Interactive Data-Centric Viewpoint Selection

Han Suk Kim[a], Didem Unat[a], Scott B. Baden[a], Jürgen P. Schulze[a]

[a]University of California San Diego, 9500 Gilman Drive, La Jolla, CA, USA

ABSTRACT

We propose a new algorithm for automatic viewpoint selection for volume data sets. While most previous algorithms depend on information theoretic frameworks, our algorithm solely focuses on the data itself without off-line rendering steps, and finds a view direction which shows the data set's features well. The algorithm consists of two main steps: feature selection and viewpoint selection. The feature selection step is an extension of the 2D Harris interest point detection algorithm. This step selects corner and/or high-intensity points as features, which captures the overall structures and local details. The second step, viewpoint selection, takes this set and finds a direction that lays out those points in a way that the variance of projected points is maximized, which can be formulated as a Principal Component Analysis (PCA) problem. The PCA solution guarantees that surfaces with detected corner points are less likely to be degenerative, and it minimizes occlusion between them. Our entire algorithm takes less than a second, which allows it to be integrated into real-time volume rendering applications where users can modify the volume with transfer functions, because the optimized viewpoint depends on the transfer function.

Keywords: Viewpoint selection, Harris interest point detection, Principal component analysis

1. INTRODUCTION

Optimal viewpoint selection is a method which finds a 2D projection of a 3D data set which 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.

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 paper, 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. 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?; and 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 problem.

Our algorithm runs at interactive speed (¡1 second) 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 point, and the optimization problem takes only a small set of points as its input, which avoids actually rendering the volume to analyze the result. We further reduce the computational cost through hardware acceleration by implementing the core parts of our algorithm in 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.

The remainder of this paper is organized as follows: Section 2 reviews previous work on viewpoint selection algorithms. In Section 3, we introduce our feature selection algorithm. Section 4 describes in detail how we compute the best view direction from the features. Section 5 discusses issues in implementing and optimizing our algorithm in order to run in real-time. Section 6 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 7. Finally, we conclude this paper and suggest future work in Section 8.

2. 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.[11–15] The Aspect Graph[13,14] 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.[12] The Canonical View[11,15] is another concept for defining good views. Palmer et al.[15] did experiments to rate the quality of viewpoints, and Blanz et al.[11] 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.[1,7,16] 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 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 the paper. Polonsky et al.[16] 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 which measures the Jensen-Shannon divergence between samples. This allows 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 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 they call 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.

3. FEATURE SELECTION

The goal of our feature selection algorithm is to find all the features in a data set which would help the viewer understand the data. For example, corners of an object and high intensity points are important features which 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*.

3.1 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.

3.1.1 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 both in the X-axis and 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 are 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):

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

where $I(x,y)$ is the opacity 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:

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

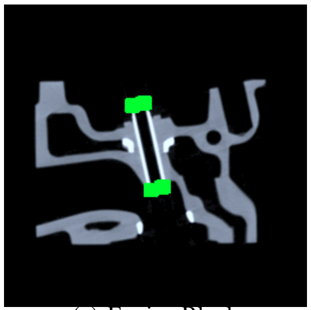
where A, B, and C are a Gaussian convolution of I_x^2, I_y^2, and I_xI_y, respectively:

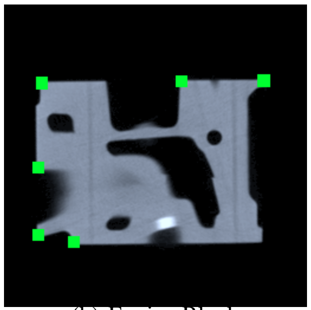
$$\begin{aligned} A &= g \otimes I_x^2 \\ B &= g \otimes I_y^2 \\ C &= g \otimes (I_x I_y) \end{aligned}$$

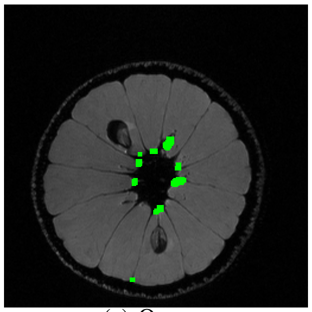
Then, $E(x,y)$ can be written in matrix form:

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

Therefore, the intensity change in E is characterized by the 2x2 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


(a) Engine Block


(b) Engine Block


(c) Orange

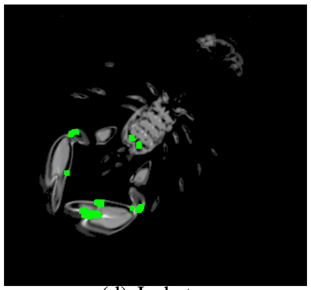

(d) Lobster

Figure 1. Examples of Harris interest points. Results for the Engine Block CT Scan data from General Electric, USA. Figure 1(a) and Figure 1(b) show 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 Figure 1(a), four corners of the cylinder are selected as interest points. Figure 1(b) shows the outermost slice of the engine block and 6 corners are detected as interest corners. The interest points of the orange in Figure 1(c), highlighted in green, include the center of the orange pieces and a small seed. The data is from the Information and Computing Sciences Division, Lawrence Berkeley Laboratory, USA. The lobster data shown in Figure 1(d) is from the VolVis distribution of SUNY Stony Brook, NY, USA. The claws and the joints of the lobster have interest points.

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:

$$R = det(M) - kTrace(M) \tag{4}$$

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.

3.1.2 Extension to 3D

In direct volume visualization, data is stored on a regular 3D grid. Thus, in order to use our approach with such data, we need to extend the algorithm to the 3D spatial domain. Previously, Laptev and Lindeberg[17] extended the algorithm to the spatio-temporal domain, and Sipiran and Bustos[18] 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:

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

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

$$I(x+u,y+v,z+w) \approx I(x,y,z) + xI_x + yI_y + zI_z + O(x^2,y^2,z^2) \tag{6}$$

Following the same logic as in Section 3.1.1 yields the matrix M to be defined as follows:

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

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 5.

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.

3.2 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 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 4 because X impacts the performance of the solver.

Figure 1(a) and Figure 1(b) show 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 CT Scan data set has two cylinders in the block, and Figure 1(a) shows that the top and bottom of the cylinders are selected as interest points. Figure 1(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 1(c) and 1(d) show two other data sets. An MRI scan of an orange is from the Information and Computing Sciences Division, Lawrence Berkeley Laboratory, USA. Figure 1(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 is a CT scan of a lobster. The data is from the VolVis distribution of SUNY Stony Brook, NY, USA. The interest points include two claws and joints as highlighted in Figure 1(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.

4. VIEW SELECTION

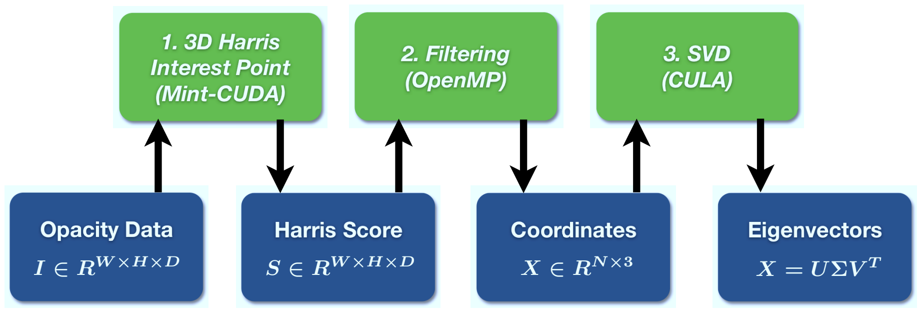
The previous steps generate a set of feature points. Those points are clustered around the corners of high opacity region. Those points are important in viewpoint selection 1) because we do not want the corners of an object to overlap each other and 2) because we want to avoid the salient features, which users highlight with transfer function manipulation, to occlude 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 has outstanding features, which are captured in the Harris interest points, and if those features has minimum occlusion, we can still see important objects in the data. Because the Harris interest point algorithm favors high opacity values, high intensity points, i.e., 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, i.e. 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 Principal Component Analysis (PCA).[19]

4.1 Principal Component Analysis

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:

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


1. 3D Harris Interest Point (Mint-CUDA)
2. Filtering (OpenMP)
3. SVD (CULA)
Opacity Data $I \in R^{W \times H \times D}$
Harris Score $S \in R^{W \times H \times D}$
Coordinates $X \in R^{N \times 3}$
Eigenvectors $X = U \Sigma V^T$
Figure 2. 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.

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

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

The detail of how to find the solution for the optimization problem in Equation 8 can be found in the literature,[19] but the solution is derived from the Lagrange dual function[20] and is the eigenvector with the largest eigenvalue of the covariance matrix of X. That is, the best viewpoint is the solution of $X^T Xu = \lambda u$. 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. 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[21] for the actual computation. SVD decomposes the input matrix X into $U\Sigma V^T$ and the right singular vector V's are the eigenvectors of $X^T X$. Suppose SVD produced V as $[v_1|v_2|v_3]$, then the 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 which 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, i.e., 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.

5. IMPLEMENTATION

We implemented the algorithm described in Section 3 and Section 4. There are three main kernels to compute the best viewpoint: 1) 3D Harris Corner Point, 2) Filtering, and 3) PCA. Figure 2 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 3.1) 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 multi-core 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[22] to automatically generate the CUDA (Compute Unified Device Architecture[23]) 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 CPU implementation, delivering a 45x speedup. More details about the parallelization and the implementation on GPU are described in Unat et al.[24]

The second kernel is filtering (Section 3.2). The input to this kernel is the Harris score computed in the previous step and the output is a matrix X, i.e., 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.[25] 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.

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 CULA library,[26] 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.

6. EVALUATION

In this section, we compare our algorithm with two previous algorithms in terms of running time and visual quality. We measured the running time under the same conditions and conducted a user study to rate the quality of images.

6.1 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 probability distribution function (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, i.e., it evaluates as many points as possible and chooses the best one, i.e., 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 use two recent approaches, one by Ji and Shen[5] and the other by Tao et al.[6] Ji and Shen use opacity to find which image is better than other images. If there are many pixels that have high opacity in an image, they consider the image to be important. Therefore, they render the volume with opacity data and measure the entropy value of the opacity distribution. The rendered image that has the highest entropy value is the best view. Although they define color and curvature information to measure the quality of a view, it requires multiple rendering processes to get entropy values from each measure. This significantly slows the running time, but we were able to get very good results with only opacity, which we discuss in detail in Section 6.3.

The algorithm by Tao et al. is based on the same framework but they measure *shape descriptor* and *detail descriptor*. Shape descriptor measures the information that represents the overall structure of a volume data set, while detail descriptor measures the small details of the data. The former is computed from the data created by bilateral filtering.[27] The algorithm filters the original volume, which smooths 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. They also argue that the 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 does not contain local features, we decided to put 0.8 weight on the shape descriptor and 0.2 weight on the detail descriptor. Both descriptors require gradient computation, and the gradient for each fragment is computed in shader.

6.2 Performance Comparison

Both approaches (Ji and Shen and Tao et al.) were implemented in C++ with OpenGL. The volume rendering algorithm uses 3D texture 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 used GLSL programs. The viewport was set to 512×512 pixels. The best viewpoint is selected from 500 viewpoints randomly selected on the viewing sphere. The center of each volume is located at the origin and the view direction is set to look at the origin. For our algorithm, as mentioned in Section 5, we used the Mint translator, 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 linbrary. We used version 3.2 of the CUDA Toolkit.

For the sake of fair comparison, we ran both approaches, as well as ours, on the same machine. The graphics card and CPU in the machine are NVIDIA GeForce GTX 285 and Intel Core i7 2.93 GHz, respectively. We measured the running time 10 times for each case and report the minimum in Table 1. During the measurement, we excluded the time for bilateral filtering in the algorithm by Tao et al., which takes more than tens of seconds for most data sets. The computation needs to be included in the comparison, but there are many algorithms proposed[28] that are much more efficient than our naive implementation. Furthermore, if one can implement the algorithm in CUDA, it is possible to run in under a second. Therefore, we decided to exclude the timing for the computation, although whenever we re-define the opacity with the transfer function, it is necessary to re-compute the filtering.

Table 1 shows the running time of the three algorithms. Our approach clearly outperforms Ji and Shen and Tao et al. On average, our approach is 27.16 times faster than the algorithm by Ji and Shen, and 38.66 times faster than the algorithm by Tao et al. The more important aspect of the running times is that our algorithm runs in under a second, which enables us to run our algorithm interactively in any volume rendering systems. More than 5 second of running time by previous approaches is not suitable for running multiple times in the middle of volume interaction. We will discuss further what kind of benefit we can obtain in Section 7.1.

6.3 Result Comparison

Figure 3 shows the results from our algorithm and previous approaches. In the first row of Figure 3, we compare the best view for the CT-Knee data. Figure 3(a) and 3(d) render the data from the back and the front, respectively. One can easily see that the data shows the symmetric shape of the legs and that it renders the knee of the leg by finding the shapes of bones. Figures 3(c) shows two legs but the legs slightly overlap each other because the viewpoint is from the side, preventing the viewer from distinguishing the two legs. The second row of Figure 3 shows the Foot CT scan data. The viewpoints of Figure 3(e) and 3(g) are from the top, both of which show the data effectively. However, the viewpoint for Figure 3(h) is slightly tilted. One can still understand the shape of the foot in Figure 3(h). The third row of Figure 3 renders the Engine CT scan data. The result of our approach (Figure 3(i)) is slightly less descriptive than Figures 3(k) and 3(l). However, in our approach, the tip of cylinder is clearly shown while the other two appraoches failed to do. For the Tooth data, most algorithms show similar results, as shown in the fourth row of Figure 3. The viewpoint from the side generates a large number of non-zero 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.), leading the side view to the best viewpoint. The Lobster data shown in the last row of Figure 3 favors our algorithm. Our result (Figure 3(q)) shows the lobster from the top, revealing all the details from the claws to the tail. Figure 3(t) shows almost identical results where the viewpoint is set slightly tilted to the side. The result from Ji and Shen (Figure 3(s)) shows the object from the side. This result is because the algorithm favors an image that has many non-zero pixels. Due to the body and the claw, the projected area of the lobster is maximized when projected to the side.

Data Set	**Data Size**	**Our Approach**		**Ji and Shen**		**Tao et al.**	
		Running Time	Rating	Running Time	Rating	Running Time	Rating
CT-Knee	$379 \times 229 \times 305$	0.8128s	4.30	6.8281s	4.30	18.2919s	5.15
Foot	$256 \times 256 \times 256$	0.5906s	5.20	8.4182s	5.90	14.0245s	4.85
Engine	$256 \times 256 \times 256$	0.5036s	4.60	8.6340s	5.60	13.9101s	6.00
Lobster	$301 \times 324 \times 56$	0.2619s	5.27	5.6118s	2.23	8.2253s	5.82
Tooth	$94 \times 103 \times 161$	0.1345s	4.42	5.0289s	5.60	6.5794s	5.40

Table 1. Performance comparison for various volume data sets. 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 under a second in all data sets, which makes it feasible to be used along with transfer function exploration. The average ratings from our user study show that our algorithm achieves comparable image qualities to the previous approaches.

6.4 User Study

In order to confirm that our approach produces comparable results to the previous algorithms, we conducted a user study. We created an on-line questionnaire and sent it to graduate students and research staff working in computer graphics and scientific visualization. We received 21 fully answered questionnaires back. The participants were asked to rate each image in Figure 3 on a scale from 1 (worst viewpoint) to 4 (neutral) to 7 (best viewpoint).

The average ratings are listed in Table 1, 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 3, are consisitently very low (between 1.95 and 2.80).

7. DISCUSSION

7.1 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.

Secondly, 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.[29,30] 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.

7.2 Limitation

Despite of 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 iso-surface 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]

8. 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, e.g. SIFT.[31] 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.

ACKNOWLEDGEMENT

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., and Heidrich, W., "Viewpoint selection using viewpoint entropy," in [*Proceedings of the Vision Modeling and Visualization Conference 2001*], *VMV '01*, 273–280, Aka GmbH (2001).

[2] Bordoloi, U. D. and Shen, H.-W., "View selection for volume rendering," *Visualization Conference, IEEE* **0**, 62 (2005).

[3] Sbert, M., Plemenos, D., Feixas, M., and González, F., "Viewpoint quality: Measures and applications," in [*Eurographics Workshop on Computational Aesthetics in Graphics, Visualization and Imaging*], Neumann, L., Sbert, M., Gooch, B., and Purgathofer, W., eds., 185–192 (2005).

[4] Takahashi, S., Fujishiro, I., Takeshima, Y., and Nishita, T., "A feature-driven approach to locating optimal viewpoints for volume visualization," *Visualization Conference, IEEE* **0**, 63 (2005).

[5] Ji, G. and Shen, H.-W., "Dynamic view selection for time-varying volumes," *IEEE Transactions on Visualization and Computer Graphics* **12**, 1109–1116 (2006).

[6] Tao, Y., Lin, H., Bao, H., Dong, F., and Clapworthy, G., "Structure-aware viewpoint selection for volume visualization," *Visualization Symposium, IEEE Pacific* **0**, 193–200 (2009).

[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*, 295–305, Springer-Verlag, Berlin, Heidelberg (2003).

[8] Lee, C. H., Varshney, A., and Jacobs, D. W., "Mesh saliency," in [*ACM SIGGRAPH 2005 Papers*], *SIGGRAPH '05*, 659–666, ACM, New York, NY, USA (2005).

[9] Harris, C. and Stephens, M., "A combined corner and edge detector," in [*Proceedings of the 4th Alvey Vision Conference*], 147–151 (1988).

[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*], 255–262, IEEE Computer Society, Washington, DC, USA (2001).

[11] Blanz, V., Tarr, M. J., and Blthoff, H. H., "What object attributes determine canonical views?," (1998).

[12] Bowyer, K. W. and Dyer, C. R., "Aspect graphs: An introduction and survey of recent results," *International Journal of Imaging Systems and Technology* **2**(4), 315–328 (1990).

[13] Koenderink, J. J. and Doorn, A. J., "The singularities of the visual mapping," *Biological Cybernetics* **24**, 51–59 (1976). 10.1007/BF00365595.

[14] Koenderink, J. J. and Doorn, A. J., "The internal representation of solid shape with respect to vision," *Biological Cybernetics* **32**, 211–216 (1979). 10.1007/BF00337644.

[15] Palmer, S., Rosch, E., and Chase, P., "Canonical perspective and the perception of objects," *Attention and Performance IX* , 131–151 (1981).

[16] Polonsky, O., Patan, G., Biasotti, S., Gotsman, C., and Spagnuolo, M., "What is an image?: Towards the computation of the "best" view of an object," *The Visual Computer (Proc. Pacific Graphics)* **21**(8-10), 840–847 (2005).

[17] Laptev, I. and Lindeberg, T., "Space-time interest points," in [*ICCV*], 432–439, IEEE Computer Society (2003).

[18] Sipiran, I. and Bustos, B., "A robust 3d interest points detector based on harris operator," in [*3DOR*], 7–14 (2010).

[19] Pearson, K., "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine* **2**(6), 559–572 (1901).

[20] Boyd, S. and Vandenberghe, L., [*Convex Optimization*], Cambridge University Press, New York, NY, USA (2004).

[21] Forsyth, D. A. and Ponce, J., [*Computer Vision: A Modern Approach*], Prentice Hall Professional Technical Reference (2002).

[22] Unat, D., Cai, X., and Baden, S. B., "Mint: realizing cuda performance in 3d stencil methods with annotated c," in [*Proceedings of the international conference on Supercomputing*], *ICS '11*, 214–224, ACM, New York, NY, USA (2011).

[23] Nickolls, J., Buck, I., Garland, M., and Skadron, K., "Scalable parallel programming with CUDA," *Queue* **6**, 40–53 (March 2008).

[24] Unat, D., Kim, H. S., Schulze, J. P., and Baden, S. B., "Auto-optimization of a feature selection algorithm," in [*Proceedings of the 4th Workshop on Emerging Applications and Many-core Architecture*], (June 2011).

[25] Dagum, L. and Menon, R., "OpenMP: an industry standard API for shared-memory programming," *Computational Science Engineering, IEEE* **5**(1), 46 –55 (1998).

[26] CULA Tools: GPU Accelerated Linear Algebra, "`http://www.culatools.com/`," (2011).

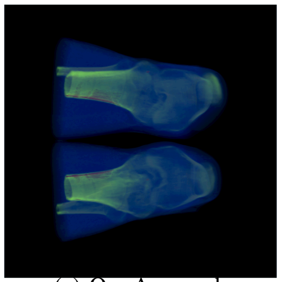
[27] Paris, S., Kornprobst, P., and Tumblin, J., [*Bilateral Filtering*], Now Publishers Inc., Hanover, MA, USA (2009).

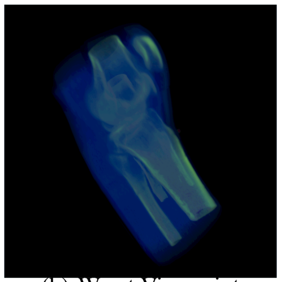
[28] Paris, S. and Durand, F., "A fast approximation of the bilateral filter using a signal processing approach," *Int. J. Comput. Vision* **81**, 24–52 (January 2009).

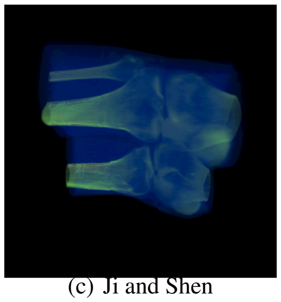
[29] Martone, M. E., Gupta, A., Wong, M., Qian, X., Sosinsky, G., Ludäscher, B., and Ellisman, M. H., "A cell-centered database for electron tomographic data," *J Struct Biol* **138**, 145–55 (Jan 2002).

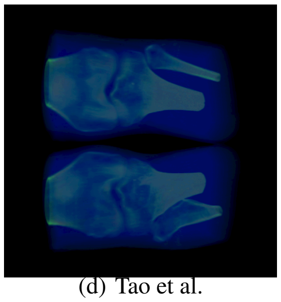
[30] Mikula, S., Trotts, I., Stone, J. M., and Jones, E. G., "Internet-enabled high-resolution brain mapping and virtual microscopy," *Neuroimage* **35**(1), 9–15.

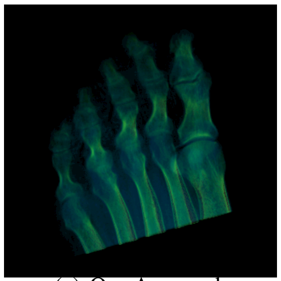
[31] Lowe, D. G., "Object recognition from local scale-invariant features," in [*Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*], *ICCV '99*, 1150–, IEEE Computer Society, Washington, DC, USA (1999).

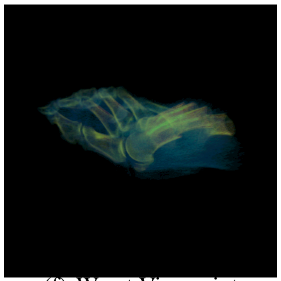

(a) Our Approach

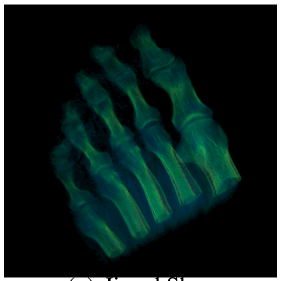

(b) Worst Viewpoint

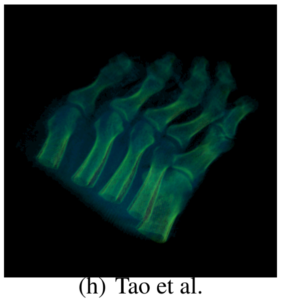

(c) Ji and Shen

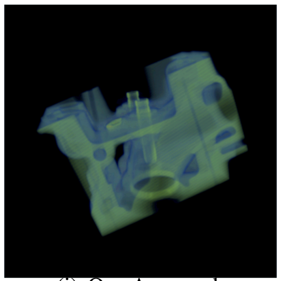

(d) Tao et al.

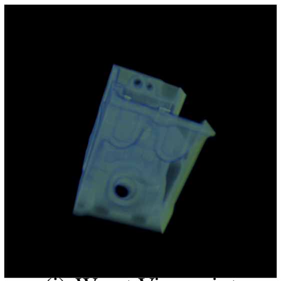

(e) Our Approach

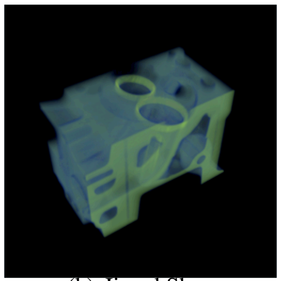

(f) Worst Viewpoint

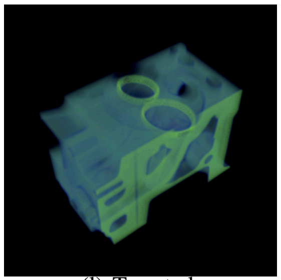

(g) Ji and Shen

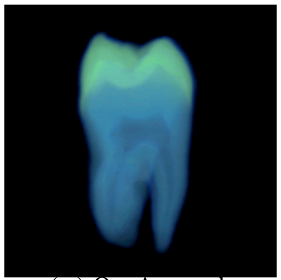

(h) Tao et al.

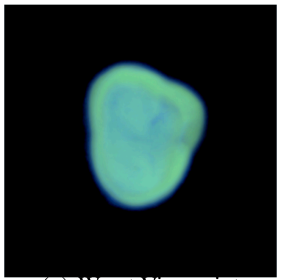

(i) Our Approach

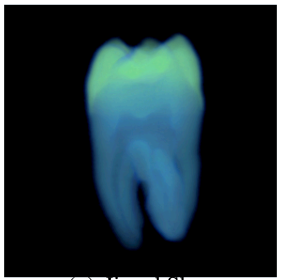

(j) Worst Viewpoint

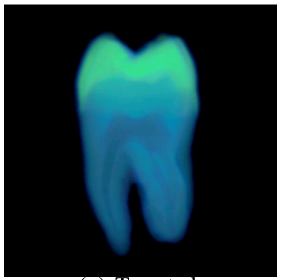

(k) Ji and Shen

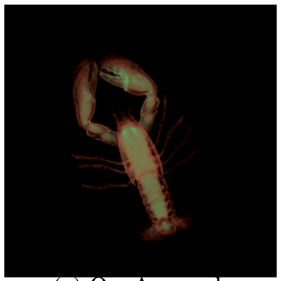

(l) Tao et al.

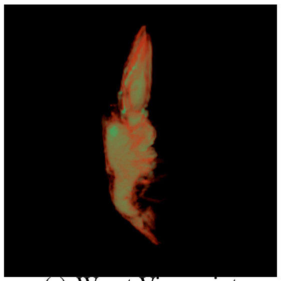

(m) Our Approach

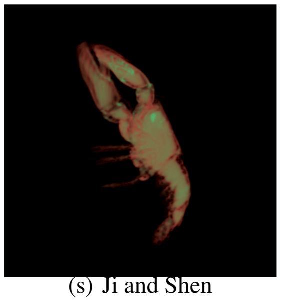

(n) Worst Viewpoint


(o) Ji and Shen


(p) Tao et al.


(q) Our Approach


(r) Worst Viewpoint


(s) Ji and Shen

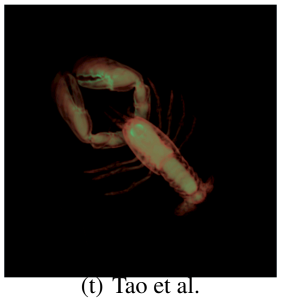

(t) Tao et al.

Figure 3. Comparison between other approaches.