A new approach to interactive viewpoint selection for volume data sets

Han Suk Kim, Didem Unat, Scott B Baden and Jürgen P Schulze

Information Visualization 2013 12: 240 originally published online 25 February 2013
DOI: 10.1177/1473871612467631

Info Vis

Article

Information Visualization
12(3-4) 240–256

DOI: 10.1177/1473871612467631
ivi.sagepub.com
SAGE

A new approach to interactive viewpoint selection for volume data sets

Han Suk Kim[1], Didem Unat[2], Scott B Baden[1] and Jürgen P Schulze[1]

Abstract

Automatic viewpoint selection algorithms try to optimize the view of a data set to best show its features. They are often based on information theoretic frameworks. Although many algorithms have shown useful results, they often take several seconds to produce a result because they render the scene from a variety of viewpoints and analyze the result. In this article, we propose a new algorithm for volume data sets that dramatically reduces the running time. Our entire algorithm takes less than a second, which allows it to be integrated into real-time volume-rendering applications. The interactive performance is achieved by solving a maximization problem with a small sample of the data set, instead of rendering it from a variety of directions. We compare performance results of our algorithm to state-of-the-art approaches and show that our algorithm achieves comparable results for the resulting viewpoints. Furthermore, we apply our algorithm to multichannel volume data sets.

Keywords

Viewpoint selection, Harris interest point detection, principal component analysis

Introduction

Optimal viewpoint selection is a method that finds a two-dimensional (2D) projection of a three-dimensional (3D) data set that best depicts its main features. A good viewpoint selection algorithm increases the amount of information a viewer perceives of a data set and can save time when exploring the features of a new data set. Although the definition of "optimal viewpoint" is somewhat subjective, previous approaches[1–6] agree that a viewpoint is better if the projected image contains more information. In prior work, how much information an image conveys is determined by entropy functions. The best viewpoint is chosen as the one that has the maximum entropy.

While previous approaches have shown promising results for various data sets, the major limitation of information theoretic frameworks is the fact that they rely on exhaustive search over a large set of samples. This method has to find a balance between accuracy and computation time. Trying out more viewpoints yields better results. This is computationally expensive, especially because volume rendering is rather slow. Reducing the number of samples will improve performance, but the optimal viewpoint might be missed.

For example, suppose we take 400 uniform samples on the viewing sphere. This is, on average, 20 samples around one circumference. Then, the granularity of the sample is $\pi/10$. If the real solution is in a small peak not captured by the granularity, increasing the number of samples is the only way to correctly locate the solution. The execution time then increases in proportion to the sample size because the framework requires rendering for each sample. Even if we exclude

[1]University of California San Diego, La Jolla, CA, USA
[2]Lawrence Berkeley Laboratory, Berkeley, CA, USA

Corresponding author:
Han Suk Kim, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093, USA.
Email: hskim@cs.ucsd.edu

feature computation time and the time for reading back the rendered image from graphics processing unit (GPU), rendering itself takes at least 8 s for 400 samples, assuming that the rendering speed is 50 frames per second. The performance aspect of algorithms has rarely been studied in previous viewpoint selection frameworks, except by Vázquez and Sbert[7] and Lee et al.[8]

In this article, we propose a new, very fast interactive viewpoint selection algorithm. To achieve this goal, we first define what the "best" viewpoint is. When volume data sets are viewed, areas with greater variation in the data are often the users' focus because users try to create a greater variation with transfer function on the area of interest to see the data variance around the area highlighted. Commonly, the transfer function provides the primary way to visually pull out these features. We call these features "visually interesting" features. The best viewpoint is found when the 2D projection of the 3D data set shows all the visually interesting features of a volume data set with the smallest possible amount of occlusion. With this definition, we aim to answer three questions: (1) How to identify visually interesting features? (2) How to find the viewpoint with the least amount of occlusion among these features? (3) How do we efficiently compute this viewpoint?

We answer the first question with an extension of the Harris interest point detection algorithm[9] to find visually interesting features in a 3D volume data set. The algorithm finds corners of an object and areas with high contrast. To answer the second question, we formulate the problem as an optimization problem, and the objective of the optimization is to maximize the variance of projected feature points. This approach tries to lay out the visually interesting features without overlaps. By finding a plane in which the variance of feature points is maximized, we can minimize the overlap between feature points. This optimization is formulated as a principal component analysis (PCA) problem.

Our algorithm runs at interactive speed (<1 s) mainly because of two key features: the simplicity of it and an efficient implementation on the graphics card (GPU). These two strengths address the third problem regarding the efficiency of our approach. Our algorithm computes one feature value for each voxel point (section "Extension to 3D"), and the optimization problem takes only a small set of points as its input, which avoids actually rendering the volume to analyze the result (section "PCA"). We further reduce the computational cost through hardware acceleration by implementing the core parts of our algorithm in Compute Unified Device Architecture (CUDA), to run in parallel on the GPU.

Our interactive viewpoint selection algorithm can help users understand volume data sets better and faster while they explore different transfer functions. Today, direct volume-rendering algorithms are fast enough for interactive frame rates, and the transfer function is crucial for the appearance of the volume.[10] We envision that our viewpoint selection algorithm could be (optionally) automatically triggered after every change of the transfer function, so that users can immediately see the effectiveness of changes of the transfer function during the explorative phase when they do not yet know what a data set contains.

This type of operation, which combines transfer function exploration and automatic viewpoint selection, becomes crucial in investigating multichannel data. Multichannel data have multiple values per voxel, usually three or four. Scientists start their investigation with all channels on. After figuring out the overall appearance, a deeper investigation requires only a small number of channels: individual channel to see each channel closely or two channels to find correlation between the two channels. Because viewers of multichannel data frequently turn on/off channels and try many different combinations, the effective viewpoints for many different views change accordingly. As our viewpoint selection algorithm finds the best view interactively, it helps viewers understand their data better and faster. A previous study[11] proposed this algorithm and compared the results with the state-of-the-art algorithms, and we continue extending the idea with a new result from multichannel data, as well as additional single-channel data sets.

The remainder of this article is organized as follows: section "Related work" reviews previous work on viewpoint selection algorithms. In section "Feature selection," we introduce our feature selection algorithm. Section "View selection" describes in detail how we compute the best view direction from the features. Section "Implementation" discusses issues in implementing and optimizing our algorithm in order to run in real time. Section "Evaluation" presents performance results for several typical data sets and compares the results with two state-of-the-art approaches in terms of performance and visual quality. We also discuss possible applications and limitations of our real-time viewpoint selection algorithm in section "Discussion." Finally, we conclude this article and suggest future work in section "Conclusion and future work."

Related work

The definition of a good view is subjective depending on the purpose of rendering, and it has a long history across many different disciplines.[12–16] The aspect graph[14,15] considers a *general view* as a node of a graph and a *visual event* as an edge of the graph. The general views are a region of views where small changes in the

view direction incur large changes in the geometry of the rendered image. The concept of a viewing sphere here is used in almost all view selection algorithms, and there have been many studies on this topic.[13] The canonical view[12,16] is another concept for defining good views. Palmer et al.[16] did experiments to rate the quality of viewpoints, and Blanz et al.[12] further studied this with psychophysical experiments. There have been many attempts to solve the viewpoint selection problem with information theoretical frameworks and through sampling from a viewing sphere.[1,7,17] These approaches consider a set of viewpoints in a viewing sphere. Then, for each candidate view, they evaluate how good the view is based on the measure they define. These approaches differ in how they capture all the important information in the scene and incorporate the amount of information into their definition of measure. Vázquez et al.[1] propose an entropy-based framework for 3D mesh data. The probability distribution function (PDF) for the entropy is defined with the relative area of the projected faces. This entropy definition prefers a view in which all faces are rendered with the same relative projected area. Vázquez and Sbert[7] extend the work of Vázquez et al.[1] by accelerating the exploration path in entropy calculation. Instead of a brute force search of samples in the viewing sphere, Vázquez and Sbert use an adaptive method to narrow down the search space. This approach has a common goal with our algorithm, achieving an interactive rendering rate, but the algorithm still depends on the size of the scene which the entropy values are computed for and the number of triangles in the 3D mesh. In volume rendering, each frame takes much longer to render than comparable 3D mesh data, but the objects in volume rendering have more detail than the ones used in this article. Polonsky et al.[17] also discuss the best viewpoint selection in the context of 3D mesh rendering. They define multiple view descriptors, such as surface area entropy, visibility ratio, curvature entropy, silhouette length, silhouette entropy, topological complexity, and surface entropy of semantic parts. Finally, Lee et al.[8] define *mesh saliency* to find visually interesting regions.

The information theoretic framework has more recently been applied to volume-rendered data sets. Bordoloi and Shen[2] define a new measure for volume data sets, called noteworthiness. The measure is a function of opacity and frequency in the histogram of data sets. In addition to the new measure, they also propose view similarity that measures the Jensen–Shannon divergence between samples. This allows us to compare view samples and to determine how two views are similar to each other. The idea of comparing samples with the measure is similar to the study by Sbert et al.,[3] where the Kullback–Leibler distance is used. Takahashi et al.[4] propose another method for viewpoint selection. In this approach, they decompose the entire volume into multiple *interval volumes* (IV). Each IV is weighted by the sum of opacity values in the IV, so that it can incorporate the information defined in the transfer function. Ji and Shen[5] further investigate different measures for volume rendering. They utilize opacity, color, and curvature, which are all important information in volume rendering. The final viewpoint is computed by summing up three measures with predefined weights. Prior knowledge about the data helps to effectively determine the weights for specific data sets. More recently, Tao et al.[6] introduce two view descriptors: shape view descriptor and detail view descriptor. The shape view descriptor favors a scene in which visible boundary structures face the viewing direction. The detail view descriptor captures the local features in the data set.

Feature selection

The goal of our feature selection algorithm is to find all the features in a data set that would help the viewer understand the data. Corners of an object and high-intensity points are important features that can help the viewer understand the object. In order to find these features, our algorithm uses two steps: *feature selection* and *filtering*. The feature selection algorithm computes a score for each voxel, while filtering picks up only a small set of voxels, the *interest points*.

Feature selection algorithm

Our feature selection algorithm is based on the Harris interest point detection algorithm.[9] Next, we briefly review the Harris interest point detection algorithm and then present how we apply it to our problem.

Background. The Harris interest point detection algorithm[9] produces a score for each pixel in 2D images. Positive score values correspond to interest points in the image. A pixel is selected as a point of interest if there is a large change in both the X-axis and the Y-axis. For example, the algorithm gives a large positive score to a corner of a shape or a point that has a high intensity. On the other hand, the algorithm maps points in a flat area to zero and edges of an object to negative values. Corner points are important features for capturing the geometry of an object. If all the corners of an object are exposed when viewed from a viewpoint, the number of degenerate surfaces is minimized, and therefore, users can understand the overall geometry of the object. High-opacity points, which are also captured as the Harris interest points, are also

crucial features because users usually set the opacity of important areas high so that they can see the objects clearly.

The algorithm measures the change $E(x, y)$ in intensity at pixel (x, y)

$$E(x,y) = \sum_{u,v} g(u,v)|I(x+u,y+v) - I(x,y)|^2 \quad (1)$$

where $I(x, y)$ is the intensity value at image coordinate (x, y) and $g(u, v)$ is defined as a Gaussian weight around (x, y). If a point (x, y) is in an area that has less change in intensity, $E(x, y)$ will be close to 0. On the other hand, around the area that has a large change, $E(x, y)$ is also large.

The computation of $E(x, y)$ is done using the Taylor expansion

$$\begin{aligned} E(x,y) &= \sum_{u,v} g(u,v)|I(x,y) + xI_x(x,y) + yI_y(x,y) - I(x,y)|^2 \\ &= \sum_{u,v} g(u,v)[xI_x(x,y) + yI_y(x,y)]^2 \\ &= Ax^2 + 2Cxy + By^2 \end{aligned} \quad (2)$$

where A, B, and C are Gaussian convolutions of I_x^2, I_y^2, and I_xI_y, respectively

$$\begin{aligned} A &= g \otimes I_x^2 \\ B &= g \otimes I_y^2 \\ C &= g \otimes (I_xI_y) \end{aligned}$$

Then, $E(x, y)$ can be written in matrix form

$$E = \begin{bmatrix} x & y \end{bmatrix} \begin{bmatrix} A & C \\ C & B \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad (3)$$

Therefore, the intensity change in E is characterized by the 2×2 matrix in equation (3), which we refer as M, and if the change is large in both directions, the two eigenvalues of M should be large. The eigenvalues of M can be computed by singular value decomposition (SVD), but it needs a long computation. In order to avoid the SVD computation to find eigenvalues for each point, Harris and Stephens[9] proposed a response function as follows

$$R = det(M) - kTrace(M) \quad (4)$$

where k is an empirical constant, usually set between 0.04 and 0.06 for 2D images. Points that have large changes in both directions have two large eigenvalues of M. If the eigenvalues are significantly large, $det(M)$ impacts more on the value of R than $Trace(M)$. If only one eigenvalue is significant, R becomes negative as $Trace(M)$ is greater in the equation. In this case, the points represent edges because the change is significant in only one direction. We refer to R as "Harris score" hereafter.

Extension to 3D. In direct volume visualization, data are stored on a regular 3D grid. Thus, in order to use our approach with such data, we need to extend the Harris corner detection algorithm to the 3D spatial domain. Previously, Laptev and Lindeberg[18] extended the algorithm to the spatiotemporal domain, and Sipiran and Bustos[19] explored how the algorithm can be extended for 3D mesh data. We adopt the same structure as in Laptev and Lindeberg, but, in our case, the third domain is also a spatial domain. The change function $E(x, y, z)$ is defined as

$$E(x,y,z) = \sum_{u,v,w} g(u,v,w)|I(x+u,y+v,z+w) - I(x,y,z)|^2 \quad (5)$$

and the Taylor expansion for $I(x + u, y + v, z + w)$ is approximated as follows

$$\begin{aligned} I(x+u,y+v,z+w) &\approx I(x,y,z) \\ &+ xI_x + yI_y + zI_z + O(x^2,y^2,z^2) \end{aligned} \quad (6)$$

Following the same logic as in section "Background" yields the matrix M to be defined as follows

$$M = g \otimes \begin{bmatrix} I_x^2 & I_xI_y & I_xI_z \\ I_xI_y & I_y^2 & I_yI_z \\ I_xI_z & I_yI_z & I_z^2 \end{bmatrix} \quad (7)$$

This involves 3D convolution for each voxel, which is computationally expensive. By using GPU acceleration techniques, we were able to significantly improve the performance of this algorithm. We further discuss the performance of this algorithm in section "Implementation."

In our algorithm, the width of the Gaussian convolution kernel is set to 5; the convolution is a weighted sum of $5 \times 5 \times 5$ patch around a point (x, y, z). The variance of the Gaussian kernel is set to 1.5. The sensitivity constant k is set to 0.004.

Filtering

The extended Harris interest point detection algorithm produces a score for each voxel. The large score points are either at corners of the geometry or in bright areas. Because we are only interested in the largest score points, we need to filter out points that represent flat regions or edges. The algorithm for selecting interest points scans the entire score data, picking the largest N values.

The set of large score points can be reorganized as a matrix $X \in \mathbb{R}^{N \times 3}$, where each row represents the coordinate of the points selected as interest points in the

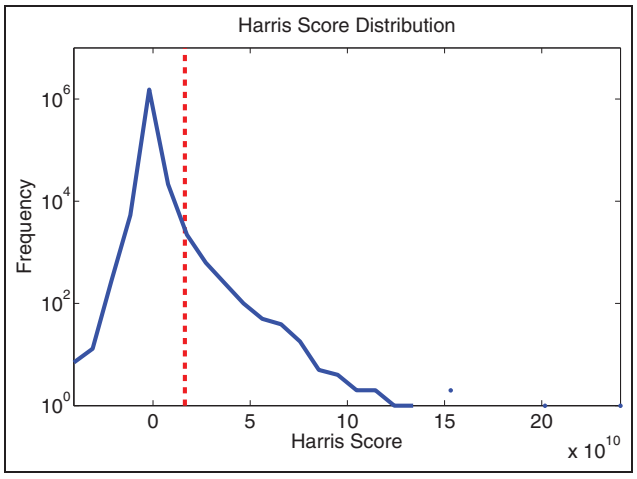

Figure 1. Harris score distribution of the tooth data. The solid line shows the histogram of the Harris score of the tooth data set. Positive values are considered interest points or corner points, near zero points are flat areas, and negative values indicate edges. The dotted line shows the threshold for X. All the points on the right-hand side of the dotted line are added to X.

previous step. N is set to 2048 in our algorithm, which is empirically determined. If we select too few points, we fail to capture all the important points in the data set. On the other hand, if we include too many points, we may end up adding noisy data points, for instance, points in flat areas or at edges. In addition, having many points in X increases the computational cost (1) in the filtering because the size of the buffer during the search increases and (2) in the optimization solver discussed in section "View selection" because X impacts the performance of the solver.

Figure 1 shows the Harris score distribution of the tooth data set. The maximum Harris score for this data set is 2.5×10^{11} and the minimum is -4.6×10^{10}. The distribution has a peak around 0, which is typical for many volume data sets because empty regions have scores around 0. Note that, however, having both positive and negative Harris scores is not typical. Depending on the characteristics of the data, it may not have many corner points, for instance, when an object has a very smooth surface. The constant k plays an important role in determining the shape of distribution.

Figure 1 also shows the threshold for the largest N values. All the points that have a Harris score greater than the threshold are included in X, and there are 2048 such points in this example. The position of the threshold shows that if we include more points, the line moves to the left, which will include some points that may not be interest points. The number of elements N in X is empirically determined by checking the distributions of R. If we select too few points, we fail to capture all the important points in the data set. On the other hand, if we include too many points, we may end up adding noisy data points, for instance, points in flat areas or at edges. In addition, having many points in X increases the computational cost (1) in the filtering because the size of the priority queue increases and (2) in the optimization solver discussed in section "View selection" because X impacts the performance of the solver.

Figure 2(a) and (b) shows examples of interest points. We applied the interest point detection algorithm and selected only large score points as described in this section. Each green rectangle denotes an interest point detected by our algorithm. The engine block

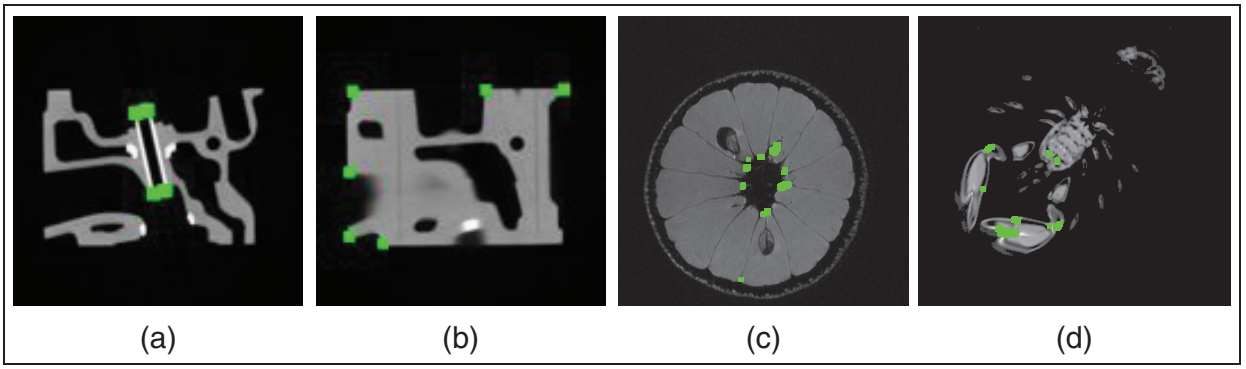

Figure 2. Examples of Harris interest points. Results for the engine block CT scan data from General Electric, USA. (a and b) The result of the 3D Harris interest point detection for the engine block CT scan data from General Electric, USA. We highlighted interest points with green squares. In (a), four corners of the cylinder are selected as interest points. (b) The outermost slice of the engine block and six corners are detected as interest corners. The interest points of the orange in (c), highlighted in green, include the center of the orange pieces and a small seed. The data are from the Information and Computing Sciences Division, Lawrence Berkeley Laboratory, USA. The lobster data shown in (d) are from the VolVis distribution of SUNY Stony Brook, NY, USA. The claws and the joints of the lobster have interest points.
CT: computed tomography; SUNY: State University of New York; 3D: three-dimensional.

computed tomography (CT) scan data set has two cylinders in the block, and Figure 2(a) shows that the top and bottom of the cylinders are selected as interest points. Figure 2(b) shows that the algorithm successfully identifies most corners in the last slice for the block. If we add more points in X by finding more large values, additional corners will be eventually added, but we did not see any improvement in the viewpoint selection.

Figure 2(c) and (d) shows two other data sets. A magnetic resonance imaging (MRI) scan of an orange is from the Information and Computing Sciences Division, Lawrence Berkeley Laboratory, USA. Figure 2(c) shows one slice of the data set with the interest points highlighted. Points on the edges of the orange pieces and one seed are selected as interest points. Two bigger seeds are too big to be captured as features by our algorithm. The second data are a CT scan of a lobster. The data are from the VolVis distribution of State University of New York (SUNY) Stony Brook, NY, USA. The interest points include two claws and joints as highlighted in Figure 2(d). The legs are not selected as interest points because the opacity of the legs is small so the difference in the adjacent voxels is not as big as for other points.

View selection

The previous steps generate a set of feature points. Those points are clustered around the corners of high-opacity region. Those points are important in viewpoint selection (1) because we do not want the corners of an object to overlap each other and (2) because we want to minimize the case when high-opacity points are occluded by each other. Corner points capture the geometry of the objects in the data. If the corner points have minimum occlusion, there is a higher chance that users can see the data from the canonical view of the data, revealing more details of the data. If the data have outstanding features, which are captured in the Harris interest points, and if those features have minimum occlusion, we can still see important objects in the data. Because the Harris interest point algorithm favors high-opacity values, high-intensity points, that is, opaque objects, are more likely to be selected as the Harris interest points. If our view selection algorithm can minimize the occlusion between the points, the occlusion between opaque objects in the data is also minimized.

Therefore, in the next step of finding the best viewpoint, we need to minimize the occlusion between the feature points. Minimizing the occlusion can be thought as maximizing the distance between the feature points, that is, spreading out far apart from each other. The direction that minimizes the occlusion can be computed by solving an optimization problem, which has the same formulation as PCA.[20]

PCA

We employ PCA to find the best view direction from the matrix X. The main idea of PCA is to find the direction $\hat{u}$ that maximizes the variance of the projected points of X when projected to $\hat{u}$. The optimization formulation is

$$\begin{aligned} &\text{maximize} Var(uX) \\ &\text{subject to} ||\vec{u}||^2 = 1 \end{aligned} \tag{8}$$

Without loss of generality, we can assume that X is centered, that is, the mean of points in X is (0, 0, 0). Then

$$Var(uX) = \frac{1}{N}\sum_{i=1}^{N}(X_i \cdot u)^2 \tag{9}$$

The detail of how to find the solution for the optimization problem in equation (8 can be found in the literature,[20] but the solution is derived from the Lagrange dual function[21] and is the eigenvector with the largest eigenvalue of the covariance matrix of X. That is, the best viewpoint is the solution of $X^T X u = \lambda u$. Note also that the solution is the global maximum, and therefore, it always gives the best possible direction out of all possible choices of u. This means that our algorithm does not trade accuracy for performance. In addition, there is no trial-and-error parameter tuning process to find the solution.

By maximizing the variance of projected points, we expect that we minimize the overlap between the selected interest points. By placing interest points far apart, we can see all the interest points in the data set with little—or ideally, no—occlusion. For example, suppose the data are just a 2D rectangle in 3D space. The corner detection algorithm picks up the four corners of the rectangle. Maximizing the variance of the four points produces the direction parallel to the normal vector of the plane, which is the best viewpoint. Degenerative plane or a single line is achieved when the variance is minimized, that is, when the viewpoint is set to perpendicular to the normal vector. Therefore, our best viewpoint guarantees that one can see as many details of the volume as possible, although we project 3D data onto a 2D plane.

We use SVD[22] for the actual computation. SVD decomposes the input matrix X into $U\Sigma V^T$, and the right singular vector Vs are the eigenvectors of $X^T X$. Suppose SVD produced V as $[v_1|v_2|v_3]$, then the

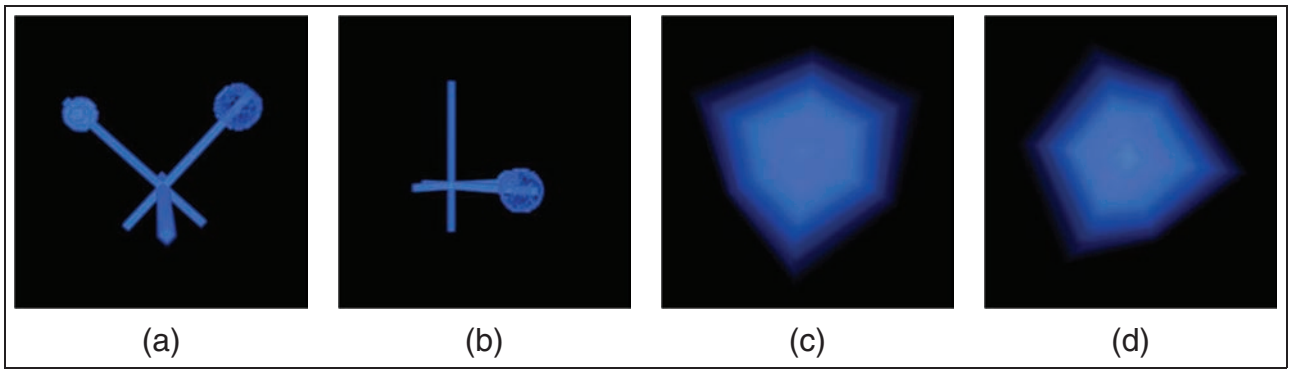

Figure 3. The results for the cross and box data from the Computer Graphics Group, University of Erlangen, Germany. The transfer function is set to show only the inner part of the cross data, leaving out outer faces. (a and b) While the cross data set shows big difference between best and worst views, (c and d) it is hard to tell which view works better for box data. The eigenvalue spectrum (shown in Figure 4) of these two data sets explains the reason why.

interest points yield the maximum variance along direction v_1. In order to see the largest variance in the rendered image, the camera's view direction should be v_3, looking at the center of the volume, because v_3 is orthogonal to two directions v_1 and v_2, the two axes in which the points achieve maximum variance. On the other hand, if the points are projected along direction v_3, the variance of the projected points are the minimum among all possible choices of u in equation (8). Thus, the "worst viewing" direction is from v_1, orthogonal to v_3 and v_2.

We also considered using the Harris score values as weights for uX in equation (8), so that larger weight points should be placed far apart even if small weight points are close to each other. This is because if two points that have large weights and are placed close to one another, the variance is penalized heavily. However, the Harris score values do not proportionally correspond to the likelihood of being interest points, that is, a point having twice as large a score as another does not mean the point is twice more likely to be an interest point. Moreover, the percentage of points we select is relatively small and those points are all important features in the data. Thus, we do not leverage the Harris score values themselves in the optimization.

Note the viewpoint direction is determined by the two principal components, and the eigenvalues indicate how much variance is generated by each eigenvector. Thus, by checking the eigenvalues of a data set, one can figure out how much symmetry the data have. The following two data sets show an example of how an eigenvalue spectrum indicates the symmetry of a data set.

Figure 3 shows the results for two different data sets. One is the cross data set and the other is the box data set, both from the Computer Graphics Group, University of Erlangen, Germany. We set the transfer function, so that the cross data set shows only the three rods and two balls attached at the end of two rods and the box looks like a cube. The interest points for the cross data are six ends of the three rods and points in the balls. Therefore, the best view is realized as in Figure 3(a). Here, we can see the two balls placed far apart although one rod perpendicular to the two rods having balls has to be perpendicular to the view direction. For this data set, one can tell the difference between the best and worst views very easily. However, for the second data set, the box, it is not clear which image shows more information than the other due to the symmetric shape.

As mentioned, this can be explained by the spectrum of eigenvalues, and this spectrum is one way of evaluating the quality of the PCA. From the SVD computation, we get three eigenvalues and each eigenvalue is proportional to the variance in that direction. Therefore, if the largest eigenvalue dominates the other two, it means that the variance of the points is mostly captured on the axis defined by the first eigenvector. In contrast, if three eigenvalues are more or less the same, the rendered images from three different directions produce similar results in terms of variances.

This difference is verified by the eigenvalue spectrum in Figure 4. For the cross data set, the first and second eigenvalues occupy 95.6% of the sum of the three eigenvalues. This indicates that the plane perpendicular to the third eigenvector, that is, the view direction, holds 95.6% of the variance of the interest points. On the other hand, the box data set produces three almost identical eigenvalues. This means that the variance of the interest points is equally explained by each axis. No matter where the data are viewed from, it produces equally good images. Three (almost) identical eigenvectors tell us that the variance of interest points in Figure 3(c) and (d), as well as a view from

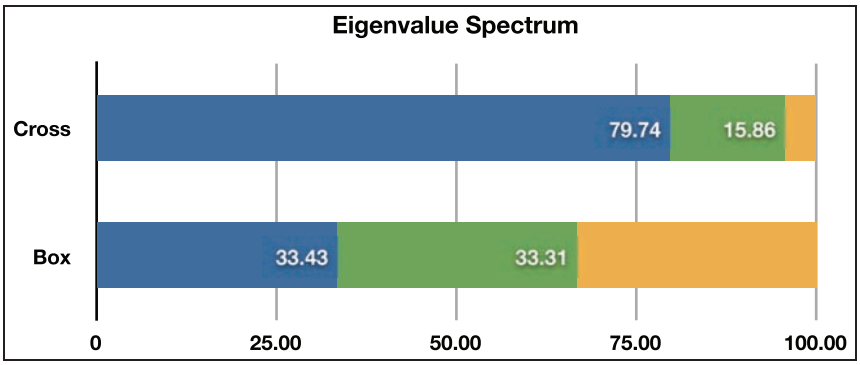

Figure 4. Eigenvalue spectrum for two different data sets. First and second eigenvalues of cross data occupy 95.6% of the three eigenvalues, which means the variance of projected points on the plane created by the first and second eigenvectors explains 95.6% of the entire variance in the original 3D space. Therefore, the projection does not lose much information. On the other hand, the box data set is symmetrical and there is little difference between best and worst views as shown in Figure 3. This is because the first and second eigenvalues add up to only 66.74% and the three eigenvalues equally split the entire 100%. This eigenvalue spectrum tells us how well the best view shows the volume data, and the spectrum gives a quantitative measure for the quality.
3D: three-dimensional.

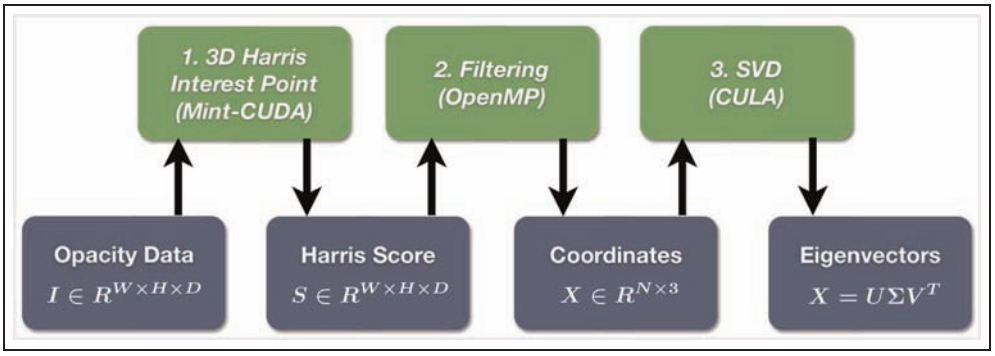

Figure 5. The computational pipeline for our viewpoint selection algorithm. The input to the algorithm is 3D opacity data stored in a structured grid. The value ranges from 0 to 255 (8 bits). The 3D Harris interest point detection algorithm produces the Harris score values for all voxels. The filtering step searches the largest values of the score and the (*x, y, z*) coordinates for the values. Finally, the last step executes the SVD of *X* and computes the eigenvectors of *X*. Note that the "3D Harris Interesting Point" and "SVD Algorithm" steps run on the GPU, whereas the filtering step is done on a multi-core CPU with OpenMP.
CPU: central processing unit; CUDA: Compute Unified Device Architecture; SVD: singular value decomposition; 3D: three-dimensional.

the side or any random direction, is identical. This comes from the symmetry of the shape, and regardless of the position, one-third of the total variation is lost during the rendering projection. This example is the worst possible case for PCA in maximizing the variance. However, this perfect symmetry rarely happens in real data sets, and even in this worst case, the algorithm correctly suggests a viewpoint that human perceives as the best.

Implementation

We implemented the algorithm described in sections "Feature selection" and "View selection." There are three main kernels to compute the best viewpoint: (1) 3D Harris corner point, (2) filtering, and (3) PCA. Figure 5 shows the pipeline of the algorithm. The input to the algorithm is the opacity data modified by the transfer function. The output of the algorithm is the best direction for displaying the volume given the settings of the transfer function.

The 3D Harris interest point algorithm (section "Feature selection algorithm") takes the opacity data and produces a Harris score for each voxel. This kernel is the most computationally challenging part of the three kernels, as it requires convolution, which performs many memory accesses and arithmetic operations per voxel. Since our goal is to make the entire

algorithm run in real time, an implementation on a traditional multicore architecture would not have met our goal. However, the algorithm is a good candidate for GPU acceleration because it is highly data parallel and can benefit from the high bandwidth to GPU memory. Therefore, we ported the algorithm to the GPU. We use the Mint source-to-source translator[23] to automatically generate the CUDA[24] code for the 3D Harris interest point algorithm. The Mint translator takes C code with simple annotations and generates optimized CUDA code. The automatic optimizations in the translator include shared memory optimization to reduce memory access operations, boundary condition optimization, loop unrolling, and constant folding. The Mint-generated CUDA implementation allowed us to achieve a substantial performance improvement over the single central processing unit (CPU) implementation, delivering a 45× speedup. More details about the parallelization and the implementation on GPU are described in the study by Unat et al.[25]

The second kernel is filtering (section "Filtering"). The input to this kernel is the Harris score computed in the previous step and the output is a matrix X, that is, the list of coordinates for largest Harris scores. Because the Harris score is stored in GPU memory and we compute the filtering step on the CPU, we perform memory transfer from GPU memory to CPU memory. The asymptotical running time of this step is linear in the number of voxels. In order to further improve the performance, we parallelized this kernel with OpenMP.[26] The parallelization algorithm works as follows: multiple threads equally divide the entire Harris score data and each thread searches the largest N values. After the parallel section is done, a single thread merges the results into one. We set N to 2048 for all our results. This is just one of many possible implementation methods, and it is feasible to implement the filtering step on GPU because filtering is just a reduction operation.[27] Furthermore, the transfer time is already included in the running time reported in Table 2, and the bandwidth between the GPU and CPU memory via PCI-Express is large enough to quickly transfer any reasonable volume data. Regarding the PCI-Express transfer time, it has been reported in other works extensively.[28]

The last step is to compute the eigenvectors of X. The output of this step is a vector in 3D, which is the best viewpoint determined by our algorithm. Because the size of X is a constant, 2048×3 in our case, the computational overhead for this step is trivial compared to the previous steps. We used the Compute Unified Linear Algebra (CULA) library,[29] a linear algebra package in CUDA. Thus, the only part not implemented in CUDA is the second kernel: filtering. We are also in the process of off-loading the second kernel to the GPU, although the cost of the memory transfers between CPU and GPU memory for the kernel is not as high. In terms of memory transfer between CPU and GPU, if rendering takes place in GPU, the entire intensity value needs to be transferred into GPU texture memory. Our algorithm can access the data directly.

Impact of parallelization

Table 1 reports the speedup of the Harris interest point detection algorithm among our three implementations in three different methods: single CPU, multicore with OpenMP (quad-core Intel Xeon CPU 2.0 GHz), and CUDA (Tesla C1060 GPU). The running time column shows the running time measured for the Harris interest point detection algorithm. The serial

Table 1. Performance comparison of the Harris interest point detection algorithm among different architectures.

Data set	Data size	Running time (s)			Speedup	
		Serial	OpenMP	Mint-generated CUDA	Serial	OpenMP
CT knee	379 × 229 × 305	24.474	6.392	0.797	30.71	7.58
Foot	256 × 256 × 256	15.580	3.933	0.340	45.82	11.57
Engine	256 × 256 × 256	15.375	3.927	0.340	45.22	11.55
Lobster	301 × 324 × 56	5.012	1.466	0.132	37.97	11.11
Orange	256 × 256 × 64	3.882	0.986	0.099	39.21	9.96
Tooth	94 × 103 × 161	1.427	0.426	0.044	32.43	9.68
Cross	66 × 66 × 66	0.264	0.072	0.0106	24.91	6.79
Box	64 × 64 × 64	0.244	0.062	0.010	24.40	6.20

CPU: central processing unit; CT: computed tomography; CUDA: Compute Unified Device Architecture.
The serial version is run on a single CPU, the OpenMP version is run on a quad-core, and the Mint-generated CUDA version is implemented in Mint-generated CUDA. The first three columns show the running time of the Harris interest point detection algorithm in seconds. The last two columns show the speedup of Mint-generated CUDA against the serial version and the OpenMP version. The CUDA version achieved huge speedups ranging from 24 to 45 against the serial version. Even compared with the OpenMP version, the CUDA version ran more than six times faster.

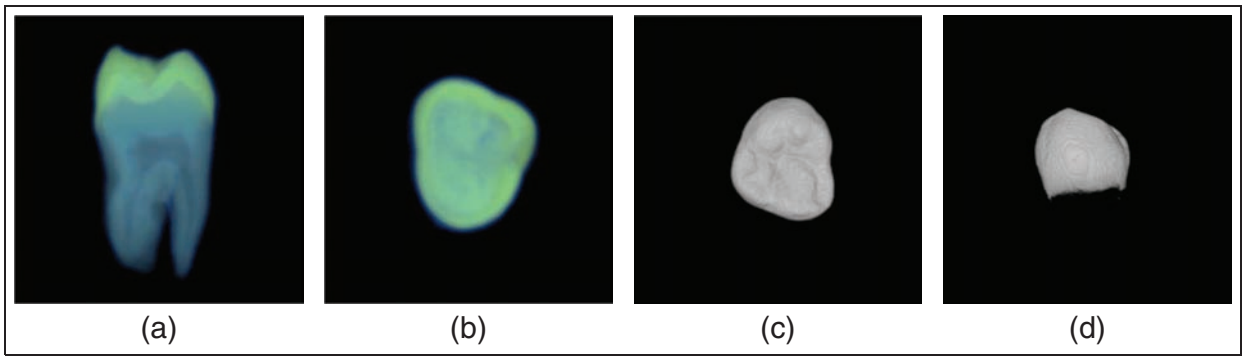

Figure 6. Results for the tooth data sets with two different transfer function settings. (a and b) The best and worst images of the tooth data set. (a) Shows the entire shape of the tooth by looking at it from the side, while (b) renders the tooth from the top and it fails to spread out interest points. On the other hand, we modify the volume with a 1D transfer function and render only (c) the crown area without the root of the tooth. In this case, looking from the side does not reveal any particularly interesting information, which is shown in (d). The most interesting information is the shape of the enamel, and thus, viewing from the top shows the interest points more effectively.
1D: one-dimensional.

Table 2. Performance comparison for various volume data sets.

Data set	Data size	Our approach		Ji and Shen		Tao et al.	
		Running time (s)	Rating	Running time (s)	Rating	Running time (s)	Rating
CT knee	379 × 229 × 305	0.8128	4.30	6.8281	4.30	18.2919	5.15
Foot	256 × 256 × 256	0.5906	5.20	8.4182	5.90	14.0245	4.85
Engine	256 × 256 × 256	0.5036	4.60	8.6340	5.60	13.9101	6.00
Lobster	301 × 324 × 56	0.2619	5.27	5.6118	2.23	8.2253	5.82
Tooth	94 × 103 × 161	0.1345	4.42	5.0289	5.60	6.5794	5.40

CT: computed tomography.
We show the running times and user study rating for various data sets. Entropy-based approaches, Ji and Shen[5] and Tao et al.,[6] are significantly slower than our approach. Our approach runs in less than a second in all data sets, which makes it feasible to be used along with transfer function exploration. The average ratings from our pilot study show that our algorithm achieves comparable image qualities to the previous approaches.

version computes the values with a single CPU, whereas the OpenMP version utilizes all four cores available. The Mint-generated CUDA version uses one of the available GPUs for the main computation. The last two columns, Speedup, show the speedup of the Mint-generated CUDA version against the serial and OpenMP version. The largest speedup achieved is with the foot and engine data, 45× speedup against the serial version and 11× speedup against the OpenMP version. The benefit of parallelism decreases as the size of the volume data set decreases (note a 45× speedup for foot data set and a 24× speedup for box data set). The overhead of transferring the data back and forth between the GPU memory and the main memory and the latency for launching the computational kernel on GPU caused the decrease. However, even with the smallest data set, the speedup and the running time itself indicate that we achieved a substantial performance improvement against other implementations.

Figure 6 shows the results of our algorithm with the tooth data set. We applied two different transfer functions to highlight different parts of the data. Figure 6(a) and (b) shows the best and worst results for the entire tooth shape. The best direction sets the view from the side allowing viewers to see the entire root of the tooth. The worst direction projects the data upward. This occludes many interesting parts of the data set, for example, the shape of the root. The second setting of the transfer function kills the signal in the root, leaving only the crown area of the tooth. Then the interesting part of the data is the shape of the enamel. The best view from our algorithm successfully shows that area in Figure 6(c). The worst view in Figure 6(d) fails to capture the interesting part of the data. This example illustrates an important point; transfer function design and the viewpoint selection algorithm should be tightly integrated with each other. Transfer functions can change the visual representation of the data set significantly as shown in Figure 6,

and depending on the setting of the transfer function, the area of interest changes as well. Therefore, the result of the viewpoint selection algorithm can change dramatically with the transfer function. Updating the results in real time based on the transfer function is critical for understanding the volume data sets.

Evaluation

In this section, we compare our algorithm to two state-of-the-art algorithms in terms of running time and visual quality. We measured the running time under the same conditions and conducted a small survey to rate the quality of the images.

Other approaches

We compare our algorithm with two previous approaches.[5,6] The approaches based on information theory evaluate the amount of information contained in each view. Bordoloi and Shen[2] define the PDF on voxels of the data. At each viewpoint, the PDF changes. Ji and Shen[5] and Tao et al.[6] define the PDF on the pixels of the rendered image. Therefore, the evaluation of entropy value requires a rendering process for each view. Because none of the three approaches considers any acceleration method for the search for best viewpoints, the basic mechanism for search is brute force; it evaluates as many points as possible and chooses the best one, which is the one that achieves the largest entropy. Then, there is a trade-off between the computation time and the correctness of the result. If one increases the number of samples, it is possible to find a viewpoint close to the optimal viewpoint. However, this presses the computation time because the running time of the algorithm is proportional to the number of renderings, which is very expensive. On the other hand, if one wants to find a solution in a reasonably short time, the only way is to reduce the number of samples. Then, the average distance between samples increases, and it is very likely to miss the optimal solution.

For the sake of comparison, we implemented two state-of-the-art approaches, one by Ji and Shen[5] and the other by Tao et al.[6] Ji and Shen use opacity of the rendered image as their quality metric. If there are more pixels with a high opacity in an image, they consider the image to be more important. Therefore, they render the volume with opacity data and measure the entropy value of the opacity distribution. The rendered image with the highest entropy value is considered the one with the best viewpoint. Although color and curvature information are used to measure the quality of a view, it requires multiple rendering processes to get entropy values from each measurement. This significantly slows down the running time, but we were able to get very good results with only opacity, which we discuss in detail in section "Result comparison."

The algorithm by Tao et al. is based on the same framework, but they measure a *shape descriptor* and a *detail descriptor*. Their shape descriptor measures the information that represents the overall structure of a volume data set, while the detail descriptor measures the small details of the data. The former is computed from the data created by bilateral filtering.[30] The algorithm filters the original volume, which smoothes out noisy information. After the filtering, the resulting volume contains only the overall structure. Then, if a majority of surfaces in the volume is orthogonal to the viewing direction, it means that the volume exposes more surfaces, which they argue provides more information.

On the other hand, the difference between the original volume and the filtered volume represents small details that do not affect the overall structure. The authors also argue that a good view should be able to expose this detail. Therefore, they find a direction where this detail can be maximally exposed. The two pieces of information are combined by a user-defined weight. Because the data we used for our experiments do not contain local features, we decided to put a 0.8 weight on the shape descriptor and a 0.2 weight on the detail descriptor. Both descriptors require gradient computation, and the gradient for each fragment is computed in a shader.

Performance comparison

Both approaches (Ji and Shen and Tao et al.) were implemented in C++ with OpenGL. The volume-rendering algorithm uses 3D textures, and the final image is a result of blending many parallel volume slices. In order to compute the correct view descriptor values for entropy-based approaches with minimum overhead, we use GLSL shader programs. Our window size (OpenGL viewport) is 512×512 pixels. The best viewpoint is selected from 500 viewpoints randomly selected on the viewing sphere. The geometrical center of each volume is located at its origin, and the camera look-at direction is this origin for all viewpoints. For our algorithm, as mentioned in section "Implementation," we used the Mint translator (in version 0.3) to generate the CUDA code for our algorithm. This code was subsequently compiled with the NVIDIA nvcc compiler and linked with the CULA R10 library. We used version 3.2 of the CUDA Toolkit.

For the sake of a fair comparison, we ran all tests on the same computer. The graphics card and CPU in this machine are an NVIDIA GeForce GTX 285 and an Intel Core i7 with 2.93 GHz. We measured the running time 10 times for each case and report the minimum in

Table 2. During the measurement, we excluded the time for bilateral filtering in the algorithm by Tao et al., which takes more than 10 s for typical data sets. We decided not to add this time to our reported numbers because many algorithms have been proposed[31] that allow us to do the preprocessing more efficiently than the original algorithm. Implemented in CUDA, it would be possible to run it in less than a second.

Table 2 shows the running time of the three algorithms. Our approach vastly outperforms Ji and Shen and Tao et al. On average, our approach is 27 times faster than the algorithm by Ji and Shen and 38 times faster than the algorithm by Tao et al. An important aspect of the running times is that our algorithm runs in less than a second, which enables us to run our algorithm interactively in any volume-rendering systems. More than 5 s running time of previous approaches limits their use by interactive applications. We will discuss the benefits of our algorithm further in section "Applications."

Result comparison

Figure 7 shows the results from our algorithm and previous approaches. In the first row of Figure 7, we compare the best view for the CT knee data. Figure 7(a) and (d) renders the data from the back and the front, respectively. One can easily see that the data show the symmetric shape of the legs and that they render the knee by finding the shapes of its bones. Figure 7(c) shows two legs but the legs slightly overlap each other because the viewpoint is from the side, making it difficult to distinguish them. The second row of Figure 7 shows the foot CT scan data. The viewpoints of Figure 7(e) and (g) are from the top, both of which show the data effectively. However, the viewpoint for Figure 7(h) is slightly tilted. One can still understand the shape of the foot in Figure 7(h). The third row of Figure 7 renders the engine CT scan data set. The result of our approach (Figure 7(i)) appears to be slightly less optimal than Figure 7(k) and (l). However, in our approach, the tip of the cylinder is clearly visible, while the other two approaches failed to do that. For the tooth data, most algorithms show similar results, as shown in the fourth row of Figure 7. The viewpoint from the side generates a large number of nonzero pixels (for the algorithm by Ji and Shen) and a large portion of surfaces in the data is exposed (for the algorithm by Tao et al.), making the best viewpoint one from the side. The lobster data set shown in the last row of Figure 7 looks best with our algorithm. Our result (Figure 7(q)) shows the lobster from the top, revealing all the details from the claws to the tail. Figure 7(t) shows almost identical results where the viewpoint is set slightly tilted to the side. The result with Ji and Shen's algorithm (Figure 7(s)) shows the lobster from the side. This result is because the algorithm favors an image that has many nonzero pixels. Due to the body and the claw, the projected area of the lobster is maximized when projected from the side.

Pilot study

In order to further confirm that our approach produces comparable results to the previous algorithms, we conducted a small survey among our team of researchers to provide quantitative measures for the image quality across multiple algorithms. "Comparable results" here mean that we are not arguing that our method produces significantly better results: our method is rather focused on the interactive performance, which we already showed in the previous sections, and the results from our pilot study prove that our method produces no worse results than competing methods, which take at least a few seconds.

We created an online questionnaire and sent it to graduate students and research staff we work with. We received 21 fully answered questionnaires back. The participants were asked to rate each image in Figure 7 on a scale from 1 (worst viewpoint) to 4 (neutral) to 7 (best viewpoint).

The average ratings are listed in Table 2 and range from 4.42 to 5.20. This range is slightly lower than the other two approaches: the algorithm by Ji and Shen achieved between 4.30 and 5.90, although the rating for the lobster data is extremely low (2.23); the algorithm by Tao et al. achieved the best rating, ranging from 4.85 to 6.00. The average ratings for the images with the worst viewpoints, shown in the second column in Figure 7, are consistently very low (between 1.95 and 2.80).

Result for multichannel data

We applied our method to a multichannel data set. When investigating multichannel data, viewpoint is more critical because each channel renders different organs in the specimen and scientists usually turn on/off a few channels and see how each channel looks or compare two channels together. In order to investigate two channels, scientists usually have to rotate the volume that shows the two channels most effectively.

We use the zebra fish head data, courtesy of Hideo Otsuna and Yong Wan from the University of Utah. The zebra fish head data consist of three channels: the first channel, rendered in red, shows eye muscles; the second channel, rendered in green, expresses eye, retina ganglion cells, and tectum; and the third channel, rendered in blue, expresses cell nuclei.

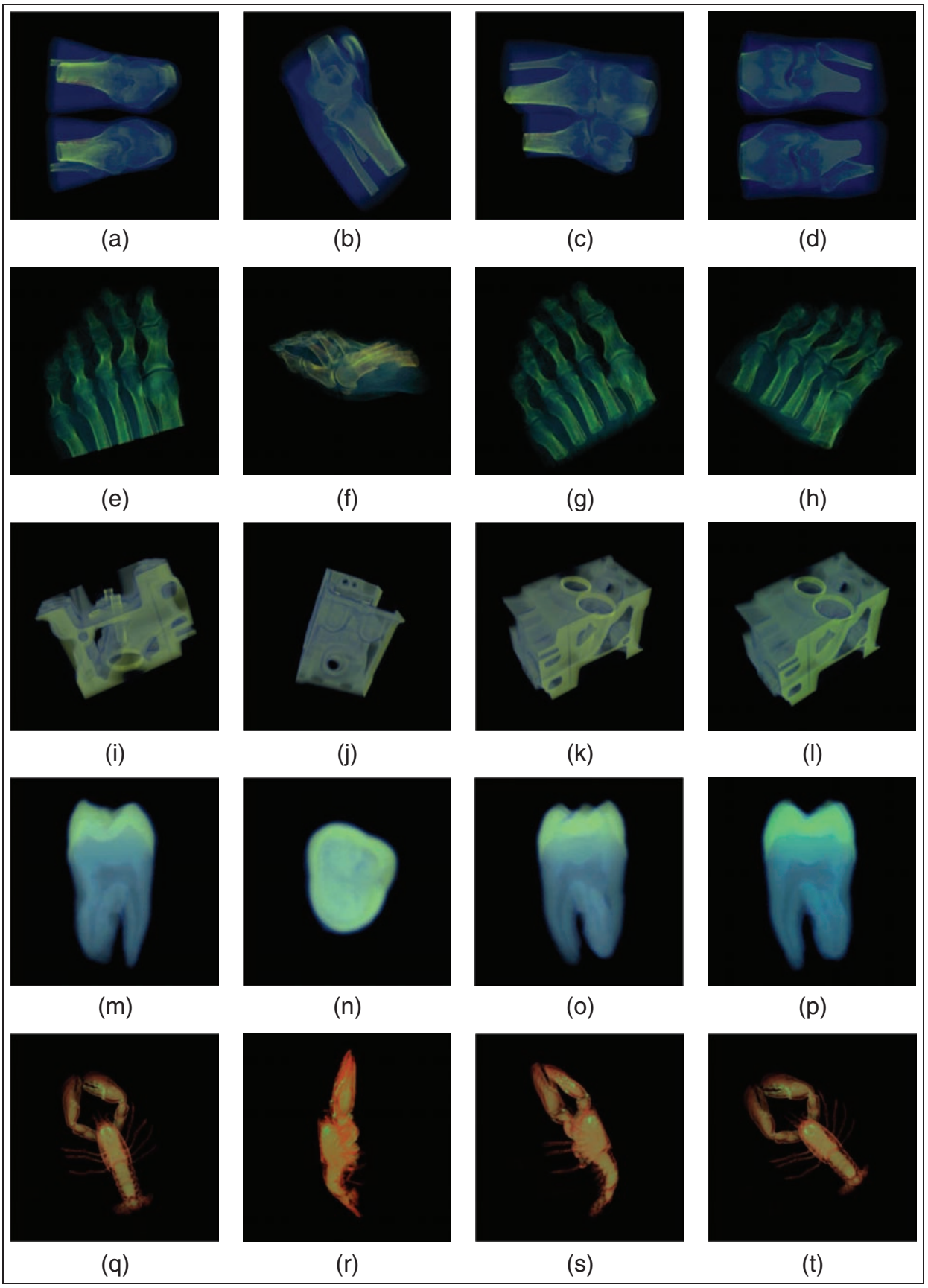

Figure 7. Comparison between other approaches: (a, e, i, m, and q) our approach, (b, f, j, n, and r) worst viewpoint, (c, g, k, o, and s) Ji and Shen, and (d, h, l, p, and t) Tao et al.

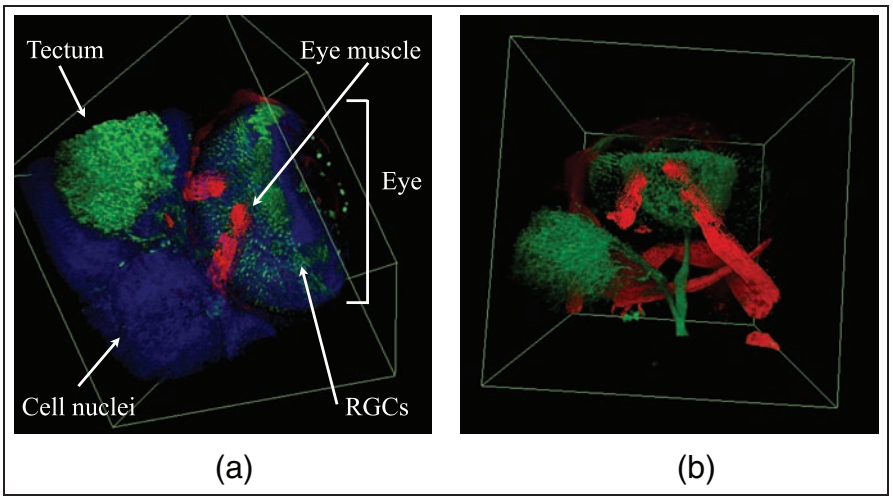

Figure 8. (a) Rendered images of a zebra fish head with all three channels turned on. The data show the area around the eye and the tectum. The eye muscle colored in red wraps around the retina ganglion cells (RGCs). (b) Rendered images of the zebra fish head with only two channels turned on. With the blue channel off, one can clearly see where the eye muscles are located around the RGCs.

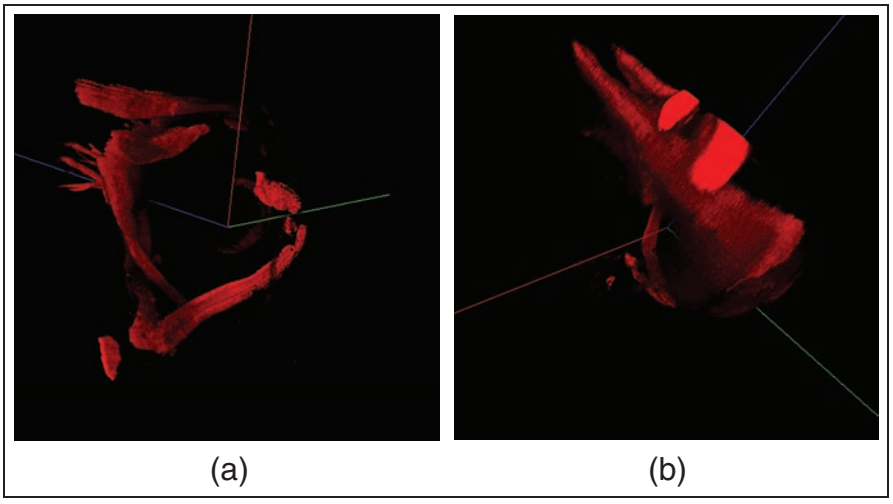

Figure 9. Results of our viewpoint selection algorithm applied to channel 1 of the zebra fish head data. One can understand the overall structure of the data from the best viewpoint.

All the rendered images in this section are rendered with a 3D texture-based rendering algorithm with alpha blending. The viewport size is set to 512×512. The center of each volume is located at the origin, and the view direction is set to look at the origin.

Figure 8(a) and (b) shows the overall screenshots showing different organs in the data. The figure shows the eye and brain region. The tectum and cells in the eye are colored in green. The eye muscles wraps around the eye region. The blue channel denoting cell nuclei wraps around the organs. Because scientists are usually more interested in channels 1 and 2, Figure 8(a) shows only two channels turned on.

As each channel already shows distinct objects, we show results of our view selection algorithm for each channel. Figure 9(a) shows the result for channel 1. From Figure 9(b), one can clearly understand how the eye muscles are spread out in the specimen. Figure 9(b), which is the worst viewpoint from our algorithm, prevents viewers from understanding the overall structure of objects expressed in channel 1. Figure 10 shows the second channel. Figure 10(a) clearly shows the two components, tectum and RGCs, in the data, with no occlusion, whereas in Figure 10(b), all the objects overlap each other. Figure 11 shows the third channel. The third channel contains cell nuclei, the small details of which

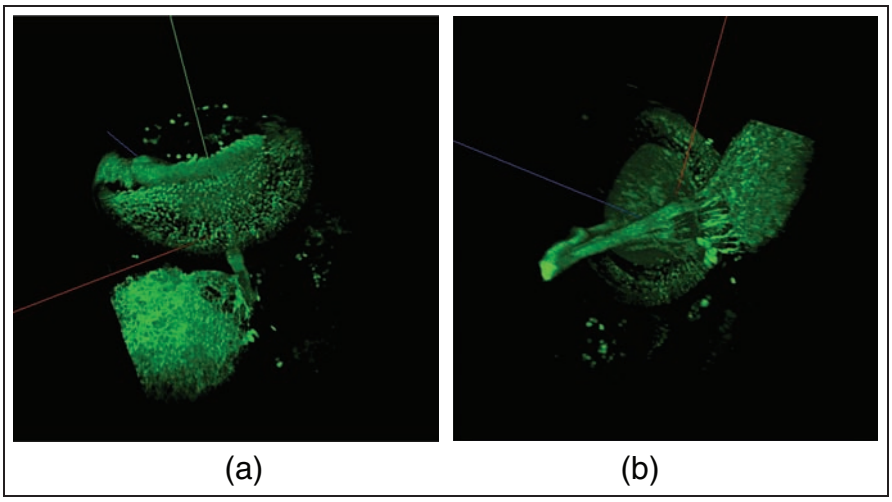

Figure 10. Results of our viewpoint selection algorithm applied to channel 2 of the zebra fish head data. One can see the shape of tectum and RGCs.
RGC: retina ganglion cells.

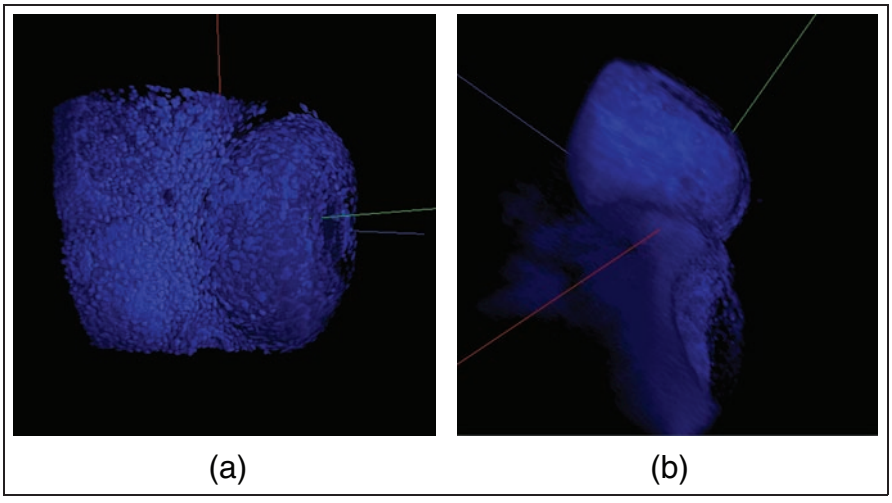

Figure 11. Results of our viewpoint selection algorithm applied to channel 3 of the zebra fish head data. One can see the cell nuclei on the surface from the best view.

can be seen in Figure 11(a). However, a side view, as shown in Figure 11(b), fails to display those details.

Discussion

Applications

Finding a good projection of a 3D data set to a 2D image plane is important in computer graphics in order to highlight certain features of the data set. As we have shown, our algorithm works well for volume data sets when using transfer functions and alpha blending, partly due to our parallelization for CUDA, which makes our algorithm real time. We propose that our algorithm is particularly suitable in the following two usage scenarios: interactive transfer function design and automatic thumbnail generation.

First, our interactive viewpoint selection can easily be incorporated into the transfer function design process by automatically rotating the volume to the best viewpoint once a new transfer function has been selected. This integration can be done by a simple engineering work; for instance, no viewport change happens while a user modifies transfer function, and as soon as user finishes transfer function exploration, a new viewpoint is computed. Instead of jumping to the new viewpoint, an animation effect that smoothly changes the viewpoint from the current to the new position will give the user a quick overview about the new transfer function.

Second, our viewpoint selection algorithm can rapidly produce thumbnail images for large databases of volume data sets. Due to the progress in 3D volume acquisition technology, it has become easy to generate many data sets in a short amount of time. These data sets are often stored in data banks and made available through web portals.[32,33] These web sites often use thumbnail images of the data sets as previews for users to distinguish different data sets without having to download them. It is not practical for the operators of such systems to manually rotate each data set to create good views for the snapshot. Our viewpoint selection algorithm can be used as a thumbnail generation tool for thousands of volume data sets in a very reasonable amount of time.

Limitation

Despite the sub-second performance and the simplicity of the algorithm, our approach has a few caveats, three of which we are going to address in more detail. First, our algorithm assumes that the rendered images would contain the most useful information if the interest points were placed far apart in the rendered images. That works for many data sets. However, the algorithm ignores any occlusion effects. If a data set is rather opaque and the visibility for the inside is limited, it may not be able to see through the volume. In this case, if interest points are selected on the inside, even if the variance of the points is maximized, the result may not produce the best view direction. However, most data sets for direct volume-rendering systems are more transparent. If a volume data set is rather opaque, there is not much benefit of rendering a scene with expensive volume-rendering process because isosurface rendering can produce better images with more efficient rendering.

A potential issue of our approach is that it uses parallel projection from the 3D data to the 2D image plane to determine the best view direction. In our experience, this works quite well, even if we render the data set with perspective projection, but in some cases, the interesting features might not end up in the optimal places when perspective projection is used to render the images.

Finally, PCA produces three eigenvectors, one of which we choose for the best viewpoint. However, the eigenvector is an axis, not a direction, on which you can see the interest points effectively. The variance we see from the eigenvector v is the same with $-v$ because all the points are projected to the plane orthogonal to v. For transparent volume data sets, this may not cause a big difference, but one direction may not generate as good an image quality as the opposite direction. One simple solution to this problem is comparing the entropy of the two rendered images to choose the better one. The probability density function for the image can be defined as in many other information theory–based frameworks.[2]

Conclusion and future work

We proposed a new approach for finding the best view direction for volume rendering. The algorithm locates interest points in a volume data set and finds a best view in which the interest points are spread out the most. The interest points are found by a 3D extension of the Harris interest point detection algorithm and the view direction is computed by solving a PCA problem. Our algorithm runs significantly faster than other approaches based on information theoretic frameworks. The performance benefit comes from the simplicity of the algorithm and an efficient implementation on the GPU architecture.

Although our algorithm assumes the Harris interest points as important features to look at, it would be interesting to consider different feature selection algorithms, for example, scale-invariant feature transform (SIFT).[34] Studying other alternatives of the feature selection, in terms of the effectiveness and efficiency, could also be valuable. We would also like to investigate the effect of PCA's linear projection when using perspective projection for rendering. To achieve this goal, we will have to formulate the optimization problem in a different form.

Funding

Didem Unat was supported by a Center of Excellence grant from the Norwegian Research Council to the Center for Biomedical Computing at the Simula Research Laboratory. Scott Baden was supported, in part, by the University of California San Diego and, in part, by the Simula Research Laboratory.

References

1. Vázquez P-P, Feixas M, Sbert M, et al. Viewpoint selection using viewpoint entropy. In: *Proceedings of the vision modeling and visualization conference 2001, VMV '01*, pp. 273–280. Aka GmbH, 2001 Stuttegart, Germany.
2. Bordoloi UD and Shen H-W. View selection for volume rendering. *Visualization, 2005. VIS 05. IEEE*, vol., no., pp. 487–494, 23–28 Oct. 2005 doi: 10.1109/VISUAL.2005.1532833.
3. Sbert M, Plemenos D, Feixas M, et al. Viewpoint quality: measures and applications. In: *Eurographics workshop on computational aesthetics in graphics, visualization and imaging* (eds L Neumann, M Sbert, B Gooch, et al.), Girona, Spain; 18–20 May, 2005. pp. 185–192.

4. Takahashi S, Fujishiro I, Takeshima Y, et al. A feature-driven approach to locating optimal viewpoints for volume visualization. *Visual Conf IEEE* 2005; pp. 495–502.
5. Ji G and Shen H-W. Dynamic view selection for time-varying volumes. *IEEE T Vis Comput Gr* 2006; 12: 1109–1116.
6. Tao Y, Lin H, Bao H, et al. Structure-aware viewpoint selection for volume visualization. In: *IEEE Pacific 2009, visualization symposium*, Bejing, China; April 20–23; pp. 193–200.
7. Vázquez P-P and Sbert M. Fast adaptive selection of best views. In: *Proceedings of the 2003 international conference on computational science and its applications: part III, ICCSA'03*, Montreal Quebec, Canada; May 18–21 May; pp. 295–305. Berlin, Heidelberg: Springer-Verlag, 2003.
8. Lee CH, Varshney A and Jacobs DW. Mesh saliency. In: *ACM SIGGRAPH 2005 Papers, SIGGRAPH '05*, Los Angeles, CA; July 31st - August 4th; pp. 659–666. New York: ACM, 2005.
9. Harris C and Stephens M. A combined corner and edge detector. In: *Proceedings of the 4th Alvey vision conference*, August 31st-September 2nd; 1988, pp. 147–151; http://www.bmva.org/bmvc/1988/avc-88-000.pdf.
10. Kniss J, Kindlmann G and Hansen C. Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In: *VIS '01: proceedings of the conference on visualization '01*, San Diego, CA, USA; October 21–26 pp. 255–262. Washington, DC: IEEE Computer Society, 2001.
11. Kim HS, Unat D, Baden SB, et al. Interactive data-centric viewpoint selection. *Visual Data Anal* 2012; 8294(5): 829405.
12. Blanz V, Tarr MJ and Blthoff HH. What object attributes determine canonical views? *1999 Perception*; 28: 575–600.
13. Bowyer KW and Dyer CR. Aspect graphs: an introduction and survey of recent results. *Int J Imag Syst Tech* 1990; 2(4): 315–328.
14. Koenderink JJ and Doorn AJ. The singularities of the visual mapping. *Biol Cybern* 1976; 24: 51–59.
15. Koenderink JJ and Doorn AJ. The internal representation of solid shape with respect to vision. *Biol Cybern* 1979; 32: 211–216.
16. Palmer S, Rosch E and Chase P. Canonical perspective and the perception of objects. *Attention Perform* 1981; IX: 131–151.
17. Polonsky O, Patan G, Biasotti S, et al. What is an image? Towards the computation of the "best" view of an object. *Visual Comput* 2005; 21(8–10): 840–847.
18. Laptev I and Lindeberg T. Space-time interest points. *IEEE Comput Soc* 2003; 432–439.
19. Sipiran I and Bustos B. A robust 3d interest points detector based on Harris operator. In *3DOR*. 2010, pp. 7–14.
20. Pearson K, On lines and planes of closest fit to systems of points in space. *Philos Mag* 1901; 2(6): 559–572.
21. Boyd S and Vandenberghe L. *Convex optimization*. New York: Cambridge University Press, 2004.
22. Forsyth DA and Ponce J. *Computer vision: a modern approach*. Prentice Hall, New Jersey, 2002.
23. Unat D, Cai X and Baden SB. Mint: realizing CUDA performance in 3D stencil methods with Annotated C. In: *Proceedings of the international conference on supercomputing, ICS '11*, Tucson, AZ May 31st-June 4th, 2011, pp. 214–224. New York: ACM, 2011.
24. Nickolls J, Buck I, Garland M, et al. Scalable parallel programming with CUDA. *Queue* 2008; 6: 40–53.
25. Unat D, Kim HS, Schulze JP, et al. Auto-optimization of a feature selection algorithm. In: *Proceedings of the 4th workshop on emerging applications and many-core architecture*, San Jose, CA. June 4th 2011.
26. Dagum L and Menon R. OpenMP: an industry standard API for shared-memory programming. *IEEE Comput Sci Eng* 1998; 5(1): 46–55.
27. Roger D, Assarsson U and Holzschuch N. Efficient stream reduction on the GPU. In: *Workshop on general purpose processing on graphics processing units*, Boston, MA, October 2007.
28. Nukada A, Ogata Y, Endo T, et al. Bandwidth intensive 3-d fft kernel for GPUS using CUDA. In: *International conference for high performance computing, networking, storage and analysis, 2008. Austin, Texas, SC 2008*. 2008, pp. 1–11.
29. CULA tools: GPU accelerated linear algebra. http://www.culatools.com/, 2011.
30. Paris S, Kornprobst P and Tumblin J. *Bilateral filtering*. Hanover, MA: Now Publishers Inc., 2009.
31. Paris S and Durand F. A fast approximation of the bilateral filter using a signal processing approach. *Int J Comput Vision* 2009; 81: 24–52.
32. Martone ME, Gupta A, Wong M, et al. A cell-centered database for electron tomographic data. *J Struct Biol* 2002; 138: 145–155.
33. Mikula S, Trotts I, Stone JM, et al. Internet-enabled high-resolution brain mapping and virtual microscopy. *Neuroimage* 35(1): 9–15.
34. Lowe DG. Object recognition from local scale-invariant features. In: *Proceedings of the international conference on computer vision-Volume 2, ICCV '99*, Kerkyra Greece, September 20–27, p.1150. Washington, DC: IEEE Computer Society, 1999.