A clustering-based system to automate transfer function design for medical image visualization

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

Abstract Finding good transfer functions for rendering medical volumes is difficult, non-intuitive, and time-consuming. We introduce a clustering-based framework for the automatic generation of transfer functions for volumetric data. The system first applies mean shift clustering to oversegment the volume boundaries according to their low-high (LH) values and their spatial coordinates, and then uses hierarchical clustering to group similar voxels. A transfer function is then automatically generated for each cluster such that the number of occlusions is reduced. The framework also allows for semi-automatic operation, where the user can vary the hierarchical clustering results or the transfer functions generated. The system improves the efficiency and effectiveness of visualizing medical images and is suitable for medical imaging applications.

Keywords Transfer function design · Volume rendering · LH histogram · Clustering

B.P. Nguyen (✉) · W.-L. Tay
Department of Electrical and Computer Engineering,
National University of Singapore, Singapore, Singapore
e-mail: phubinh@nus.edu.sg

W.-L. Tay
e-mail: tayweiliang@nus.edu.sg

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

S.-H. Ong
Department of Electrical and Computer Engineering, and Division of Bioengineering, National University of Singapore, Singapore, Singapore
e-mail: eleongsh@nus.edu.sg

1 Introduction

Direct volume rendering is a powerful tool that enables 3-D volumetric data to be visualized. Direct volume rendering has several important applications, particularly in medicine where several imaging modalities such as computed tomography and magnetic resonance imaging can generate 3-D volumes. However, it is often difficult to obtain a volume rendering that reveals all the relevant and important structures within the volume. In practice, much manual parameter tweaking of the transfer function (TF), which maps the data properties (e.g. scalar value and gradient magnitude) to optical properties (opacity and color), is required for good visualization results. There are three significant difficulties in TF design for volume rendering: it is non-intuitive to manipulate TFs to obtain desired visualization results, it requires operators with high technical expertise and experience, and it is time-consuming.

Several techniques have been developed to assist the design of TFs. Early approaches applied multiple different TFs and then analyze the resulting rendered images to construct an optimal transfer function. This class of methods is known as image-driven methods, as they operate on the rendered images rather than directly on the data. Image-driven methods are intuitive as the tuning of TFs is abstracted away from the objective of finding good renderings [13]. Unfortunately, image-driven methods suffer from dependency on image-related parameters and often require further user input to evaluate the renderings. Consequently, the majority of transfer function design techniques are data-driven methods that directly analyze the underlying data.

Transfer functions typically operate on the input domains of data values and data gradients, and additional feature domains such as size [3] have been proposed to increase the separability of different structures within volumes. However, multi-dimensional transfer functions are typically difficult to manipulate, motivating dimensionality reduction [23] and/or clustering [30] techniques to reduce the interaction space. TFs have also been generated by applying region growing [12, 29] or machine learning techniques [31, 32] to learn the properties of regions of interest based on user selected features/regions. However, while these methods can improve the ease and intuitiveness of TF design by simplifying the interaction space, the degree of interaction is usually reduced as well. Essentially, the TF space has been pruned and the user loses some control over the resulting rendering.

The LH space, which describes voxels based on the two materials forming the boundary, has recently been proposed as a feature domain in TF design [33, 34]. Compared to the more conventional TF domains of scalar value and its derivatives, the LH histogram, a 2-D representation of the LH space, represents boundaries more compactly and robustly. Furthermore, in LH space boundaries appear as blobs rather than arches, which reduces the amount of overlap and increases the separability of each boundary. A further advantage is that clustering techniques can be applied to the LH space without the need for complex distance functions.

The system introduced in this paper incorporates multiple clustering steps to automatically identify the material boundaries in volumetric data. Transfer functions can then be applied to each material boundary using an automated transfer function design module. The system allows users to interactively select the number of clusters and to modify the transfer functions assigned to each cluster. Further changes to the visualization output can also be achieved by changing the clustering parameters. In summary, the proposed system significantly reduces 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

Transfer function design has been the focus of several studies in the past decade. The most established approach involves the use of the derivatives to design TFs, because the derivatives are the simplest indicator of whether a voxel lies on a boundary. Kindlmann et al. [14] generated multi-dimensional TFs incorporating the first derivative of the scalar value (i.e., gradient). It was found that under the 2-D TF domain of intensity and gradient magnitude, material boundaries are distributed as arches. Therefore, optical properties can be assigned to each different material boundary simply by selecting or approximating the arch shapes through the use of TF interaction widgets. Unfortunately, for many complex datasets, the arches of different boundaries would often overlap in the TF space, making it difficult or impossible to properly isolate one material from another. Some works [16, 17] have included the second directional derivative to help disambiguate between different boundaries. Nonetheless, the issue of blurring in the intensity-derivative histogram, caused by noise, was not adequately resolved. Lum et al. [18] used the two intensity values on both sides of a boundary to design TFs, making the assumption that the width of the boundary represented by the distance between these two sample positions varies with the amount of blur in the volume. Šereda et al. [33] proposed another method to represent boundaries in terms of the materials that form the boundary interface. They traced the gradient path along the negative and positive gradient directions of voxels to obtain the low and high intensity values representing the materials forming the boundary. By compiling the LH values of the volume voxels into a 2-D histogram, it was discovered that in the *LH histogram*, boundaries appeared as blobs rather than arches (under the intensity-gradient magnitude histograms). LH histograms present a number of advantages over the intensity-gradient histogram. Blobs are easier to parameterize for clustering, and blobs are less likely to overlap as compared to arches. LH histograms are also more robust to noise, bias, and partial volume effects. These advantages mean that LH histograms can enable boundaries to be more easily separated through manual selection or automatic clustering. One weakness with LH histograms is the time and expense of computing the LH values, though Praßni et al. [26] have recently proposed an alternative and less expensive method for the efficient construction of LH histograms. Praßni et al. then used their efficiently generated LH histograms to perform a semi-automatic generation of LH TFs.

Apart from gradient-based methods, another popular approach to designing TFs is to employ curvature-based information. Bajaj et al. [1] introduced the *contour spectrum* to describe isosurfaces using a variety of scalar and contour attributes. The appropriate isosurfaces were then visualized by selecting the relevant isovalues from an interactive user interface. Pekar et al. [25] proposed a method for automatically detecting and visualizing isosurfaces using a cumulative Laplacian-weighted intensity histogram. Kindlmann et al. [15] also proposed a methodology inspired by Hladuvka et al. [10] for computing high quality curvature measurements. The curvature measurements were subsequently applied to generate curvature-based TFs for volume rendering.

Recently, *feature size* has also been employed as a feature in multi-dimensional TFs in order to separate structures with similar intensity or gradient values. Correa et al. [3] represented feature size using scale fields by assigning to each voxel a scale parameter dependent on the local scale of the feature containing the voxel. The relative size of features was then used in a size-based TF to map visual parameters to the voxels. Instead of using scale fields, Hadwiger et al.

[9] used region growing to estimate the feature size. A similar approach by Wesarg et al. [35, 36] estimates the structure size by searching for neighboring voxels falling within an user-specified intensity tolerance range. The image generated from the structure size data was used as the second property of a *structure size enhanced* (SSE) histogram. In our method, the size of regions is estimated using a new method and then employed to minimize occlusions during the automatic assignment of visual parameters.

A problem with multi-dimensional TF approaches is that the features typically include only local information such as derivatives or curvature measurements. Therefore, many methods have attempted to extend the traditional 1-D or 2-D histograms with spatial information. Local histograms were proposed by Lundström et al. [19, 20] to separate different tissues by considering the distribution of intensities in the local neighborhood. They also introduced the *α-histogram* [21] to incorporate the property of spatial coherence to produce a global histogram. The global histogram consists of the normalized sum of local histograms of various disjoint local regions, each raised to a power of $\alpha > 1$, which amplifies spatially concentrated value ranges and enlarges the peaks belonging to different materials. Röttger el al. [27] bin the voxels in a 2-D histogram and computed the means and variances of all voxels in each bin, then used these statistics to produce a maximum feature radius for classifying the histogram. Tappenbeck et al. [28] introduced a distance-based TF which assigns optical properties to structures based on the distance to a selected reference structure in order to hide or emphasize structures.

One difficulty of the TF design is the lack of a measure to quantify the quality of TFs. Research works by Correa et al. [5, 6] use *visibility*, the contribution of a sample in terms of opacity to the final image, as a primitive to guide TF design in both manual and automatic modes. With the help of *visibility histograms*, which is a multi-dimensional representations of the distribution of visibility in a rendered image, users can produce TFs that maximize the visibility of the intervals of interest. In the automatic mode, TFs are generated in order to minimize the mismatch between the opacity TF defined by the user and the computed one, and maximize the visibility of important structures. Visibility was also used in Chan et al. [2] together with shape and transparency to evaluate and enhance the perceptual quality of transparent structures in the rendered image. Correa et al. [4] introduced the *occlusion spectrum*, which is the distribution of weighted averages of the intensities in a spherical neighborhood of each voxel. With this spectrum, better 2-D TFs that can help classify complex datasets in terms of the spatial relationships among features can be produced.

Apart from finding new feature domains for TFs, much work has also been put into developing algorithms to separate different regions in the TF domain. The ISODATA technique was used by Tzeng et al. [30] to extract material classes from the cluster space, and the material classes were then used to design the TFs. Šereda et al. [34] grouped voxels based on their LH values and spatial similarity using hierarchical clustering. Maciejewski et al. [22] used non-parametric clustering on the TF features space to guide the design of TFs. Zhou et al. [37] developed a parallel mean shift technique to assign visual parameters to different regions of the volume by clustering voxels based on their scalar values and spatial locations. Mean shift clustering was also used in our earlier work [24] to cluster voxels in LH space, where each cluster represents a single material boundary. Each material boundary was then assigned visual parameters based on the extent the boundary occludes other boundaries. In this paper, we apply a multi-stage clustering process that uses mean shift and hierarchical clustering. This multi-stage clustering is non-parametric and robust and preserves a large degree of freedom for user manipulation of results.

3 Architecture overview

Volumetric data consist of multiple material boundaries where the boundary voxels have similar LH values and spatial locations. Our system offers three modes of operation, two automatic and one semi-automatic, to visualize volumetric data. The first automatic mode uses mean shift clustering to group voxels based on their LH values, and then performs hierarchical clustering on these groups. Each resulting cluster is assigned optical properties using an automated TF design module. This *two-step clustering* automatic mode is suitable for simple datasets. The second automatic mode uses three-stage clustering to group voxels into boundary clusters where the visual parameters of color and opacity are then assigned to the voxels within each separate cluster. The clustering of boundary voxels and the assignment of visual parameters is performed by an automatic transfer function module, as with the first automatic mode. This *three-step clustering* automatic mode requires more computational resources, but is capable of generating better visualization results on more complex datasets. Finally, our system is capable of operating in the semi-automatic mode, where a user interaction module allows the user to dynamically modify the system parameters or intermediate results obtained from either automatic mode. Figure 1 presents an overview of our system.

3.1 Pre-processing

The purpose of the pre-processing module is to precompute and store the gradient vector and LH values corresponding to each voxel. The gradients are calculated using

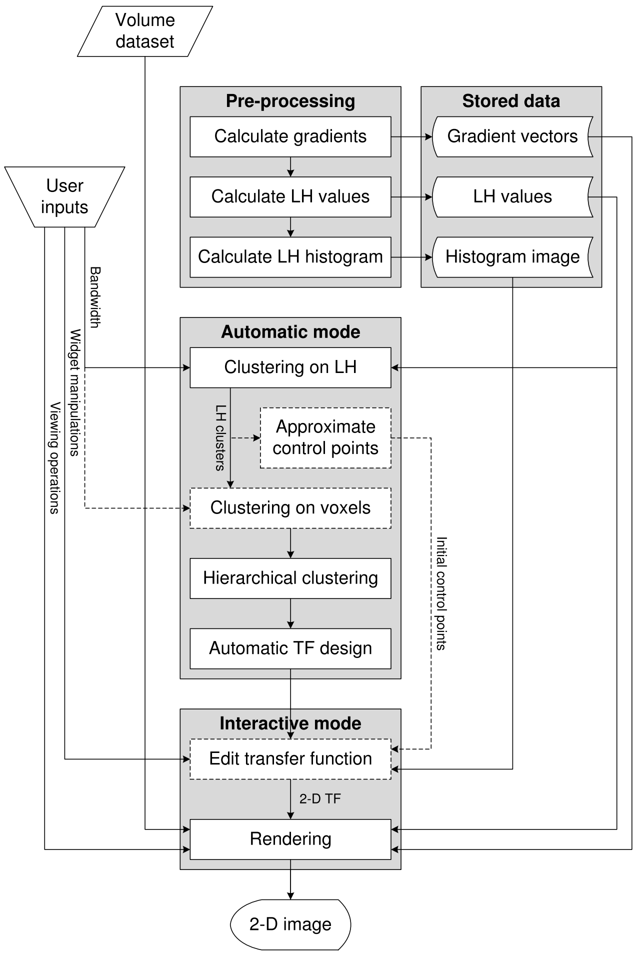

Fig. 1 Overview of the system. *Dotted boxes* represent optional modes

Hong's method [11], which uses a second-degree polynomial function to approximate the density function in a local neighborhood whereby the polynomial coefficients are obtained by minimizing the approximation error. The polynomial coefficients allow for the voxel first partial derivatives (i.e., gradient) and second directional derivatives to be determined from the voxel intensity and position. This approximation method offers several advantages: (1) it is possible to account for the difference between the pixel spacing and the spacing between slices which often exists in medical datasets; (2) estimating the gradient vector of an arbitrary sampling point between voxels is computationally cheap (3) this method is robust to noise because it does not interpolate the curve passing through all the given data points. From the gradient values, the lower (L) and higher (H) intensity values for each voxel can be computed by tracking the boundary path in both directions [24]. The LH histogram obtained by binning the voxel LH values is stored as an image of 512×512 pixels. 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 Automatic TF design using two-step clustering

Mean shift clustering [8] is used to oversegment the LH space into multiple groups. Hierarchical clustering using a similarity measure based on neighborhood relations [34] then merges these groups into clusters. An automatic TF design module is then used to assign colors and opacities to each cluster such that the number of occlusions is reduced. If the resulting visualization is unsatisfactory, the user can modify the visualization results using the bounding polygon widget in the user interaction module. This automatic mode is partly based on the contributions in [24]. In this paper, the mean shift clustering algorithm is refactored to produce more stable and robust results. The new mean shift algorithm is implemented in CUDA and uses the GPU (graphics processing unit) to reduce the clustering time.

3.3 Automatic TF design using three-step clustering

The automatic mode with three-step clustering uses a more complex clustering scheme that incorporates spatial information, and is more suitable for complex data compared to that of the automatic mode described above. The automatic TF design module is responsible for generating the boundary voxel clusters, and for the assignment of TFs to each cluster. To generate the boundary voxel clusters, a three-stage clustering process is performed. First, mean shift clustering is applied twice in series. The first mean shift clustering is performed to segment the LH space into multiple LH clusters, while the second mean shift clustering further segments voxels in each LH cluster based on their spatial proximity. A third clustering step using hierarchical clustering then generates a cluster hierarchy. The cluster hierarchy is used to obtain a final set of voxel clusters, each of which represents a single material-material boundary in the volume. All clusters are then automatically assigned colors and opacities based on 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 adjust the visualization results using the user interaction module.

3.4 Semi-automatic user interaction

The user interaction module operates in parallel with the automatic TF design module and allows the user to adjust the system parameters to obtain more desirable visualizations. The user can influence the result at various stages in the system to achieve varying effects on the visualization output. First, the linking threshold can be interactively modified to vary the number of material boundaries. Second, the automatically generated transfer function assigned to a cluster can also be interactively adjusted to change the opacity or color of the target cluster. Third, if a greater degree of adjustment is desired, the mean shift bandwidths for the LH and spatial clustering steps can be changed to refine the boundaries, or the results of the LH clustering step can be adjusted using an interaction widget; however, these changes are more extensive and may require more time to compute.

4 Processes in the automatic transfer function design system

In this paper, a new system is introduced for the assignment of optical properties to each distinct material boundary in volumetric data. This algorithm assumes that boundary voxels have similar LH values and are located close together in the volume. Based on these two assumptions, the algorithm performs clustering to group similar voxels together. For simple datasets where there is little overlap between blobs in the LH space, clustering on the LH space is sufficient to separate the different structures [24, 33]. However, for more complicated volumes where structures cannot be separated by their LH value alone, two-step clustering is not sufficient. Hence, a series of clustering steps is applied to oversegment the volume voxels into many small clusters. The oversegmentation ensures that the voxels within each cluster have a high probability of belonging to a single "true" material boundary. Subsequently, the set of oversegmented clusters can then be recombined into larger clusters using hierarchical clustering. The advantage of using hierarchical clustering as the final clustering step is that it generates the results as a hierarchy of clusters. The linking threshold for the hierarchy of clusters can be cheaply adjusted on-the-fly to generate more or fewer number of clusters, depending on user preference, without needing to restarting the expensive clustering algorithm.

Once the clustering processes have been completed, each boundary in the volume is grouped as a separate cluster for the assignment of visual parameters. A transfer function design algorithm then automatically computes and assigns transfer functions for each cluster. These TFs for each cluster can also be individually modified according to user preference.

The algorithm consists of three main clustering processes and a transfer function generation process. These processes are described below.

4.1 Mean shift clustering in LH space

Mean shift is a popular non-parametric clustering algorithm that seeks the modes of the given sample space. Mean shift clustering offers two main advantages over other clustering methods: (1) no limitations or assumptions on the structure or distribution of the data are made, (2) and the number of clusters does not need to be specified a priori. For the first clustering step, each voxel is clustered according to its LH value. A bandwidth parameter B_{LH} controls the sensitivity of the mean shift clustering. From our experiments, a good B_{LH} lies between 7%–9% of the maximum LH value, $\max_{LH} = \max(\max_L, \max_H)$. As a further performance optimization, mean shift clustering is computed over discrete values in the LH histogram, and each LH point is weighted during the mean computation step by the number of times it occurs in the volume. Since all voxels can only have discrete LH values and the LH histogram is relatively sparse, this reduces the time and memory required for mean shift clustering. The procedure of mean shift clustering is summarized in the following algorithm:

1. Define a clustering parameter, window bandwidth B_{LH}.
2. For a point in the LH histogram, find all points that have LH values within the bandwidth B_{LH}.
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_{LH} where ρ is a threshold 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 $B_{LH}/2$ of each other are also grouped as one cluster.

After this clustering step, the data will be grouped into a few hundred clusters, each containing a large number of voxels with similar LH values.

In our implementation of mean shift clustering using CUDA, we use two 1-D textures to store the LH points and their corresponding weights. The cache mechanism of texture memory can recur memory traffic when the LH points and their weights are read by the kernel. The kernel in each thread is programmed to process a set of LH points by finding their means according to the algorithm above. After all the means are determined, the LH points are grouped into different clusters depending on the distance between their means. This GPU-based implementation is about 10 times faster than the program in our previous work [24], making the proposed multi-stage clustering algorithm more practical.

4.2 Mean shift clustering on spatial domain

At the end of the previous step, each cluster contains voxels with similar LH values. However, it cannot be assumed that these voxels belong to the same boundary, unless they are also located close together. In this step, mean shift clustering is applied to cluster each LH cluster based on their spatial coordinates. As a non-parametric clustering method, mean shift clustering is particularly suitable for this task because the material boundaries may have complex shapes. A spatial bandwidth parameter B_{XYZ} controls the sensitivity of the clustering process. After all clusters have been processed, the volume voxels will be grouped into clusters, where each group has similar LH values and are spatially close.

The mean shift clustering algorithm used to cluster the voxels is the same algorithm used in the previous step, with the appropriate change of feature domain and bandwidth parameter.

4.3 Hierarchical clustering of all clusters

Hierarchical clustering is a clustering method which builds a hierarchy of clusters by incrementally merging pairs of clusters together [7]. By stopping the merging based on some external criteria, for example a threshold distance between two clusters, the number of final clusters can be controlled as desired. Here, hierarchical clustering is initialized using each output cluster from the previous step. Then, the two clusters with the lowest pairwise distance are merged together, and this merging process continues until all voxels are grouped under one single cluster. The user is then able to vary the number of boundary clusters by reversing the agglomerative process and breaking clusters in the reverse merge order. The agglomerative hierarchical clustering algorithm is summarized below:

1. Begin with a set of clusters from the previous clustering step.
2. Compute the pairwise distance between all pairs of clusters in C.
3. Let the pair of clusters with the lowest pairwise distance be C_a, C_b. Merge C_a, C_b. Also record the clusters that have been merged, the order of merging, and the distance between C_a and C_b.
4. Update the pairwise distance between the newly merged cluster and all other clusters.
5. Repeat steps 3 and 4 until only one cluster remains.
6. Cluster links are repeatedly cut from the top of the hierarchy until N_C clusters remain.

For the pairwise distance between clusters, the metric is based on the volume similarity measure introduced by Šereda et al. [34]. This volume similarity measure evaluates the number of neighborhood relations between the two clusters. First, for each cluster C_i, the number of neighborhood relations is

$$NR(C_i) = \sum_j NR(C_i, C_j), \tag{1}$$

whereas the neighborhood relations between C_i and C_j is computed by counting the number of 26-neighbors belong to C_j for each voxel v_i in C_i:

$$NR(C_i, C_j) = \sum_{v_i \in C_i} \sum_{v_j \in C_J} N_{26}(v_i, v_j); \quad C_i \neq C_j. \tag{2}$$

The similarity measure is then computed by taking the maximum of the normalized sum of neighborhood relations between the two clusters:

$$s(C_i, C_j) = \max\left\{\frac{NR(C_i, C_j)}{NR(C_i)}, \frac{NR(C_i, C_j)}{NR(C_j)}\right\}. \tag{3}$$

This similarity measure is updated after each merging of clusters. The number of relations in the merged cluster $C_{i\cup j}$ is updated with the following approximation:

$$NR(C_{i\cup j}) = NR(C_i) + NR(C_j) - 2NR(C_i, C_j) \tag{4}$$

while the number of relations and the similarity measure with all other clusters C_k are, respectively, recalculated as

$$NR(C_{i\cup j}, C_k) = NR(C_i, C_k) + NR(C_j, C_k) \tag{5}$$

$$s(C_{i\cup k}, C_k) = \max\left\{\frac{NR(C_{i\cup k}, C_k)}{NR(C_{i\cup k})}, \frac{NR(C_{i\cup k}, C_k)}{NR(C_k)}\right\}. \tag{6}$$

4.4 Assignment of visual parameters

The TF design module is built to satisfy four objectives that describe good visualizations. First, each cluster and material boundary must be visually distinct from all other clusters and material boundaries. Second, within each individual cluster and material boundary, voxels closer to the boundary interface are more important than voxels farther away from the boundary interface. Third, within each individual cluster and material boundary, voxels with different scalar values should have slightly different visual appearances. Fourth, internal structures should not be occluded by larger or outer structures. Therefore, the TF design module first assigns a different opacity and color to each separate cluster, depending on how much the cluster occludes other clusters. Each voxel in a cluster then inherits visual properties from its cluster, modulated by a scaling factor dependent on the voxel's gradient magnitude relative to the gradient magnitude of other voxels in the cluster.

A region is more likely to occlude other regions when it is large and has several close neighboring regions. Hence, the degree by which a region occludes other regions can be estimated by the size of the region in the volume 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 $v_j = (v_x^j, v_y^j, v_z^j) \in R_i$

$$\sigma_i = \sqrt{\frac{1}{N_i} \sum_{v_j \in R_i} (v_j - \mu_i)^2}, \tag{7}$$

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:

$$\mu_i = \frac{1}{N_i} \sum_{v_j \in R_i} v_j. \tag{8}$$

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

$$D(R_i, R_j) = \sqrt{(\mu_x^i - \mu_x^j)^2 + (\mu_y^i - \mu_y^j)^2 + (\mu_z^i - \mu_z^j)^2}. \tag{9}$$

A region R_i *occludes* region R_j if

$$\begin{cases} \sigma_i > \sigma_j \\ \sigma_i > k_d D(R_i, R_j) \end{cases} \tag{10}$$

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

$$\alpha_i = \frac{\alpha_i^*}{k_s(S_i + 1)}, \tag{11}$$

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 $[\min_j \sigma_j, \max_j \sigma_j]$ to a pre-defined opacity range $[\alpha_{\min}, \alpha_{\max}]$:

$$\alpha_i^* = \frac{\max_j \sigma_j - \sigma_i}{\max_j \sigma_j - \min_j \sigma_j} (\alpha_{\max} - \alpha_{\min}) + \alpha_{\min}. \tag{12}$$

Each voxel in a cluster inherits the cluster opacity scaled to the enhance voxels nearer to 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:

$$\alpha_v^i = \alpha_i \frac{\|\nabla v\|}{\max_{u \in R_i} \|\nabla u\|}. \tag{13}$$

The true-colors of the materials cannot be determined from the data volumes alone, hence colors in TFs are typically assigned according to some external criteria or with external knowledge. Here, the color of each region is assigned according to the relative size of the structure, mapped onto a cold-to-hot spectrum (Fig. 2). Hence, small structures will be mapped to hot colors (red) while large regions will be mapped to cool colors (blue). Alternatively, a pre-defined

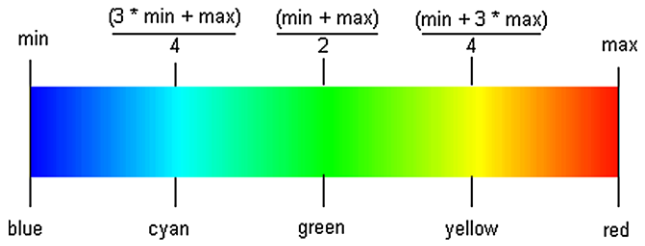

Fig. 2 Cold-to-hot color ramp

color array can be applied for the color mapping as the number of regions tends be small. The color c_v^i of each voxel within a region R_i is further modulated by the ratio of its intensity value f_v and the maximum intensity of all voxels in the region:

$$c_v^i = c_i \frac{f_v}{\max_{u \in R_i} f_u}. \tag{14}$$

This scaling is only applied to the brightness value of the corresponding color in the HSV color space, which means that voxels within the same region have the same hue and saturation values and can hence be differentiated from voxels belonging to other regions.

4.5 Interaction widget for modifying LH clusters

Cluster bounding polygons are a tool to facilitate the manipulation of clusters generated in LH space. This functionality was retained in our system as a interaction widget in the user interaction module. If the user wishes to modify the results of the LH clustering, a convex hull algorithm is first applied to each LH cluster generated using the clustering algorithm to obtain a bounding polygon that describes each cluster. Then, a disambiguation scheme is run on the bounding polygons to resolve any overlaps between pairs of bounding polygons. Each automatically generated cluster is now represented as a convex polygon, and the user can manipulate the clusters by performing vertex or polygon operations. Figure 3 shows the cluster bounding polygons for a sample dataset.

5 Evaluation and discussion

Three 16-bit CT volumes were used in our experiments: Feet ($256 \times 256 \times 125$), Head ($128 \times 256 \times 156$), and Pig ($256 \times 256 \times 128$) datasets. The Feet dataset is from University Hospital of Geneva, Switzerland, and the Head dataset is from National Library of Medicine, National Institutes of Health, USA. The Pig dataset is from our own surgical planning experiments. Results were obtained on a 2.66 GHz Intel i5-750 system equipped with 4 GB RAM and a NVIDIA Quadro FX 3800 graphics card using C++ and

CUDA. The pre-processing time was less than 3 minutes for all datasets used. A GPU-based renderer employing ray marching through a 3-D texture was used for rendering the results and achieved real-time frame rates for all datasets used. The parameter k_d controlling the effect of occlusions in the automatic TF design algorithm was set to 1 in all our trials.

Table 1 contains the system parameters used to generate Figs. 4, 5 and 6. The "Method" column describes the clustering process (two-step or three-step) used for each trial. All figures were generated using the automatic mode with the stated B_{LH} parameters for the LH clustering step and the stated number of clusters for the hierarchical clustering step.

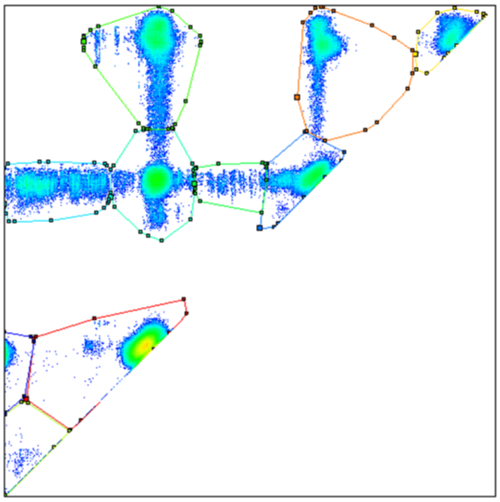
Fig. 3 Demonstration of cluster bounding polygons

For the Feet dataset (Fig. 4), the automatic mode was run with two-step clustering and a B_{LH} of 9% of $\max_{LH}$ to generate the initial LH clusters (Fig. 4(a)). The number of clusters generated with hierarchical clustering was varied to obtain Fig. 4(b) (15 clusters) and Fig. 4(c) (18 clusters). The clustering process took around 10 s to complete, with the majority of the time (90%) being spent on the hierarchical clustering operation. All colors and opacities were automatically assigned using the automatic TF design module. The visualizations clearly show the bones within the feet, with the metatarsal bones being colored differently from the other tarsal bones. The results demonstrates the potential of the TF design system for identifying and visualizing structures with medical volumes.

Table 1 Evaluation parameters

Figure	Dataset	Method	B_{LH}	Clusters
4(b)	Feet	Two-step	9%	15
4(c)	Feet	Two-step	8%	18
5(b)	Head	Two-step	9%	17
5(c)	Head	Two-step	9%	6
6(b, c)	Pig	Three-step	7%	32

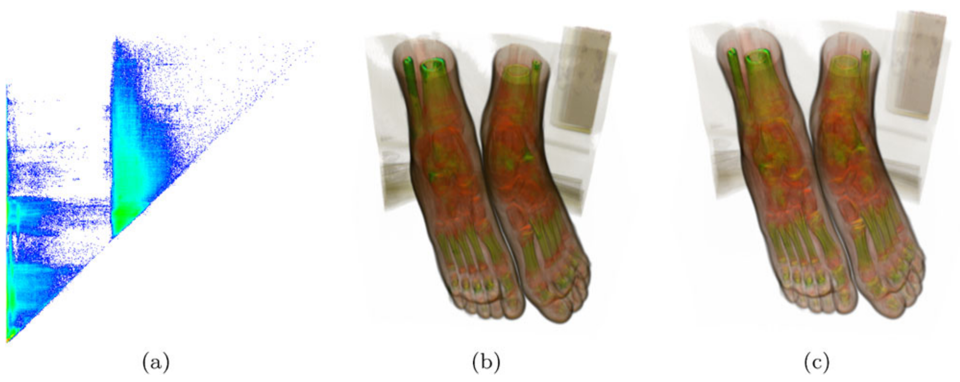

Fig. 4 Volume rendering of the Feet dataset: (**a**) LH histogram; (**b**) Rendered image with 15 clusters; (**c**) Rendered image with 18 clusters

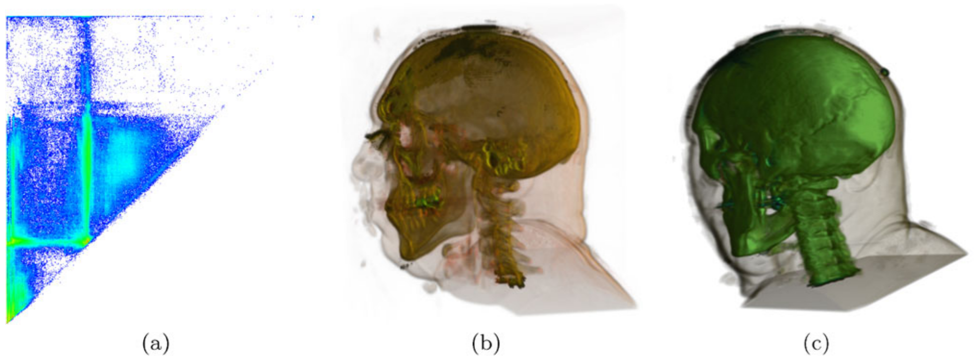

Fig. 5 Volume rendering of the VisMaleHead dataset: (**a**) LH histogram; (**b**) Rendered image with 17 clusters; (**c**) Rendered image with six clusters with a different viewing angle

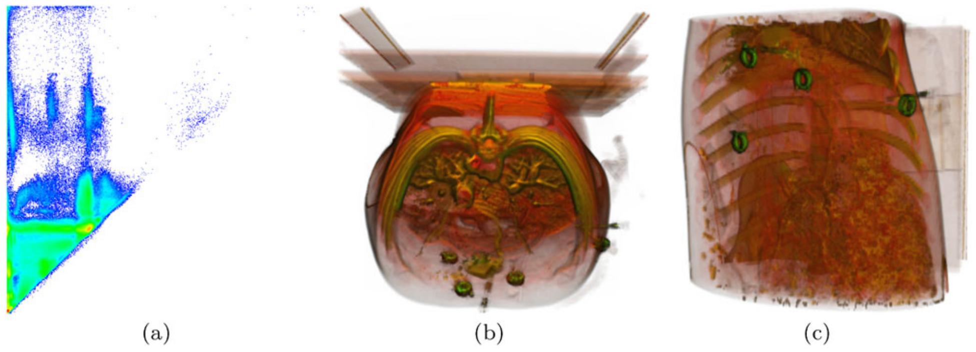

Fig. 6 Volume rendering of the Pig dataset: (**a**) LH histogram; (**b**, **c**) Rendered image with 32 clusters

For the Head dataset (Fig. 5), two-step clustering was used again with a B_{LH} of 9% to generate the initial LH clusters (Fig. 5(a)). Figures 5(b) and 5(c) were generated using hierarchical clustering with 17 and six clusters, respectively. The clustering time was also around 10 s. The results show that adjustment of the number of clusters influences the number of distinct structures visible in the region. By setting the system to output more clusters, more objects are visible in Fig. 5(b). Conversely, the number of unique structures can be reduced by lowering the cluster number, such as in Fig. 5(c) where the main focus was on capturing the crack in the skull region.

The Pig dataset (Fig. 6) is a complex volume with a number of similar structures within the volume. The CT scans were obtained from a robot-assisted surgical experiment conducted on a live pig. It is difficult to select, whether manually or automatically using clustering algorithms, the clusters from the LH histogram. Furthermore, it is not possible to separating the structures from the LH values alone. Figures 6(b), 6(c) were generated using the automatic mode with three-step clustering. The system was capable of producing a visualization that differentiates between the surgically important structures (surgical markers, bones, major blood vessels) within the volume. With minor operator intervention to further improve the visualization, the TF design system is useful for medical and surgical visualization, and it enables the surgeon to accurately and easily plan the surgical intervention during operations.

Our previous work [24] allows for the optical properties of color and opacity to be assigned automatically. The new results using multi-step clustering extend our previous work by demonstrating that difficult and complex structures can be also identified automatically. Taken together, the proposed system is a more complete and robust method for medical volume visualization.

6 Conclusion

We have presented a system for the automatic and semi-automatic generation of TFs for medical volume visualization. The multi-step clustering process incorporates LH and spatial information to cluster and identify complex material boundaries, while the automatic TF design module is able to assign good TFs such that boundaries are not occluded. The proposed system automatically generates good visualizations while preserving a high degree of freedom for the user to adjust the rendering results. Compared with our prior approach [24], the multi-step clustering process introduced in this paper offers a higher degree of user interaction. Furthermore, the addition of voxel spatial relations in the clustering step improves the quality of the extraction material boundaries without the need for region growing. The visualizations generated by the proposed method are comparable to existing state-of-the-art approaches. The user-centric medical visualization system is part of a computer integrated robot-assisted surgical system developed in partnership with the National University Health System in Singapore.

References

1. Bajaj, C.L., Pascucci, V., Schikore, D.R.: The contour spectrum. In: Proceedings of IEEE Visualization, pp. 167–173 (1997)
2. Chan, M.Y., Wu, Y., Mak, W.H., Chen, W., Qu, H.: Perception-based transparency optimization for direct volume rendering. IEEE Trans. Vis. Comput. Graph. **15**(6), 1077–2626 (2009)
3. Correa, C.D., Ma, K.L.: Size-based transfer functions: a new volume exploration technique. IEEE Trans. Vis. Comput. Graph. **14**(6), 1380–1387 (2008)
4. Correa, C.D., Ma, K.L.: The occlusion spectrum for volume visualization and classification. IEEE Trans. Vis. Comput. Graph. **15**(6), 1465–1472 (2009)
5. Correa, C.D., Ma, K.L.: Visibility driven transfer functions. In: IEEE Pacific Visualization Symposium, pp. 177–184 (2009)
6. Correa, C.D., Ma, K.L.: Visibility histograms and visibility-driven transfer functions. IEEE Trans. Vis. Comput. Graph. **17**(2), 1077–2626 (2010)
7. Duda, R.O., Ehart, P., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience, New York (2001)
8. Fukunaga, K., Larry, H.D.: The estimation of the gradient of a density function, with applications in pattern recognition. IEEE Trans. Inf. Theory **21**(1), 32–40 (1975)
9. Hadwiger, M., Fritz, L., Rezk-Salama, C., Höllt, T., Geier, G., Pabel, T.: Interactive volume exploration for feature detection and quantification in industrial CT data. IEEE Trans. Vis. Comput. Graph. **14**(6), 1507–1514 (2008)

10. Hladuvka, J., König, A., Gröller, E.: Curvature-based transfer functions for direct volume rendering. In: Proceedings of Spring Conference on Computer Graphics, pp. 58–65 (2000)
11. Hong, D., Ning, G., Zhao, T., Zhang, M., Zheng, X.: Method of normal estimation based on approximation for visualization. J. Electron. Imaging **12**(3), 470–477 (2003)
12. Huang, R., Ma, K.L.: RGVis: region growing based techniques for volume visualization. In: Proceedings of Pacific Conference on Computer Graphics and Applications, pp. 355–363 (2003)
13. Kindlmann, G.: Transfer functions in direct volume rendering: design, interface, interaction. In: Course Note of ACM SIGGRAPH (2002)
14. Kindlmann, G., Durkin, J.W.: Semi-automatic generation of transfer functions for direct volume rendering. In: Proceedings of IEEE Symposium on Volume Visualization, pp. 79–86 (1998)
15. Kindlmann, G., Whitaker, R., Tasdizen, T., Möller, T.: Curvature-based transfer functions for direct volume rendering: methods and applications. In: Proceedings of IEEE Symposium on Volume Visualization, pp. 513–520 (2003)
16. Kniss, J., Kindlmann, G., Hansen, C.: Interactive volume rendering using multi-dimensional transfer functions and direct manipulation widgets. In: Proceedings of IEEE Symposium on Volume Visualization, pp. 255–262 (2001)
17. Kniss, J., Kindlmann, G., Hansen, C.: Multi-dimensional transfer functions for interactive volume rendering. IEEE Trans. Vis. Comput. Graph. **8**(3), 270–285 (2002)
18. Lum, E.B., Ma, K.L.: Lighting transfer functions using gradient aligned sampling. In: Proceedings of IEEE Visualization, pp. 289–296 (2004)
19. Lundström, C., Ljung, P., Ynnerman, A.: Extending and simplifying transfer function design in medical volume rendering using local histograms. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp. 263–270 (2005)
20. Lundström, C., Ljung, P., Ynnerman, A.: Local histograms for design of transfer functions in direct volume rendering. IEEE Trans. Vis. Comput. Graph. **12**(6), 1570–1579 (2006)
21. Lundström, C., Ynnerman, A., Ljung, P., Persson, A., Knutsson, H.: The α-histogram: using spatial coherence to enhance histograms and transfer function design. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp. 227–234 (2006)
22. Maciejewski, R., Chen, W., Woo, I., Ebert, D.S.: Structuring feature space—a non-parametric method for volumetric transfer function generation. IEEE Trans. Vis. Comput. Graph. **15**(6), 1473–1480 (2009)
23. de Moura Pinto, F., Freitas, C.M.D.S.: Design of multi-dimensional transfer functions using dimensional reduction. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp. 131–138 (2007)
24. Nguyen, B.P., Tay, W.L., Chui, C.K., Ong, S.H.: Automatic transfer function design for volumetric data visualization using clustering on LH space. In: Proceedings of Computer Graphics International (2011)
25. Pekar, V., Wiemker, R., Hempel, D.: Fast detection of meaningful isosurfaces for volume data visualization. In: Proceedings of IEEE Visualization, pp. 223–230 (2001)
26. Praßni, J.S., Ropinski, T., Hinrichs, K.H.: 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 (2009)
27. Röttger, S., Bauer, M., Stamminger, M.: Spatialized transfer functions. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp. 271–278 (2005)
28. Tappenbeck, A., Preim, B., Dicken, V.: Distance-based transfer function design: specification methods and applications. In: Proceedings of Simulation und Visualisierung, pp. 259–274 (2006)
29. Teistler, M., Breiman, R.S., Liong, S.M., Ho, L.Y., Shahab, A., Nowinski, W.L.: Interactive definition of transfer functions in volume rendering based on image markers. Int. J. Comput. Assisted Radiol. Surg. **2**(1), 55–64 (2007)
30. Tzeng, F.Y., Ma, K.L.: A cluster-space visual interface for arbitrary dimensional classification of volume data. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp. 17–24 (2004)
31. Tzeng, F.Y., Lum, E.B., Ma, K.L.: A novel interface for higher-dimensional classification of volume data. In: Proceedings of IEEE Visualization, pp. 505–512 (2003)
32. Tzeng, F.Y., Lum, E.B., Ma, K.L.: An intelligent system approach to higher-dimensional classification of volume data. IEEE Trans. Vis. Comput. Graph. **11**(3), 273–284 (2005)
33. Šereda, P., Bartroli, A.V., Serlie, I.W.O., Gerritsen, F.A.: Visualization of boundaries in volumetric data sets using LH histograms. IEEE Trans. Vis. Comput. Graph. **12**(2), 208–218 (2006)
34. Šereda, P., Vilanova, A., Gerritsen, F.A.: Automating transfer function design for volume rendering using hierarchical clustering of material boundaries. In: Proceedings of IEEE/Eurographics Symposium on Visualization, pp. 243–250 (2006)
35. Wesarg, S., Kirschner, M.: Structure size enhanced histogram. In: Brauer, W., Meinzer, H.P., Deserno, T.M., Handels, H., Tolxdorff, T. (eds.) Bildverarbeitung für die Medizin 2009. Informatik aktuell, pp. 16–20. Springer, Berlin (2009)
36. Wesarg, S., Kirschner, M., Khan, M.F.: 2D histogram based volume visualization: combining intensity and size of anatomical structures. Int. J. Comput. Assist. Radiol. Surg. **5**(6), 655–666 (2010)
37. Zhou, F., Zhao, Y., Ma, K.: Parallel mean shift for interactive volume segmentation. Mach. Learn. Med. Imaging, 67–75 (2010)

Binh P. Nguyen received the B.Eng. (Hons.) in Information Technology and M.Sc. in Information Processing and Communications from Hanoi University of Technology, Vietnam in 2002 and 2004, respectively. He is currently a lecturer at the School of Information and Communication Technology, Hanoi University of Technology, Vietnam and working toward the Ph.D. degree in the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research interests include visualization of medical images, GPU-based algorithms, multi-core and multi-GPU computing.

Wei-Liang Tay received the B.Eng. (Hons.) in Electrical and Computer Engineering from the National University of Singapore in 2009. He is currently a Ph.D. student in the Department of Electrical and Computer Engineering at the National University of Singapore. His research interests include machine learning for computer-aided detection and diagnosis, medical image understanding, and volume rendering.

Chee-Kong Chui is currently an Assistant Professor in the Control and Mechatronics Group, Department of Mechanical Engineering, National University of Singapore, Singapore. He was the principal investigator of the Biomedical Simulation & Device Design Project at the then Institute of Bioengineering prior to pursing a Ph.D. in Biomedical Precision Engineering Lab, The University of Tokyo, Japan. His research interests include computer integrated and robot-assisted surgery, human-machine interface, medical device design, biomechanical modeling and simulation.

Sim-Heng Ong is an associate professor in the Department of Electrical and Computer Engineering and the Division of Bioengineering, National University of Singapore. He received his B.E. (Hons.) from the University of Western Australia and his Ph.D. from the University of Sydney. His major fields of interest are computer vision and biomedical image processing. He has published over 250 papers in international journals and conference proceedings.