Automatic Transfer Function Design for Volumetric Data Visualization using Clustering on LH Space

Binh P. Nguyen · Wei-Liang Tay · Chee-Kong Chui · Sim-Heng Ong

Abstract A major difficulty in volumetric data rendering is finding good transfer functions to display the regions of interest within a volume. We introduce a method that automatically detects boundaries, improves the segmentation of each boundary, and finally generates a transfer function that minimizes occlusion of the boundaries. This method applies mean shift clustering on LH space, and then generates polygons which approximate the resulting clusters on the LH histogram. Each polygon represents a boundary surface of the volume and can be easily modified to obtain the desired visualization. A post-processing step using region growing further improves the segmentation result. The transfer function is generated automatically based on the size and location of the boundaries to minimize occlusions. The effectiveness of the proposed method is demonstrated in our experiments.

Keywords Transfer function design · Volume rendering · LH histogram · Mean-shift clustering

1 Introduction

Direct volume rendering is a powerful technique in scientific visualization, especially in diagnostic imaging where images are volumetric data generated by CT and MR scanners. Volume rendering is a method that could overcome the lack of spatial perception in the traditional 2D slice-by-slice viewing. However, the efficiency

Binh P. Nguyen (contact author) · W.-L. Tay · S.-H. Ong
Department of Electrical and Computer Engineering, National University of Singapore, Singapore.
{phubinh, tayweiliang, eleongsh}@nus.edu.sg

C.-K. Chui
Department of Mechanical Engineering, National University of Singapore, Singapore.
mpecck@nus.edu.sg

of volume rendering strongly depends on the transfer function (TF), which is a mapping from data properties (e.g., scalar value and gradient magnitude) to optical properties (e.g., opacity and color). Although a good TF can reveal the important structures in the data, finding an appropriate mapping that yields the desired visual appearance is not a trivial task. It requires an understanding of the TF domain and manually tweaking parameters on the part of the user.

Various techniques have been developed to facilitate the design of TFs. Some of these can automatically determine an appropriate TF, or at least propose an initial suggestion that can be modified by the user. Several methods generate and apply different TFs, and then analyze the rendered images to derive an optimal set of parameters [20, 6, 12]. Although these image-driven solutions can work well, their drawbacks, including the dependency on image-related parameters (e.g., viewing direction and image resolution), limit their application in practice. The majority of other methods generally are data-driven techniques that analyze the volume data itself instead of the rendered images. Due to the limitation of 1-D TFs when rendering data values associated with multiple materials, multidimensional TFs are often used. However, if the number of dimensions is larger than two, there are difficulties in the design of user interfaces. Even if the TF feature domain is known, manipulation in multidimensional feature space is very difficult for the user. Existing methods use dimension reduction and feature extraction [21] and/or clustering [27] techniques in order to facilitate the TF design since interaction with low-dimension data or clusters seems to be more intuitive than manipulating the TFs directly. There are other methods that allow the user to select interesting features directly from the data domain, followed by region growing [9, 26] or using a neural network [28, 29] to generate the TFs. Unfortunately,

the simplification of the TF space using machine learning algorithms in these approaches tends to restrict the variety of available TFs and hence reduce the user's control over the resulting renderings.

Some recent works propose the use of LH space in TF design [30, 31]. These studies show that the LH histogram, a representation of LH space, can represent boundary information in a more compact and robust manner than the commonly used scalar value and its derivatives. Moreover, LH space can provide an unambiguous classification of boundaries with distinct LH values. Therefore, applying a clustering technique to the LH space is more suitable than the traditional intensity-gradient space in which boundaries appear as arches that tend to overlap.

The method presented in this paper involves application of a clustering technique that is particularly efficient in LH space. The technique can be easily tuned since the tuning can be achieved using only one parameter. We also propose a number of automatic operations that greatly reduce the conventional trial-and-error process for designing TFs. One of these operations is the approximation of the resulting clusters in LH space in polygon form so that the user can edit and manipulate the TFs more easily. Furthermore, a data-driven post-processing step that takes into account the spatial information in the dataset improves the quality of the TFs. In summary, the proposed innovations significantly reduce the time and effort required to obtain good TFs for volume rendering and enable visualizations with quality approaching that of existing methods to be automatically generated.

2 Related Work

Interactive TF design has been addressed through many different studies. Due to the ease of implementation and parallelization, most approaches employ derivatives to design TFs. Kindlmann et al. [10] used the first derivative (i.e., gradient) as an attribute to generate multi-dimensional TFs. In the 2-D TF domain which incorporates the intensity and gradient magnitude, material boundaries can be interpreted as arches. Thus, they can be selected and visualized by manipulating certain TF widgets to approximate the arches. However, these arches often overlap, which prevents proper isolation of a material from others. One possible approach to overcome this drawback is to include the second directional derivative along with the gradient direction [13, 14]. Nevertheless, these methods cannot fully solve the blur effect in the intensity-derivative histogram which is caused by noise. Lum et al. [15] used the two intensity values on both sides of the border

to set up a TF with the assumption that the width of the border represented by the distance between these two sample positions varies with the amount of blur in the volume. Šereda et al. [30] proposed another method to represent boundaries by searching for low and high intensity values in both the negative and positive gradient directions of the voxels in a boundary. The representation of those low and high values in a 2-D plane is called the *LH histogram*. An important advantage of LH histograms over the 2D intensity-gradient magnitude TF is that boundaries appear as blobs rather than arches. Blobs are easier to parameterize for clustering and are less likely to overlap in complicated datasets than arches; thus LH histograms allow for boundaries to be more easily separated either manually or automatically through clustering. Another advantage is that LH histograms have greater robustness to noise, bias and partial volume effects than intensity-gradient magnitude histograms. Recently, a semi-automatic generation of LH TFs using a fast generation of LH values has been introduced by Praßni et al. [23].

Some methods use curvature-based information to design TFs. Bajaj et al. [2] introduced the *contour spectrum* which consists of a variety of scalar data and contour attributes to describe isosurfaces. They also provided an interactive user inferface for the selection of relevant isovalues. Another method for isosurface visualization that uses a cumulative Laplacian-weighted intensity histogram was proposed by Pekar et al. [22]. Inspired by the work of Hladuvka et al. [7], Kindlmann et al. [11] proposed a methodology for computing high quality curvature measurements, and the application of curvature-based TFs in volume rendering.

Since derivatives or curvature measurements only represent local information, many methods have attempted to extend the traditional 1-D or 2-D histograms with spatial information. Lundström et al. [16, 17] introduced *local histograms*, which utilize the distribution of intensity values in a given neighborhood to differentiate between different tissues. They also introduced another method to incorporate spatial coherence into an enhanced histogram, the α-histogram [18]. The local histograms of disjoint local regions are calculated, and then raised to the power of $\alpha > 1$ before being summed and normalized to produce the global histogram. Thus, spatially concentrated value ranges are amplified to enlarge peaks corresponding to different materials in the global histogram. Röttger el al. [24] also applied the idea of using spatial features of the dataset for TF design. They compute the mean and variance values of all voxels belonging to one single bin in the 2-D histogram, and then use these measures with a maximum feature radius to classify the histogram. Tappenbeck et al. [25]

employed a distance-based TF, which allows the optical properties of structures to be modified based on their distance to a reference structure.

Recently, *feature size* has been used in several approaches as a parameter to design TFs for separating structures with similar intensity values. Correa et al. [3] used scale fields for continuous representation of feature size then assigned to each voxel an additional parameter depending on the local scale of the feature containing the voxel. Size-based TFs then map the opacity and color based on the relative size of features. Hadwiger et al. [5] used region growing to derive feature size information. Instead of using region growing, in Wesarg et al. [32, 33], information of structure size is estimated by searching voxels with intensity satisfying an user specified tolerance value along 26 directions connecting each voxel with its neighbors. The image generated from the structure size data is used as the second property of a *structure size enhanced* (SSE) histogram. In our method, we also use the size of regions, estimated using a new method, to automatically assign visual parameters to minimize occlusions.

In addition to finding new presentations of features, algorithms to separate different regions in the feature domain have been investigated as well. Tzeng et al. [27] presented a method to create TFs based on material classes extracted from the cluster space using the ISO-DATA technique. Šereda et al. [31] applied hierarchical clustering to LH space to group voxels based on their LH values. Maciejewski et al. [19] used non-parametric clustering to extract patterns from TF feature space and guide the generation of TFs. In our work, we also apply a non-parametric clustering method in LH space. Our method is robust and does not require the user to input the number of clusters or modify the cluster hierarchy. It is thus more suitable for non-experts.

3 Transfer Function Design Based on LH Histogram

Our method considers volumetric data that consists of multiple boundaries, each of which is represented by a cluster in the LH histogram. The visual parameters of color and opacity are assigned to the voxels in each cluster. The algorithm can operate in ether manual, automatic, or semi-automatic modes. Figure 1 presents an overview of our method.

3.1 Pre-processing

In all the operating modes, the gradient vector and LH values corresponding to each voxel are first computed
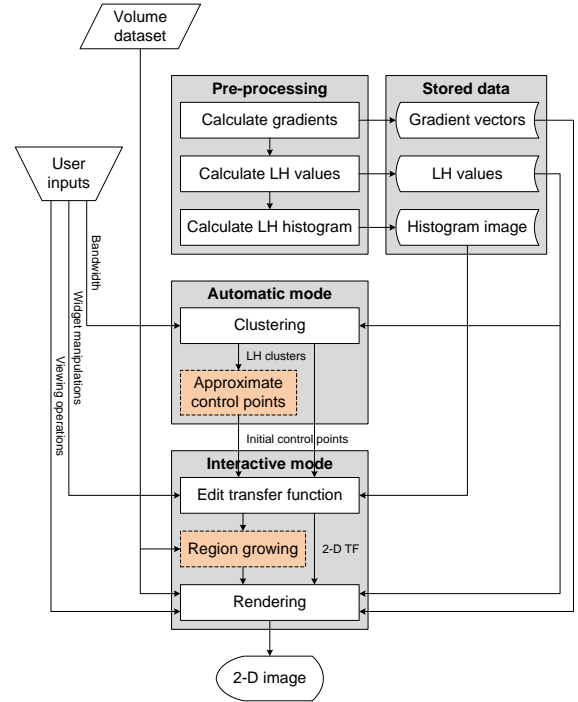

Fig. 1 Overview of the method. Dotted rectangles represent optional processes for the semi-automatic mode.

in a pre-processing step. To calculate the gradients, the method proposed by Hong et al. [8] is used to determine a second-degree polynominal function to approximate the density function in a local neighborhood. The coefficients of the polynomial function can be solved by minimizing the error of the approximation. Once these coefficients are known, the first partial derivatives, i.e. the gradient, and the second directional derivatives corresponding to each voxel can be determined from its intensity and its position. The advantages of this approximation method are: (1) the difference between the pixel spacing and the spacing between slices can be accounted for; (2) no computationally expensive interpolation method is needed to estimate the gradient vector of an arbitrary sampling point between voxels; (3) this method is robust to noise since it does not interpolate the curve passing through all the given data points.

Based on the computed gradients, the lower (L) intensity and higher (H) intensity values of each voxel can be determined by tracking the boundary path using gradient integration in both directions. We use Heun's method, which is a modified Euler's method, to integrate the gradient field:

$$u_{i+1} = u_i + \frac{1}{2}d\left(\nabla f\left(u_i\right) + \nabla f\left(u_i + d\nabla f\left(u_i\right)\right)\right), \quad (1)$$

where u_i and u_{i+1} are positions of the current and the next sampling voxels, respectively, ∇f denotes normalized gradient vector when tracking H or the inverse one
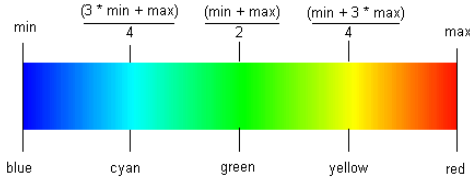
Fig. 2 Cold-to-hot color ramp.

when tracking L, and d is the step size of the integration. In our experiments, a step size of one voxel is chosen as a good balance between accuracy and computation speed. The integration stops when a local extremum or an inflexion point is reached. In order to emphasize voxels on the boundary or near the middle layer between two materrials, each pair [L, H] is weighted by a factor w when being accumulated to create the LH histogram. The weight w is determined from

$$w = 1 - \frac{|d_L - d_H|}{d_L + d_H}, \tag{2}$$

where d_L and d_H are the accumulated distances along the boundary path from the current voxel to the sampling voxels corresponding to L and H, respectively. Preliminary experiments suggest that the new interpolation method based on approximation is superior to conventional gradient estimation methods. Combined with the distance weighting factor, it can reduce the noise in the LH histogram and improve the resulting visual quality.

The LH histogram in our method is represented as an image of $N \times N$ pixels. To balance between the memory needed and the visual quality, N is chosen as 512 pixels. The image is constructed by determining the correct bin for each [L, H] pair, scaling the sum of all corresponding weight factors taking the logarithm, and then mapping the resulting value to a color band, e.g. the cold-to-hot spectrum (Figure 2). At the end of this pre-processing step, all the gradient vectors, the LH values, and the histogram image are stored in an intermediate data file for further processing.

3.2 TF Design Modes

Our rendering algorithm operate in one of two modes. In the manual mode, a selection of specific materials and boundaries can be made using certain TF widgets, e.g. polygons. The user can create a polygon to approximate a region on the histogram by using the mouse to click onto desired positions to add control points of the polygon. Through these points, the user can select, remove, change the shape, and assign optical properties to each polygon easily. Based on the properties of all polygons, a 2-D TF is generated and transferred to the

renderer to produce the final image. Optionally, the user can apply a post-processing step using region growing to improve the visualization result. The region growing process incorporates the spatial information of all voxels corresponding to each polygon in the histogram in order to fill up discontinuities, remove isolated small regions, and merge two overlapping clusters. This leads to an enhancement of the rendering result.

Alternatively, the automatic mode can be chosen. Mean shift clustering [4] is first applied to segment the LH space into multiple clusters. All clusters are then automatically assigned colors and opacities based on the total number and the distribution of voxels in each cluster so that this minimizes the occlusion and maximizes the discrimination of the boundaries. If the rendering result is not satisfactory, the user can switch to the semi-automatic mode to reassign optical properties to each cluster or to change the shape of clusters by editing approximated boundary polygons. This processing step is not required in the automatic mode, but boundary polygons are more easily visualized and edited by the user as compared to clusters of points. The renderer image also can be improved by using region growing based post-processing. Details of the automatic and semi-automatic mode are presented in the next section.

4 Clustering-based Method for Automatic Transfer Function Design

Our method automatically generates a suitable initial TF which can be easily modified by the user. Each boundary in the volume is grouped as a separate cluster for the assignment of visual parameters. Mean shift clustering is used to divide the LH histogram into clusters. Bounding polygons are generated to enclose each cluster. These bounding polygons can be easily modified by the user using polygon or vertex operations. A post-processing operation of region growing is then used to enhance the quality of volume rendering. Finally, the TFs are automatically computed and assigned for each cluster. These TFs for each cluster can also be individually modified according to user preference.

4.1 Mean Shift Clustering in LH Space

Mean shift clustering is a non-parametric feature-space analysis technique that seeks the modes of the given sample space. As a non-parametric method, mean shift clustering does not assume any specific distribution of the data and is more robust for general data. Also,

mean shift clustering relies only on the bandwidth parameter B_w and thus is intuitive for the user to tune as it correlates to the sensitivity of the clustering process. From our experiments, a good B_w often lies between 3%-12% of the maximum LH value, $max_{LH} = max(max_L, max_H)$. We apply mean shift clustering on the LH space to divide the LH histogram into multiple clusters. To speed up and to reduce the memory requirements for mean-shift clustering, mean shift clustering is computed over discrete values in the LH histogram (since all voxels can only have discrete LH values, and since the LH histogram is relatively sparse) rather than over all the points in the volume. Mean shift clustering will be equivalent as long as each LH point is weighted by its voxel frequency (the number of occurences of a particular LH value in the voxel volume) during the mean computation step. The procedure of mean shift clustering is summarized in the following algorithm:

1. Set the window bandwidth B_w.
2. For a point in the LH histogram, find all points that have LH values within the bandwidth B_w.
3. Find the mean μ_n of the set of neighboring points, with each point weighted by its voxel frequency.
4. Shift the window center to the new mean, and continue steps 2-4 until convergence. A cluster is deemed to have converged if the distance between successive means is less than ρB_w where ρ is a threshold which is preset as 0.001 in our experiments.
5. Repeat steps 2-4 for each point in the LH histogram.
6. Points that converge to the same modes (the converged cluster mean) are grouped as a single cluster, and clusters that have modes within $B_w/2$ of each other are also grouped as one cluster.

4.2 Cluster Bounding Polygons

While mean shift clustering automatically assigns labels to each voxel in the volume, the results may not be immediately satisfactory to the user. No automatic method can simultaneously satisfy the requirements of all users, since different users have different visualization requirements and regions of interest. Minor adjustments made by the user will improve the quality and relevance of the visualization. To facilitate easy modification of the automatically extracted clusters, the voxel cluster labels are used to generate a set of cluster-bounding polygons. The advantage of cluster polygons is that they are easy to manipulate and modify via polygon and vertex operations. Entire clusters or individual vertices can thus be edited on the LH histogram.

Ideally, each cluster polygon should only contain all voxels assigned to that cluster, but this requires
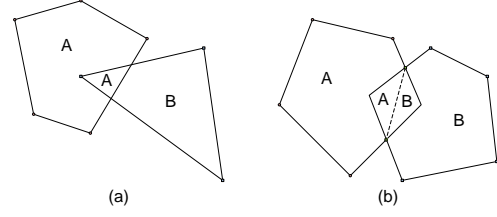

Fig. 3 Two demonstrations of the overlap disambiguation scheme.

computing concave bounding polygons which is computationally expensive. Furthermore, concave polygons tend to be more complicated than convex polygons. To simplify the computation we assume that the bounding polygons are convex polygons. Then, the bounding polygon can be computed in $\Omega(n \log(n))$ time by fast convex hull algorithms such as Andrew's monotone chain algorithm [1]. To resolve overlaps between bounding polygons we perform collision detection for each pair of polygons. For each overlap, there are two intersections. A dividing line is drawn between the two intersections and each partitioned area is assigned to the cluster it is nearest to.

The regions along the main diagonal of the LH histogram belong to voxels lying within the same material, i.e. material not lying on the material interfaces[30]. These clusters may not be important for visualization and can be discarded or rendered with a low opacity value. After the cluster bounding polygons are generated, a simple check is rendered to detect and discard such clusters. All polygons with at least one vertex within a diagonal window of the main diagonal of the LH histogram are treated as clusters of non-boundary material. The diagonal window is experimentally defined to have a width of 2% of the range of LH values. The procedure for computing the cluster bounding polygons is summarized in the following algorithm:

1. For each cluster C_i obtained from the mean shift algorithm, obtain the set of points P_i and compute a convex hull H_i containing all the points in P_i.
2. Construct a convex polygon H_{diag} using the following 6 coordinates: $[0, 0]$, $[0, 0.01 \times max_H]$, $[0.99 \times max_L, max_H]$, $[max_L, max_H]$, $[max_L, 0.99 \times max_H]$, $[0.01 \times max_L, max_H]$, where max_L and max_H are the maximum values in the LH histogram. For each convex hull H_i, if any vertex in H_i lies in H_{diag}, the cluster C_i is treated as a non-boundary cluster and is removed or rendered with low opacity.
3. For each pair of remaining convex hulls H_a and H_b, compute the intersection, if any, between each combination of hull segments. If there are intersections denote them as I_a and I_b. Add both I_a and I_b to
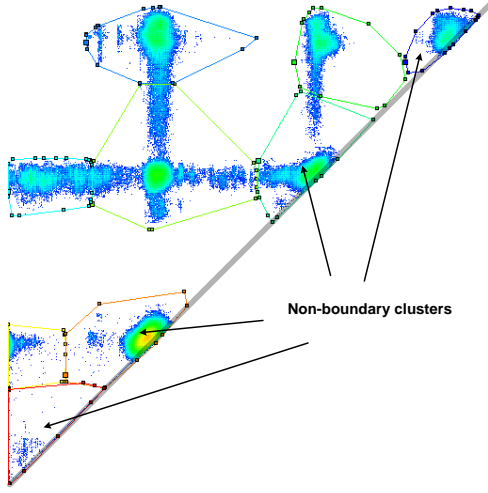
Fig. 4 Demonstration of non-boundary cluster removal on the Tooth dataset.

both hulls H_a and H_b, and remove all hull points interior to the line segment created by H_a and H_b.

4.3 Region Growing

The results of the mean shift clustering are sufficient for simple datasets, but for the more complicated datasets that are typical of medical imaging applications, additional information is needed to have a sufficient image quality. After mean shift clustering, each cluster is further passed through a region growing algorithm. The region growing algorithm is a means of incorporating spatial information to improve the visualization results.

In earlier work by Huang and Ma [9] on using region growing for volume visualization, a number of prior information and parameters, such as the initial seed points and the weighting factors for the cost function, must be provided to the region growing algorithm. In our approach, manual tuning of the region growing parameters is not necessary as the parameters and seed points will be assigned automatically based on the clusters obtained earlier during mean shift clustering.

For each cluster previously extracted after mean shift clustering (and after manual user adjustment), the cluster voxels are used as the initial seed points. The standard deviation of the LH values of the cluster voxels are used to set the similarity tolerance of the region growing algorithm.

After each cluster has been passed through the region growing algorithm, the separate volumes must be merged into a single volume. If there are voxels belonging to more than one cluster, we simply merge all overlapping clusters. In future work, other criteria may be added to restrict merging to only cases where there

is significant overlap between clusters. The algorithm for region growing enhancement is given below:

1. For each convex hull H_i obtained earlier, obtain the set of voxels V_i that have LH values lying within H_i.
2. For each cluster i, use the set of voxels V_i as the initial seeds for the region growing. The parameter τ_i is used as the LH tolerance.
   (a) Add each voxel in V_i to the output volume O_i.
   (b) For each voxel in O_i, add neighboring voxels to O_i only if they do not already belong to O_i and have LH values within τ_i of the seed voxel.
   (c) Repeat step 2b until no more voxels can be added to O_i.
3. For each pair of enhanced output volumes O_a and O_b, merge them if they overlap.

4.4 Assignment of Visual Parameters for TF Design

Our strategy to assign visual parameters to a cluster is based on the size of the region in the volume described by the cluster and the relative distance between that region and its neighbors. The *size* of the region R_i corresponding to the cluster C_i is coarsely estimated by the standard deviation σ_i of the positions of all the voxels $v_j = \left(v_x^j, v_y^j, v_z^j\right) \in R_i$

$$\sigma_i = \sqrt{\frac{1}{N_i} \sum_{v_j \in R_i} (v_j - \mu_i)^2}, \tag{3}$$

where N_i is the number of voxels in R_i, and μ_i is the mean of the positions of all voxels in R_i:

$$\mu_i = \frac{1}{N_i} \sum_{v_j \in R_i} v_j. \tag{4}$$

The *distance* between two regions R_i and R_j is defined as the Euclidean distance between the two corresponding mean values:

$$D\left(R_i, R_j\right) = \sqrt{\left(\mu_x^i - \mu_x^j\right)^2 + \left(\mu_y^i - \mu_y^j\right)^2 + \left(\mu_z^i - \mu_z^j\right)^2} \tag{5}$$

A region R_i *occludes* region R_j if

$$\begin{cases} \sigma_i > \sigma_j \\ \sigma_i > k_d D\left(R_i, R_j\right) \end{cases}, \tag{6}$$

where $k_d \geq 1$ is a pre-defined value. The opacity α_i assigned to region R_i is calculated by

$$\alpha_i = \frac{\alpha_i^*}{k_s\left(S_i + 1\right)}, \tag{7}$$

where k_s is an adjustable factor, S_i is the number of regions occluded by R_i, and α_i^* is the value corresponding to σ_i in the linear mapping of $\left[\min_j \sigma_j, \max_j \sigma_j\right]$ to a predefined opacity range $[\alpha_{\min}, \alpha_{\max}]$:

$$\alpha_i^* = \frac{\max_j \sigma_j - \sigma_i}{\max_j \sigma_j - \min_j \sigma_j}\left(\alpha_{\max} - \alpha_{\min}\right) + \alpha_{\min} \tag{8}$$

Since smaller structures are more likely to be occluded than larger structures, this method of opacity assignment renders large structures more transparent than small structures. For the enhancement of voxels near the boundaries, the voxel opacity α_v^i corresponding to the voxel v in the region R_i is individually modulated by the ratio of its gradient magnitude and the maximum gradient magnitude of all the voxels in the region:

$$\alpha_v^i = \alpha_i \frac{\|\nabla v\|}{\max_{u \in R_i} \|\nabla u\|}. \tag{9}$$

The color parameter is difficult to assign as the materials have true-colors that cannot be discerned from the CT/MRI volumes; assigning appropriate colors thus requires external knowledge. In our method, the color of each region can be assigned according to the ratio between the size of the region and the maximum size of all the regions, mapped onto a cold-to-hot spectrum. This operation will map small regions to hot colors, and large regions to cooler colors. Alternatively, since the number of regions is relatively small in most cases, we can use a pre-defined color array for this mapping. In addition, the color c_v^i of the individual voxel v in the region R_i is scaled by the ratio of its intensity value f_v and the maximum intensity of all voxels in the region:

$$c_v^i = c_i \frac{f_v}{\max_{u \in R_i} f_u}. \tag{10}$$

For a better rendering result, this scaling is only applied to the brightness value of the corresponding color in the HSV color space.

5 Results and Discussion

Four 16-bit CT volumes were used in our experiments: the Tooth ($256 \times 256 \times 161$), Feet ($256 \times 256 \times 125$), Head ($128 \times 256 \times 156$), and Pig ($256 \times 256 \times 128$) datasets. The computing platform was a 2.66 GHz Intel i5-750 system equipped with 4 GB RAM and a NVIDIA Quadro FX 3800 graphics card. The times required to compute the pre-processed data were 151s, 213s, 200s, and 205s, respectively. Using a GPU-based renderer employing ray marching through a 3-D texture and setting the display window resolution to 512×512 pixels, a real-time frame rate was achieved for all the four datasets. In our experiments, the value of k_d was set to 1.

Figure 5 shows the result of applying our method in automatic mode to the Tooth data set. The bandwidth B_w was chosen as 7% of max_{LH}, and the total time for clustering was 157ms. The clustering result generated by the mean shift clustering algorithm (Figure 5(b)) closely resembles the optimal manual LH

clustering from a previous work [30]. Hence, the mean shift algorithm is capable of quickly generating clusters of similar quality to semi-automatic methods. The clustering speed is also sufficiently fast to allow the user to interact with the bandwidth parameter and receive the updated visualization results on the fly. When operating in the automatic mode, non-boundary clusters (clusters along the main diagonal of the LH histogram) are rendered with a low opacity. Occluding regions are also assigned lower opacity values to ensure that smaller and interior structures are visible. These steps ensure that separate regions within the volume are visible and distinct in the resulting visualization (Figure 5(c)).

Figure 6 shows the same volume rendered in the semi-automatic mode. The semi-automatic mode allows user to modify the TFs generated previously in the automatic mode. This TF modification is performed on the polygonal approximations of the clusters. In Figure 6(a) the opacity of the cylinder was set to 0 and the pulp-dentine boundaries were set to the same color. This pulp-dentine boundary was separated into two disjoint clusters on the LH histogram because of the thin object effect[30]. After manually adjusting the color and opacities for the two clusters, the rendering result was improved (Figure 6(b)). However, some discontinuities in the pulp boundary still existed. These discontinuities cannot be resolved by clustering on the LH space, or similar methods that rely solely on the LH histogram for classification. Our algorithm includes a region growing step to address these issues. Figure 6(c) shows the result after region growing was performed. The discontinuity in the pulp has been filled by the region growing algorithm to yield a single continuous boundary.

For the Feet dataset (Figure 7), the automatic mode with B_w as 7% of max_{LH} was employed to generate the initial clusters for the LH histogram (Figure 7(a)) and initial rendering (Figure 7(b)). The clustering operation took 3578ms to complete. The opacities of the skin and base plate were edited in the semi-automatic mode to obtain the final visualization (Figure 7(c)), which clearly showed the bones within the feet.

For the Head dataset (Figure 8), we used our algorithm in the automatic mode and varied k_s to examine its effect on the visualization. B_w was set to 6% of max_{LH} and the clustering operation was completed in 4594ms. In Figures 8(b) and 8(c), the TF assignment algorithm was run with k_s set to $k_s = 0.1$ and $k_s = 0.3$, respectively. The results confirm that by increasing k_s, occluded internal regions can be selectively revealed. This demonstrates that our algorithm is capable of automatically assigning colors to distinct regions within volumes, and also capable of automatically assigning

the opacities of each region such that all regions are visible and not occluded.

For the Pig dataset (Figure 9) which we acquired from a surgical planning experiment, finding a suitable TF is difficult due to the complexity and number of structures within the volume. It is difficult for the user to properly select any clusters from the LH histogram. Mean shift clustering ($B_w = 4\%max_{LH}$) alleviates this problem by producing an initial set of clusters (Figure 9(a)) which can be quickly modified to achieve the desired visualization. Due to the complexity of the volume, clustering took relatively more time to complete (7500ms). The results from the automatic mode (Figures 9(b), 9(c)) show that the regions of the volume that could be important for surgical planning, such as the bones, blood vessels, and surgical markers, are clearly visible. Hence, our automatic method is suitable for medical visualization, particularly for surgical planning tasks, where good visualization with clear indication of the regions of interest is important.

The visualization results show that our automatic method is capable of assigning the visual properties of color and opacity to obtain good renderings. Comparing with Šereda's method that uses hierarchical clustering [31], our method does not need to generate initial clusters which may strongly affect the rendering results. Furthermore, the user is not required to adjust the cluster colors or opacities as these are determined automatically by our algorithm.

6 Conclusion

We have shown in our experimental results that the proposed method is able to automatically generate useful LH clusters, and that the method of assigning TFs based on the size and locations of the clustered regions is feasible in many cases. The automatically generated clusters, represented in bounding polygons, are simple to adjust manually. The automatically generated TFs, colored based on the relative sizes of regions, also help the user to identify and manipulate the desired regions. These features significantly reduce the time and effort required to obtain TFs for good visualization of volumetric images. Our proposed method is automatic and avoids the extensive parameter tuning required in existing semi-automatic approaches.

Our automatic method has visual quality close to that of existing semi-automatic methods. This visual quality can be further improved while maintaining or improving the current level of automation. One idea that we are currently investigating is to incorporate other features, preferably spatial features, to reduce our method's reliance on the separability of boundaries in

LH space. This should improve both the quality and the robustness of the automated algorithm.

References

1. Andrew A (1979) Another efficient algorithm for convex hulls in two dimensions. Information Processing Letters 9(5):216–219
2. Bajaj CL, Pascucci V, Schikore DR (1997) The contour spectrum. In: Proceedings of IEEE Visualization, pp 167–173
3. Correa CD, Ma KL (2008) Size-based transfer functions: A new volume exploration technique. IEEE Transactions on Visualization and Computer Graphics 14(6):1380–1387
4. Fukunaga K, Larry HD (1975) The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Transactions on Information Theory 21(1):32–40
5. Hadwiger M, Fritz L, Rezk-Salama C, Höllt T, Geier G, Pabel T (2008) Interactive volume exploration for feature detection and quantification in industrial CT data. IEEE Transactions on Visualization and Computer Graphics 14(6):1507–1514
6. He T, Hong L, Kaufman A, Pfister H (1996) Generation of transfer functions with stochastic search techniques. In: Proceedings of IEEE Visualization, pp 227–234
7. Hladuvka J, König A, Gröller E (2000) Curvature-based transfer functions for direct volume rendering. In: Proceedings of Spring Conference on Computer Graphics, pp 58–65
8. Hong D, Ning G, Zhao T, Zhang M, Zheng X (2003) Method of normal estimation based on approximation for visualization. Journal of Electronic Imaging 12(3):470–477
9. Huang R, Ma KL (2003) RGVis: Region growing based techniques for volume visualization. In: Proceedings of Pacific Conference on Computer Graphics and Applications, pp 355–363
10. Kindlmann G, Durkin JW (1998) Semi-automatic generation of transfer functions for direct volume rendering. In: Proceedings of IEEE Symposium on Volume Visualization, pp 79–86
11. Kindlmann G, Whitaker R, Tasdizen T, Möller T (2003) Curvature-based transfer functions for direct volume rendering: Methods and applications. In: Proceedings of IEEE Symposium on Volume Visualization, pp 513–520
12. Knig AH, Gröller EM (1999) Mastering transfer function specification by using VolumePro technology. In: Proceedings of the 17th Spring Conference on Computer Graphics
13. Kniss J, Kindlmann G, Hansen C (2001) Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In: Proceedings of IEEE Symposium on Volume Visualization, pp 255–262
14. Kniss J, Kindlmann G, Hansen C (2002) Multi-dimensional transfer functions for interactive volume rendering. IEEE Transactions on Visualization and Computer Graphics 8(3):270–285
15. Lum EB, Ma KL (2004) Lighting transfer functions using gradient aligned sampling. In: Proceedings of IEEE Visualization, pp 289–296
16. Lundström C, Ljung P, Ynnerman A (2005) Extending and simplifying transfer function design in medical volume rendering using local histograms. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp 263–270
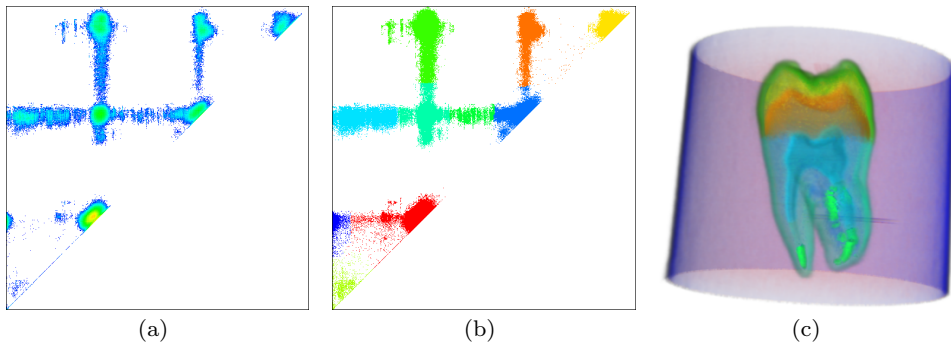
Fig. 5 Automatic TF design for rendering the Tooth dataset: (a) The LH histogram; (b) Generated clusters; (c) Rendered image from clusters.
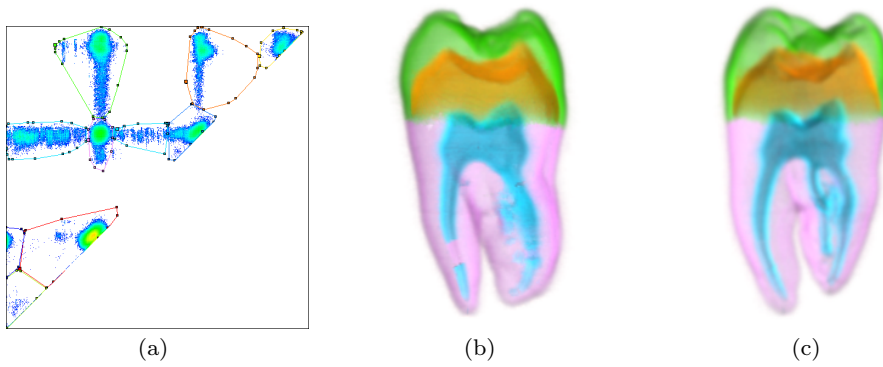

Fig. 6 Semi-automatic TF design for rendering the Tooth dataset: (a) New TF based on approximated polygons; (b) Rendered image; (c) Rendered image using Region growing.
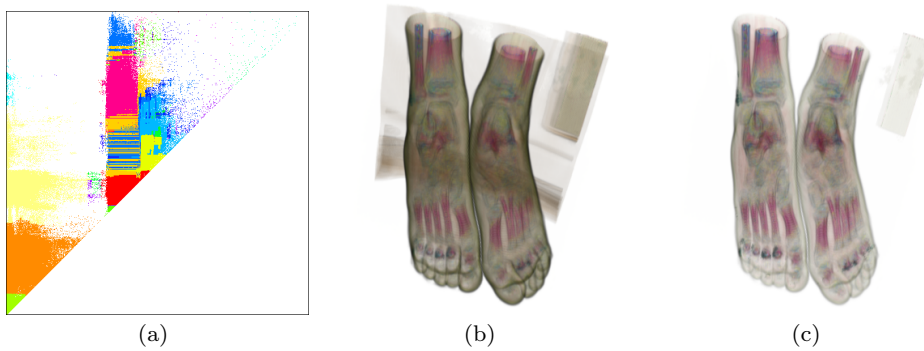

Fig. 7 Volume rendering of the Feet dataset: (a) Clusters; (b) Rendered image with $k_s = 0.3$; (c) Rendered image with $k_s = 0.3$ then decrease the opacity of the skin and set zero-opacity for the back plate.

17. Lundström C, Ljung P, Ynnerman A (2006) Local histograms for design of transfer functions in direct volume rendering. IEEE Transactions on Visualization and Computer Graphics 12(6):1570–1579

18. Lundström C, Ynnerman A, Ljung P, Persson A, Knutsson H (2006) The α-histogram: Using spatial coherence to enhance histograms and transfer function design. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp 227–234

19. Maciejewski R, Chen W, Woo I, Ebert DS (2009) Structuring feature space - a non-parametric method for volumetric transfer function generation. IEEE Transactions on Visualization and Computer Graphics 15(6):1473–1480

20. Marks J, Andalman B, Beardsley PA, Freeman W, Gibson S, Hodgins J, Kang T, Mirtich B, Pfister H, Ruml W, Ryall K, Seims J, Shieber S (1997) Design galleries: a general approach to setting parameters for computer graphics and animation. In: Proceedings of SIGGRAPH, pp 389–400

21. de Moura Pinto F, Freitas CMDS (2007) Design of multi-dimensional transfer functions using dimensional reduction. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp 131–138

22. Pekar V, Wiemker R, Hempel D (2001) Fast detection of meaningful isosurfaces for volume data visualization. In:
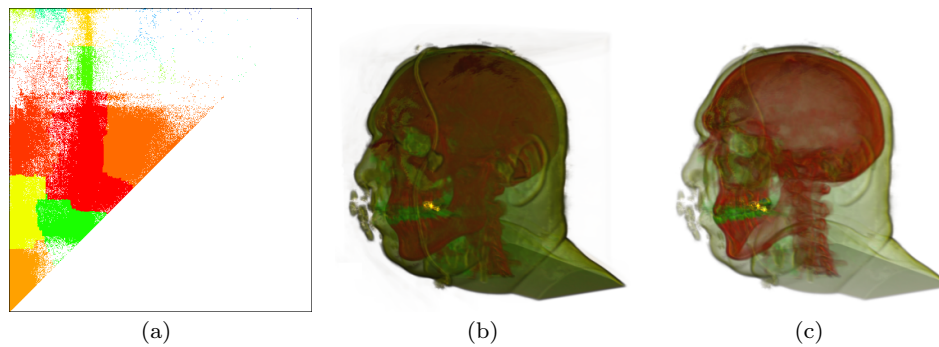
Fig. 8 Volume rendering of the VisMaleHead dataset: (a) Clusters; (b) Rendered image with $k_s = 0.1$; (c) Rendered image with $k_s = 0.3$.
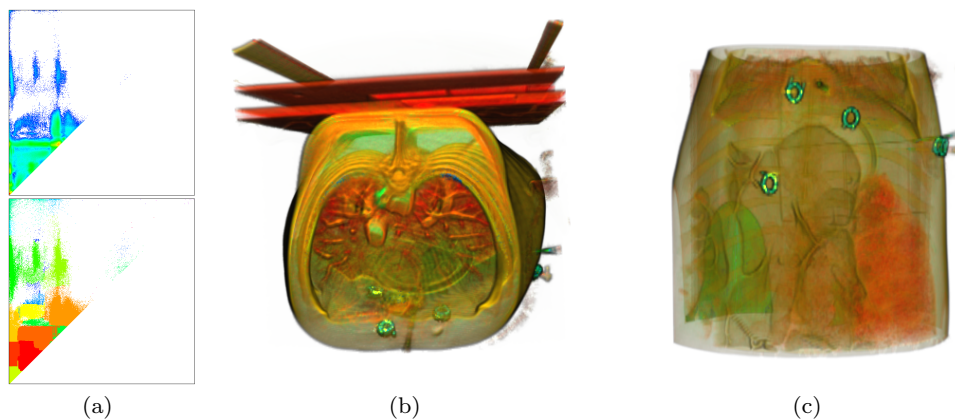

Fig. 9 Volume rendering of the Pig dataset: (a) LH histogram (upper) and clusters (lower); (b) and (c) Rendered images using automatic mode.

Proceedings of IEEE Visualization, pp 223–230

23. Praßni JS, Ropinski T, Hinrichs KH (2009) Efficient boundary detection and transfer function generation in direct volume rendering. In: Proceedings of the 14th International Fall Workshop on Vision, Modeling, and Visualization (VMV09), pp 285–294

24. Röttger S, Bauer M, Stamminger M (2005) Spatialized transfer functions. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp 271–278

25. Tappenbeck A, Preim B, Dicken V (2006) Distance-based transfer function design: Specification methods and applications. In: Proceedings of Simulation und Visualisierung, pp 259–274

26. Teistler M, Breiman RS, Liong SM, Ho LY, Shahab A, Nowinski WL (2007) Interactive definition of transfer functions in volume rendering based on image markers. International Journal of Computer Assisted Radiology and Surgery 2(1):55–64

27. Tzeng FY, Ma KL (2004) A cluster-space visual interface for arbitrary dimensional classification of volume data. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp 17–24

28. Tzeng FY, Lum EB, Ma KL (2003) A novel interface for higher-dimensional classification of volume data. In: Proceedings of IEEE Visualization, pp 505–512

29. Tzeng FY, Lum EB, Ma KL (2005) An intelligent system approach to higher-dimensional classification of volume data. IEEE Transactions on Visualization and Computer Graphics 11(3):273–284

30. Šereda P, Bartroli AV, Serlie IWO, Gerritsen FA (2006) Visualization of boundaries in volumetric data sets using LH histograms. IEEE Transactions on Visualization and Computer Graphics 12:208–218

31. Šereda P, Vilanova A, Gerritsen FA (2006) Automating transfer function design for volume rendering using hierarchical clustering of material boundaries. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp 243–250

32. Wesarg S, Kirschner M (2009) Structure size enhanced histogram. In: Brauer W, Meinzer HP, Deserno TM, Handels H, Tolxdorff T (eds) Bildverarbeitung für die Medizin 2009, Informatik aktuell, Springer Berlin Heidelberg, pp 16–20

33. Wesarg S, Kirschner M, Khan MF (2010) 2D histogram based volume visualization: combining intensity and size of anatomical structures. International Journal of Computer Assisted Radiology and Surgery pp 1–12