) T1c, (c) T2, and (d) Flair channels. (e) The groundtruth labels of the necrosis (red), edema (green), non-enhancing tumor (blue), and enhancing tumor (yellow).

Here we report the visualization results based on the deep features extracted from a recently published CNN-based method [5], which is designed specifically for the BRATS dataset. Instead of training from scratch, we use the trained CNN model provided by the authors of the original study as a feature extractor that generates 256 deep features<sup>4</sup> for each voxel. We focus on showcasing the application of our visualization design given an alternative high-dimensional feature space, which in this case is also derived by a trained CNN model.

3. <http://braintumorsegmentation.org/>

4. Output of the second fully-connected layer (layer 10 in the original paper).

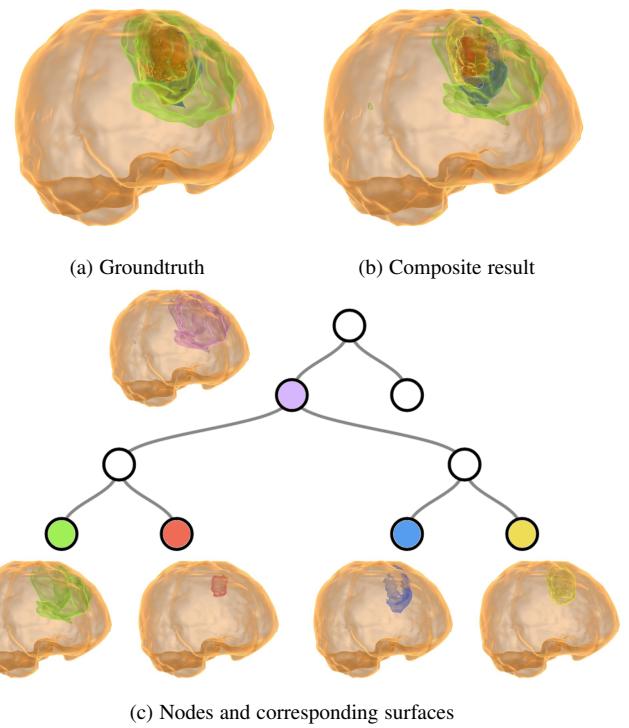


Figure 14. (a) The visualization of groundtruth labels. (b) The composite result generated using the semi-automatic method. (c) The binary tree allows hierarchical exploration of the volume. The four child subvolumes (represented by the leaf nodes) are created by partitioning the parent subvolume (represented by the purple node).

The visualization presented here is created for a sample with diagnosis of high-grade gliomas (i.e. HG\_0011) in the BRATS dataset (Figure 14). Following the hierarchical exploration procedure from the top to the bottom of the tree, the final four surfaces correspond to four leaf nodes in the binary partition tree (Figure 14c). Comparing the composite result (Figure 14b) created with the four surfaces obtained using the semi-automatic method with the four surfaces obtained using the semi-automatic method (Figure 14a), we can see some inconsistencies near the center of the tumorous region. For example, the yellow region (enhancing tumor) that wraps around the red region (necrosis) is more apparent in our composite result. These inconsistencies are due to the difficulty in precisely identifying the ambiguous boundaries between tissues of different types. In fact, the BRATS dataset is so challenging that the evaluation of segmentation performance is done after merging multiple voxel classes (e.g. complete tumor for all four classes and core tumor for the three classes excluding edema [47]). We can also flexibly choose to visualize the complete tumor by selecting the purple node (top left corner of Figure 14c), which represents the union of the four child subvolumes represented by the leaf nodes.

As more and more research applied CNN-based method to this dataset recently (from three in 2014 to nine in 2016), we expect the derived high-level features to be more powerful in the future. Our deep-learning-assisted approach will continue to benefit from the advances in the deep learning methods. Another interesting remark in applying our deep-learning-assisted approach to multimodal data is that the interactions among the four different channels are addressed by the first convolution layer. Therefore, the final deep features we extracted are derived both from a larger context and across various channels. Combining information across channels

is crucial to effectively visualize multimodal data. In particular for MRI, using the joint histogram of multiple channels as the feature space has been shown to be effective in showing normal tissues such as cerebrospinal fluid and gray matter [48]. For the tumorous tissues targeted here, the deep features may provide the required context information and descriptive power that lead to better visualizations.

### 4.3 Rendering speed and computation time

Our deep-learning-assisted volume visualization introduces extra steps to decode each voxel and create the binary scalar field if compared with the conventional marching cubes algorithm. In our implementation referencing a codebook, which maps the code of a voxel to its corresponding feature vector  $\mathbf{v}$ , is a simple texture look-up. Updating the binary scalar field requires re-evaluating Equation 1, which can be calculated in parallel efficiently by vector arithmetic in the GPU.

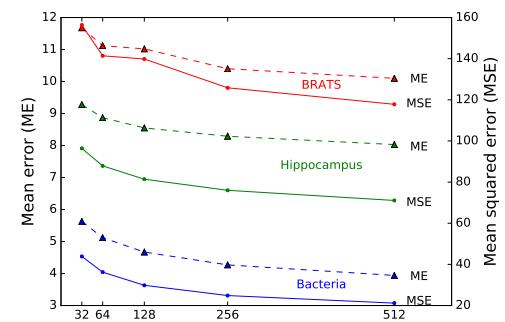
We measure the rendering speed with an NVIDIA Quadro K6000 GPU<sup>5</sup>. For a volume with 43 million voxels, our GPU-based volume renderer can render surfaces of similar size (470k triangles) about 45 and 40 frames per second using the conventional marching cubes algorithm and our deep-learning-assisted approach, respectively. Five iterations of smoothing of the mesh with 470k triangles took about 0.26 second, which is too slow if we perform smoothing every time the mesh changes while users interact with the design widget. Consequently we perform mesh smoothing optionally.

Figure 15 shows the mean and mean squared error (Figure 15a) and the time to perform  $k$ -means clustering and VQ with  $N_c$  ranging from 32 to 512 (Figure 15b). Here we measure error as the  $L^2$  distance between the actual and reconstructed vectors. As the value of  $N_c$  increases, both errors decrease exponentially, whereas the running time increases linearly. The reconstruction error for the BRATS dataset is larger than the other two datasets possibly because the deep features are generated from multimodal data, which contain much more information.

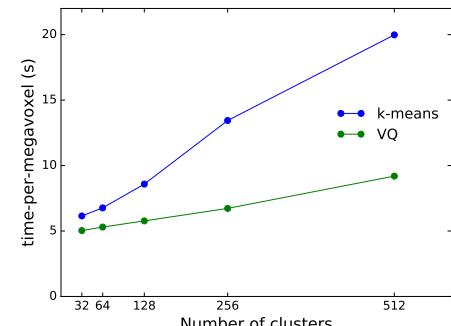
### 4.4 Discussion

In the previous sections, we have shown that the proposed deep-learning-assisted volume visualization is powerful in discerning complex structures using high-level features extracted from a trained CNN. Ideally we would want to know the corresponding high-level concept each individual feature represents. Understanding these derived features is an ongoing pursuit in the deep learning community [6], [49]. As of now the limited interpretability of deep features indeed causes difficulties in establishing the connections between a deep-feature-based characteristic feature vector and the corresponding visualization result. Interestingly, by organizing features using spectral ordering, users can still design useful volume visualizations without directly interpreting each feature dimension that defines the feature space. An alternative strategy, used in our semi-automatic method, is to circumvent the interpretability problem by providing another layer of abstraction and hide the underlying features from users. This strategy is also used by many other semi-automatic methods.

Although the choice of features evolves from intensity to locally-derived features (e.g. texture) and, as introduced in this paper, deep features, volume visualization design remains a



(a) Reconstruction error



(b) Running Time

Figure 15. (a) The reconstruction error of vector quantization decreases as the size of codebook (i.e. number of clusters) increases. (b) Larger codebook requires much more time running  $k$ -means and VQ.

manual and trial-and-error process. A distinct line of research focuses on minimizing human involvement by creating volume visualizations automatically [50]. These automatic techniques can also be adapted to search for solutions in the deep-feature-based feature space we introduced.

Our current supervised-learning-based approach requires labeled data for training the CNN; this limitation could be mitigated by applying unsupervised learning. For example, instead of training a CNN for classification, train an autoencoder [51] that learns to reconstruct voxels from intermediate codes, which also represent high-level features derived directly from the data. The other techniques used in our deep-learning-assisted approach still apply to those derived abstract features despite of the way they are generated. Nevertheless, unsupervised techniques, while not relying on user inputs, at the same time eliminate valuable supplementary domain knowledge from the learning process. Without incorporating any prior knowledge on the data, the derived features will not always be suitable for the target structures. Further studies are needed to understand whether and how unsupervised learning can help visualize complex volumetric data.

## 5 CONCLUSIONS AND FUTURE WORK

Designing volume visualizations is challenging because the current workflow requires users to explicitly define a feasible feature space for the target structures. Existing studies have focused on visualizing structures based on specific handcrafted local features. As the complexity of structures increases, the difficulty of defining a suitable feature space also increases significantly.

In this work, instead of relying on handcrafted features, we use convolutional neural networks (CNNs) to derive automatically

5. This is different from the GPU used for training the CNN.

useful features from the data. In contrast to the local features that have been used in the past, the features extracted from the CNNs depict high-level concepts that are difficult to describe by local features. We organize the extracted high-dimensional abstract features by spectral ordering such that similar abstract features are close together in the design widget, thereby facilitating interactive exploratory visualization. In addition, we present a semi-automatic technique that creates a binary tree of volume visualizations, which allow users to hierarchically explore volumetric data by traversing the tree.

In the future we plan to apply fully-convolutional network with 3D convolutions to further address the spatial relationships among voxels and accelerate the time-consuming voxel-wise prediction. Another possible extension is combining features from different layers in the CNN to design visualizations at various granularities, possibly showing substructures within a structure.

## ACKNOWLEDGMENTS

## REFERENCES

- [1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
- [2] Q. Le, “Building high-level features using large scale unsupervised learning,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 8595–8598.
- [3] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in *Proceedings of the Annual Conference on Advances in Neural Information Processing Systems*, 2014, pp. 1988–1996.
- [4] D. Ciresan, U. Meier, and J. Schmidhuber, “Multi-column deep neural networks for image classification,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3642–3649.
- [5] S. Pereira, A. Pinto, V. Alves, and C. A. Silva, “Brain tumor segmentation using convolutional neural networks in MRI images,” *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1240–1251, May 2016.
- [6] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, “Striving for simplicity: The all convolutional net,” in *Proceedings of International Conference on Learning Representations Workshop*, 2015.
- [7] 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.
- [8] 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.
- [9] 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, Nov. 2008.
- [10] C. 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, Feb. 2011.
- [11] 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.
- [12] X. Zhao and A. E. Kaufman, “Multi-dimensional reduction and transfer function design using parallel coordinates,” in *Proceedings of IEEE International Conference on Volume Graphics*, 2010, pp. 69–76.
- [13] R. Maciejewski, Y. Jang, I. Woo, H. Jänicke, K. Gaither, and D. Ebert, “Abstracting attribute space for transfer function exploration and design,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 1, pp. 94–107, Jan. 2013.
- [14] R. Maciejewski, I. Woo, W. Chen, and D. S. 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.
- [15] C. Y. Ip, A. Varshney, and J. JaJa, “Hierarchical exploration of volumes using multilevel segmentation of the intensity-gradient histograms,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 12, pp. 2355–2363, 2012.
- [16] H. S. Kim, J. P. Schulze, A. C. Cone, G. E. Sosinsky, and M. E. Martone, “Dimensionality reduction on multi-dimensional transfer functions for multi-channel volume data sets,” *Information Visualization*, vol. 9, no. 3, pp. 167–180, 2010.
- [17] W. E. Lorensen and H. E. Cline, “Marching Cubes: A high resolution 3D surface construction algorithm,” in *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’87, New York, NY, USA: ACM, 1987, pp. 163–169.
- [18] 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 IEEE Conference on Computer Vision and Pattern Recognition*, Sep. 2014, pp. 1–9.
- [19] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proceedings of International Conference on Learning Representations*, 2015.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
- [21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [22] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of International Conference on Machine Learning*, 2015, pp. 448–456.
- [23] J. Kruger and R. Westermann, “Acceleration techniques for GPU-based volume rendering,” in *Proceedings of IEEE Conference on Visualization*, Washington, DC, USA, 2003, pp. 287–292.
- [24] I. Fujishiro, Y. Maeda, H. Sato, and Y. Takeshima, “Volumetric data exploration using interval volume,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 2, no. 2, pp. 144–155, 1996.
- [25] P. Bhaniramka, C. Zhang, D. Xue, R. Crawfis, and R. Wenger, “Volume interval segmentation and rendering,” in *Proceedings of IEEE Symposium on Volume Visualization and Graphics*, Oct. 2004, pp. 55–62.
- [26] G. Kindlmann, R. Whitaker, T. Tasdizen, and T. Moller, “Curvature-based transfer functions for direct volume rendering: Methods and applications,” in *Proceedings of IEEE Conference on Visualization*, 2003, pp. 513–520.
- [27] T. Gerstner, “Multiresolution extraction and rendering of transparent isosurfaces,” *Computers & Graphics*, vol. 26, no. 2, pp. 219–228, Apr. 2002.
- [28] R. M. Haralick, “Statistical and structural approaches to texture,” *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, 1979.
- [29] A. Lucchi, C. Becker, P. M. Neila, and P. Fua, “Exploiting enclosing membranes and contextual cues for mitochondria segmentation,” in *Proceedings of International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2014, pp. 65–72.
- [30] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.
- [31] M. D. Zeiler, “ADADELTA: An adaptive learning rate method,” *arXiv:1212.5701 [cs]*, Dec. 2012.
- [32] C. Dyken, G. Ziegler, C. Theobalt, and H.-P. Seidel, “High-speed marching cubes using HistoPyramids,” *Computer Graphics Forum*, vol. 27, no. 8, pp. 2028–2039, Dec. 2008.
- [33] M. Fiedler, “Algebraic connectivity of graphs,” *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.
- [34] F. R. Chung, *Spectral Graph Theory*. American Mathematical Soc., 1997, vol. 92.
- [35] J. Shi and J. Malik, “Normalized cuts and image segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.
- [36] D. Sculley, “Web-scale k-means clustering,” in *Proceedings of the International Conference on World Wide Web*. ACM, 2010, pp. 1177–1178.
- [37] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the ACM International Conference on Multimedia*, 2014, pp. 675–678.
- [38] A. Bartesaghi, A. Merk, S. Banerjee, D. Matthies, X. Wu, J. L. S. Milne, and S. Subramaniam, “2.2 Å resolution cryo-EM structure of  $\beta$ -galactosidase in complex with a cell-permeant inhibitor,” *Science*, vol. 348, no. 6239, pp. 1147–1151, Jun. 2015.
- [39] K. Narayan and S. Subramaniam, “Focused ion beams in biology,” *Nature Methods*, vol. 12, no. 11, pp. 1021–1031, Oct. 2015.

- [40] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.
- [41] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971.
- [42] L. Hubert and P. Arabie, "Comparing partitions," *Journal of Classification*, vol. 2, no. 1, pp. 193–218, Dec. 1985.
- [43] K. Narayan, C. M. Danielson, K. Lagarcic, B. C. Lowekamp, P. Coffman, A. Laquerre, M. W. Phaneuf, T. J. Hope, and S. Subramaniam, "Multi-resolution correlative focused ion beam scanning electron microscopy: Applications to cell biology," *Journal of Structural Biology*, vol. 185, no. 3, pp. 278–284, Mar. 2014.
- [44] I. S. Tan and K. S. Ramamurthi, "Spore formation in *Bacillus subtilis*," *Environmental Microbiology Reports*, vol. 6, no. 3, pp. 212–225, Jun. 2014.
- [45] A. Lucchi, Y. Li, and P. Fua, "Learning for structured prediction using approximate subgradient descent with working sets," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2013, pp. 1987–1994.
- [46] A. C. Poole, R. E. Thomas, L. A. Andrews, H. M. McBride, A. J. Whitworth, and L. J. Pallanck, "The PINK1/Parkin pathway regulates mitochondrial morphology," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 105, no. 5, pp. 1638–1643, Feb. 2008.
- [47] B. H. Menze, A. Jakab *et al.*, "The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993–2024, Oct. 2015.
- [48] J. Kniss, J. P. Schulze, U. Wössner, P. Winkler, U. Lang, and C. Hansen, "Medical applications of multi-field volume rendering and VR techniques," in *Proceedings of the Joint Eurographics - IEEE Conference on Visualization*, ser. VISSYM'04. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2004, pp. 249–254.
- [49] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *Proceedings of International Conference on Machine Learning*, 2014, pp. 647–655.
- [50] M. Ruiz, A. Bardera, I. Boada, I. Viola, M. Feixas, and M. Sbert, "Automatic transfer functions based on informational divergence," *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 12, pp. 1932–1941, 2011.
- [51] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.

PLACE  
PHOTO  
HERE

**Somay Jain** Somay Jain received a B.Tech (Honors) in Computer Science from International Institute of Information Technology, Hyderabad in 2015. He is currently pursuing M.S. in Computer Science from University of Maryland, College Park. His research interests include computer graphics, computer vision and using them in biomedical applications.

PLACE  
PHOTO  
HERE

**Eric Krokos** Biography text here.

PLACE  
PHOTO  
HERE

**Kedar Narayan** Biography text here.

PLACE  
PHOTO  
HERE

**Sriram Subramaniam** Biography text here.

PLACE  
PHOTO  
HERE

**Hsueh-Chien Cheng** received the B.S. degree in Computer Science from National Tsing Hua University and M.S. degree in Computer Science and Information Engineering from National Taiwan University. He is currently a Ph.D. candidate in the Department of Computer Science at the University of Maryland. His research interests include computer graphics and data visualization with biomedical applications.

PLACE  
PHOTO  
HERE

**Antonio Cardone** earned a Ph.D. in Mechanical Engineering at University of Maryland, College Park in 2005. Shortly after he joined the National Institute of Standards and Technology (NIST) as Research Associate. Since 2010 Dr. Cardone is also a Research Scientist for the Institute for Advanced Computer Studies at the University of Maryland. His research is funded by a NIST Research Grant, on which he is Principal Investigator. Dr. Cardones research focuses on computational geometry, image processing and CAD/CAM with application in bioinformatics, material science and engineering design. His work is carried out in collaboration with scientists at the National Institutes of Health, where Dr. Cardone is a Special Volunteer since 2006.

PLACE  
PHOTO  
HERE

**Amitabh Varshney** is the Director of the Institute for Advanced Computer Studies (UMIACS) and Professor of Computer Science at the University of Maryland at College Park. He has a Ph.D. in Computer Science from UNC Chapel Hill and a B.Tech. in Computer Science from IIT Delhi. Dr. Varshneys research is on exploring large data visualization and interaction including spatial and temporal summarization, multiresolution hierarchies, gesture recognition for virtual environments, and visual interaction with wall-sized tiled displays. He is the Director of the NVIDIA CUDA Center of Excellence at Maryland. Varshney received a NSF CAREER Award and the IEEE Visualization Technical Achievement Award in 2004. He served as the Chair of the IEEE Visualization and Graphics Technical Committee 2008–2011. He has served as the Associate Editor for IEEE TVCG 1999–2003, and is currently the Associate Editor-in-Chief of IEEE TVCG.