Journal of AI and Data Mining
Vol 8, No 1, 2020, 67-82.
DOI: 10.22044/JADM.2019.7207.1854

Segmentation Assisted Object Distinction for Direct Volume Rendering

A. Azimzadeh Irani* and R. Pourgholi

School of Mathematics and Computer Science, Damghan University, Damghan, Iran.

Received 25 June 2018; Revised 12 August 2018; Accepted 07 April 2019
**Corresponding author: a.azimzadeh@du.ac.ir (A. Azimzadeh Irani).*

Abstract
Ray Casting is a direct volume rendering technique for visualizing 3D arrays of sampled data. It has vital applications in medical and biological imaging. Nevertheless, it is inherently open to the cluttered classification results. It suffers from the overlapping transfer function values and lacks a sufficiently powerful voxel parsing mechanism for object distinction. In this work, we propose an image processing-based approach towards the enhancing ray casting technique for the object distinction process. The rendering mode is modified to accommodate masking information generated by a K-means-based hybrid segmentation algorithm. An effective set of image processing technique is creatively employed in the construction of a generic segmentation system capable of generating object membership information.

Keywords: *Hybrid Image Segmentation, Volume Rendering, Enhanced Visualization Effect.*

1. Introduction

Given the great dependency of human analysis and judgment abilities on realistic and sensible visualization, appropriate volume rendering techniques for effective visualization may be deemed fundamental. Most of the currently available direct volume rendering methods such as ray casting, shear-warp, splatting, and texture mapping employ basic transfer function(s) for object distinction. However, a major drawback of using these methods is that all voxels of a volumetric dataset are treated in an identical manner without using any priori information that specifies object membership on a per-voxel basis [1]. Inability to properly distinguish among multiple objects of interest solely based on transfer function is often the case. Segmentation is a formidable approach for handling this problem. It infers object membership information for each object of interest, yielding a tag for each voxel in the volume. In this context, explicit and implicit designs have been introduced by different research works. The former is concerned with technicality of adapting readily available object membership information into a particular rendering mode or combination of rendering modes, while the latter summons its own set of object membership information via a carefully designed segmentation system, giving peculiar emphasis on integration issues. So far, consideration of an integrated visualization framework that could enable the adaption of a viable segmentation design into a particular rendering architecture has been an open domain for creative solutions. The literature present on segmentation-based volume visualization suggests significantly exclusive approaches, yielding a gap for generic solutions with reasonable outcome for handy visualization tasks.

1.2 Problem statement

The first question that comes to the mind of a person who intends to enhance or improve a particular technique may be what the shortcomings of that technique are. Only by knowing the shortcomings or weaknesses of a technique, one could propose a relevant solution. In order to highlight or demonstrate the overall problem that we are going to solve within this work, we have divided the weakness of ray casting technique into three main groups, which are cluttered classification, cumbersome transfer function design, and ambiguity/wasted resources. In ray casting, classification is basic. It is directly based on the raw data. A distinct range of values or, in

other words, intensity is assigned to each particular object. However, it often occurs that an object does not entirely fall within its pre-defined range. Thus occupying or overlapping other object(s) ranges that leads to a poor classification. Since the transfer function design is based on classification, a poor classification leads to an imprecise transfer function design. On the left side of figure 1, we have air, fat, soft tissue, and bone as the four overlapping objects, and on the right side of the figure, we have their relevant transfer function design that evidently mixes the relevant colors and opacities of distinct objects. Performing consecutive interpolations and compositions throughout the volume, as a whole, without a sense of direction not only consumes graphic resources but also reduces the level of details for the objects we are actually interested in. As it may be noticed in figure 2, in ray casting, an ambiguous situation occurs when four elemental vector values at each cubical vertex are used to carry out interpolation. Interpolating all of these parameters at once leads to serious visual artifacts.

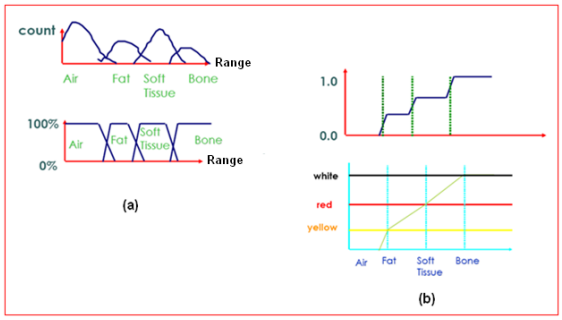

Figure 1. (a) Overlapping classification (b) Imprecise transfer function design.

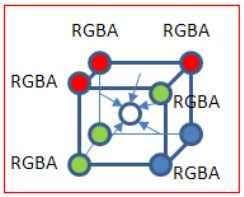

Figure 2. An ambiguous interpolation result originates from multiple vector parameters.

1.3 Objectives

The main objective of this work is to achieve enhanced visualization effect through amendment of ray casting based direct volume rendering. In general, direct volume rendering process involves classification, transfer function formulation and rendering as its three respective and interdependent phases. Thus, in order to improve a particular phase one may need to modify its prerequisite phase(s). Our main objective could therefore be broken into three parts. First, adaptation of a segmentation algorithm instead of the basic classification module of ray casting. Second, establishment of representative transfer functions that properly manifest classification. Third, adjustment of the rendering pipeline for optimum implementation of transfer functions.

2. Literature review

Hessian-based line filters and fast marching active contour were used to segment coronary arteries and pericardial cavity respectively [2]. Explicit segmentation information was utilized as a main contributing factor to the rendering pipeline. Each object is separated by an ID at voxel level [3]. Two level volume rendering approach allows combination of local per object compositing mode with global direct volume rendering (DVR) [3]. For instance, non-photorealistic rendering and maximum intensity projection (MIP) could be used in combination with DVR to enable higher quality results [3]. In the context of GPU based volume rendering where no actual ray exists, conceptual ray should be able to combine the contributions of different compositing modes. Local and global buffers are used to track the current composition mode for each pixel. The status of each pixel is switched between local and global buffers [3]. Figure 3 demonstrates the idea. As can be noticed, MIP, NPR and DVR are used for different objects along a global pass. Thus, each object utilizes the mode most suitable for it. A semi-automatic volume segmentation algorithm (skeleton-cut) based on skeleton topology is presented. Complex structures could be simplified by appropriately representative descriptors [4]. Skeletons are utilized to provide meaningful clues for object distinction. Skeletons are identified by initially extracting coarse boundaries and then Euclidean transformation between the foreground and background voxels [4]. Once identified, skeletons are connected to their neighbors to form a graph [4]. The weights among the nodes are assigned based on intersection area of the cells, intensity means and radii of inscribed spheres [4]. The min-cut / max-flow operations induce the graph outcome [4].

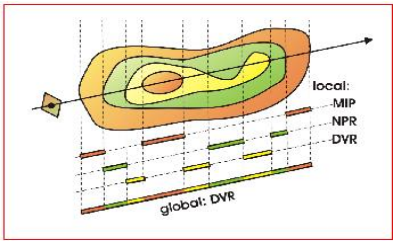

Figure 3. Two conceptual levels of volume rendering [3].

Currently, design of transfer functions that successfully associate data values with visual properties is a difficult and "non-intuitive" task [5]. DVR is naturally open to ambiguous classification and may require segmentation based masking to get occlusion free view of the desired objects [5]. There is however, a risk of erroneous masking or unintentional assignment of background to object (s) of interest. This may lead to fatal misinterpretations, especially in sensitive medical fields [5]. For instance, an error may rise in volumetric measurement of a tumor. An uncertainty (risk) assessment of volume segmentation is performed and a formidable risk reduction and control system is proposed. The proposed system demonstrates a close interplay with user [5]. Its four main phases are probabilistic random walker segmentation to produce segmentation results with estimable uncertainty level, risk analysis, guiding user to the regions which are possibly prone to error and identification of error by user and insertion of desired fixing to initial segmentation parameters [5]. Nero Trace is presented as a semiautomatic volume segmentation and visualization system for neural processing of nervous organ(s) [6].It consists of preprocessing, multiphase level set segmentation and 3D tracking and a special ellipse based rendering method for electron microscopy (EM) data visualization. Prior to segmentation, volume rendering is used to search the input volume (volume constructed from raw EM data) for a region of interest (ROI) [6]. User can select center of ROI (desired set of neural cells) on an arbitrary 2D plane using volumetric view [6]. There have been Many attempts (ranging from simple to advance) to improve transfer design with no or little reference to segmentation. For instance, in a simple case, trial and error methods are introduced to generate numerous transfer functions and perform rendering based on each of them. Then pick the one with most suitable rendering outcome [7]. Advance cases however introduce the multidimensional concept of transfer function design [8], which involves extracting one or more representative features from dataset and adding them to the singular feature space of a one dimensional transfer function.

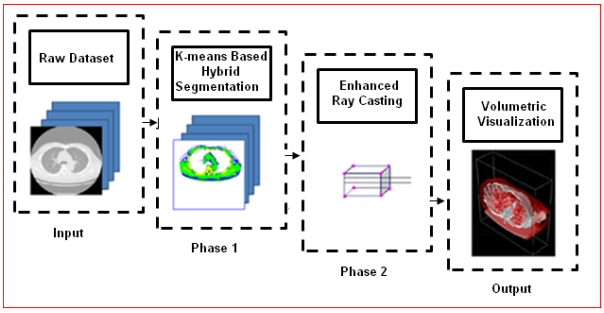

Figure 4. An overall view of the integrated visualization design.

2.1 An integrated visualization design

So far, the weaknesses of ray casting, in particular, and other direct volume rendering techniques, in general, were identified. As it may be noticed in figure 4, this section provides an integrated framework, based on which the aforementioned weaknesses could be handled in a creative manner. In Phase 1 of figure 4, K-means-based hybrid segmentation algorithm is performed on our input images in order to extract objects or, in other words, produce object membership information. The goal of this phase is to boost the ray casting's classification ability. As mentioned earlier, ray casting originally possesses overlapping or, in other words, inferior classification ability. In Phase 2 of figure 4, the object membership information generated in Phase 1 is adapted into an enhanced ray casting architecture via ID tags. Table 1 compares our proposed solution with the standard ray casting technique.

Table 1. Comparison between ray casting and this work.

Distinctive features	**Ray casting** [9]	**This work**
Sampling rate	arbitrary	arbitrary
Sampling nature	point	point
Interpolation	tri-linear	N. N. approx / optimized tri-linear
Kernel	linear	linear
Exceptionality	-	multiple rendering modes
Selected voxels	all	segmented
Acceleration	early ray termination	ID-based space skipping

As it can be noticed from the interpolation field in table 1, none of the other direct volume rendering techniques and particularly standard ray casting has the option of using more than one interpolation method because they lack the means for separating complex voxels from simple voxels. Here, by complex voxels, we mean voxels that represent

more than one object, and by simple voxels, we mean voxels that represent only one object. However, since our proposed rendering design uses ID tags, it can easily distinguish between the simple and complex voxels, and therefore, employ a suitable interpolation method for each of them. Furthermore, in contrast to standard tri-linear interpolation used by ray casting, the tri-linear interpolation method is optimized so that it can avoid edge ambiguity, particularly when interpolation inputs are from more than one object. Exceptionality field refers to exceptional characteristics of each technique. These exceptional characteristics could be a strength and advantage as is the case for our proposed solution. In the case of our proposed solution multiple local rendering traversals that are embedded within a single global traversal enable two important tasks, which are localization of transfer function design and depth-based manipulation of interpolation and composition operations. Acceleration refers to the mechanism a technique uses to speed-up its overall rendering process. In standard ray casting, early ray termination is used as a way of controlling perceptual details. It is used to stop composition at a particular depth before the ray reaches the end of volume. In our proposed solution, the notion of ID tags focuses on the rendering process. Thus pushing non-tagged (anonymous) regions of the volume to the context or, in other words, out of rendering process. There are other fields in table 1 such as sampling rate, kernel, and sampling nature. Sampling rate refers to the interval based on which volume is sampled. More frequent sampling leads to smoother outcome. Sampling interval usually determines the number of interpolations we are required to perform. Most direct volume rendering technique kernels are linear. This work particularly employs tri-linear interpolation and nearest neighbor approximation, both of which have a linear nature. However, splatting technique for instance employs Gaussian interpolation due to radial complexity of its kernel. Sampling nature depends on the kernel and interpolation characteristics of a particular technique.

3. Proposed solution

In this work, we propose an image processing-based approach towards enhancing ray casting the technique object distinction process. The ray casting architecture is modified to accommodate object membership information generated by a K-means-based hybrid segmentation algorithm. Object membership information is assigned to cubical vertices in the form of ID tags. An intra-object buffer is devised and coordinated with inter-object buffer, allowing the otherwise global rendering module to embed multiple local (secondary) rendering processes. A local rendering process adds two advantageous aspects to the global rendering module: first, depth oriented manipulation of interpolation and composition operations that lead to freedom of interpolation method of choice based on the number of available objects in various volumetric depths, improvement of LOD (level of details) for desired objects, and reduced number of required mathematical computations; Second, localization of transfer function design that enables the utilization of binary (non-overlapping) transfer functions for color and opacity assignment. A set of image processing techniques are creatively employed in the design of K-means-based hybrid segmentation algorithm. Pre-processing methods such as high pass/low pass filters and histogram equalization are optionally used for noise removal and harmony rectification. An unsupervised neural network is used to initialize cluster centers and improve the clustering accuracy. Few rounds of K-means clustering are performed to identify preliminary groups. Fisher discriminant ratio is used to convert preliminary groups into optimum segments representative of inherent color and formation characteristics. Edge detection is used to exempt important boundary/pattern information from irrelevant recombination. Recombination and epsilon tolerance factor are used for spatial tuning.

3.1 K-means-based hybrid segmentation design

Figure 5 provides a flow chart representative of our segmentation architecture. A sub-numbering scheme is used in order to indicate the connection between figure 5 and phase 1 of figure 4. Following the flow chart, we initially have the input image that could include either a gray scale or color image. Next, we have Phase 1.1, which is an optional phase. It has a role to play only when the input data is low in quality (ex. has considerable noise or overall intensity imbalance). Its job is to improve the resolution of important features in a low-quality input image. Such improvement helps toward increasing the overall segmentation precision. As it may be evident in figure 5, the input of each phase is dependent on its pre-requisite phase. Therefore, the output of phase 1.1 is passed to phase 1.2 as input.

In phase 1.2, a neural network is used to generate cluster centers. K-means clustering, which is performed in Phase 1.3, requires initial cluster centers or seed points to carry out clustering.

However, it is known that allowing K-means to generate its own initial cluster centers at random

without any sense of direction or rational may lead to local optima convergences. To be more specific, by local optima converge, we mean the clustering result that is not representative of the entire image. In order to avoid local optima convergence and ensure a globally optimum clustering, we used a neural network that spread the initial cluster centers based on the relevant color and spatial properties of the image at hand.

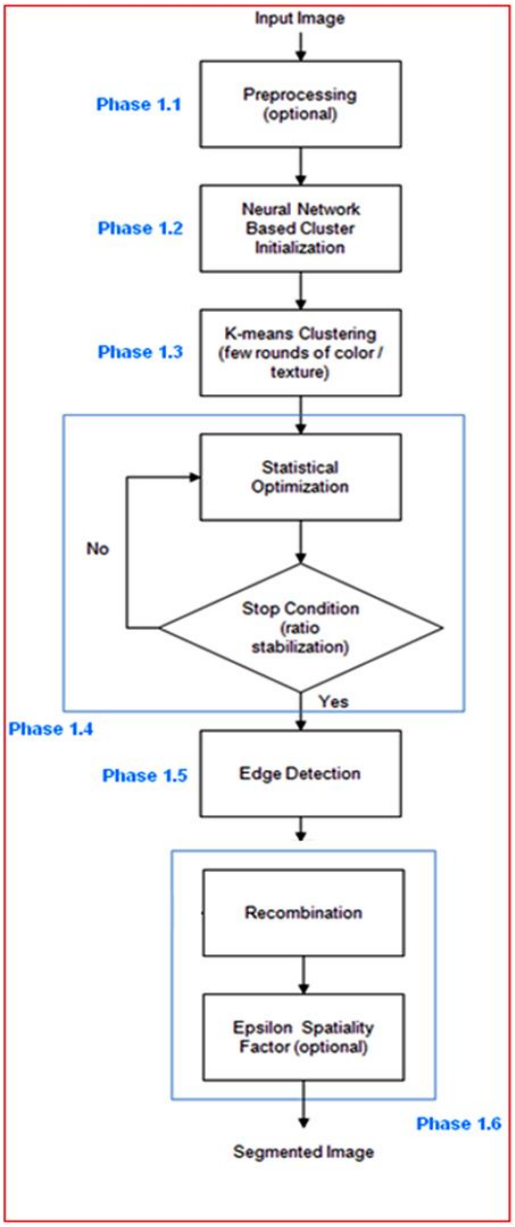

Figure 5. K-means-based hybrid segmentation design.

In phase 1.3, a few rounds of K-means clustering is performed in order to generate only coarse or preliminary clusters. The reason for why only few iterative rounds of K-means algorithm is performed is that we have dedicated the refinement task to a superior statistical algorithm (particularly, fisher discriminant ratio) that could produce a much better refinement than K-means algorithm only in few rounds.

In phase 1.4, the goal is to avoid formation of too large or too small clusters via statistical refinement. The preliminary clusters produced in phase 1.3 are refined such that both intra-homogeneity (the homogeneity inside each cluster) and inter-heterogeneity (the heterogeneity of each cluster with other clusters) increase. Figure 6 demonstrates the relationship between phases 1.3 and 1.4. It briefly highlights the evolution of clusters from coarse to fine.

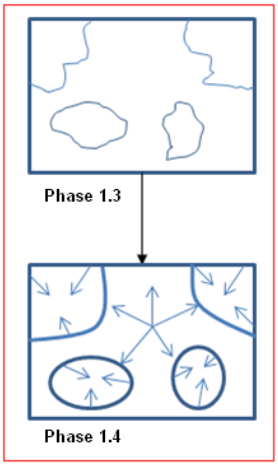

Figure 6. Interrelationship between phases 1.3 and 1.4.

In phase 1.5, edge detection is performed in order to improve the precision of Phase 1.6's recombination task. During recombination, the spatial neighborhood of each pixel is searched in order to ensure the homogeneity with respect to neighborhood majority. If a particular pixel, for instance, has a different value from the majority of its neighbors, its value should change to the value of the majority unless it is an edge pixel. It is only natural for an edge pixel to have a different value from its neighbors because edges are usually defined as sharp turns among objects. In phase 1.6, apart from recombination, which was explained above, we have an epsilon spatiality factor that is basically a tolerance factor. Its job is to ensure a logical spatial distance among objects. For instance, if we have mistakenly classified two objects that have similar color and texture properties but irrelevant spatial properties under the same group, we can make a correction via the epsilon factor.

3.1.1 Pre-processing

Low-quality images are the specific target of this optional phase. Prior to the actual segmentation

operations such as histogram, equalization and noise removal via readily available filters could take place in order to highlight the otherwise vague but important features. As it may be noticed from an instance in Chapter four, an appropriate early amendment such as histogram equalization could boost feature resolution and ensure an enhanced segmentation result.

3.1.2 Unsupervised neural map for cluster center initialization

Unguided cluster center initializations often mislead the overall clustering task. Inability to determine the spread and number of possible clusters may often trigger imprecise and locally optimum solutions. An enhanced unsupervised neural network is used to help identify the required numbers and spatial locations of potential cluster centers. Automation and approximation are the two important reasons for why a self-contained neural network was used. As compared to the back-propagation technique, in which manual feedback-oriented training rounds are required to reach a reasonably accurate result, a self-organized network could impulsively produce useful approximation suitable for cluster center initializations.

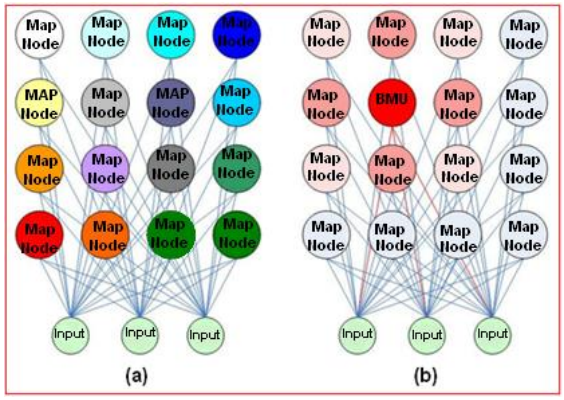

Figure 7. (a) Network initialization (b) Neighborhood adjustments.

Figure 7 demonstrates a typical 4×4 self-contained network with three inputs. Here, the three RGB color bands form the input values (gray level values could also be represented by elemental blend). The network is established such that each map node includes three feature vectors and is only connected to the input nodes. Prior to training, the range of RGB colors available within the image is obtained. As indicated in figure 8, in order to define a range, the smallest (lower bound) and largest (upper bound) color values available within the image are found and their difference is calculated. Then an interval is calculated. Feature vectors for each map node are initialized by traversing from a lower bound to an upper bound based on interval (please take note of figure 7 part (a)). To make the pseudo-code (figure 8) clear, it is explained from top to bottom. First, the product of map width and length is calculated in order to obtain the total number nodes we need to initialize. Next, the smallest and largest color values available within the image at hand are calculated. Next, an interval by evenly dividing the available color values among map nodes is obtained. Last, initialize each map node by assigning color values based on the previously obtained interval.

```
PROCEDURE  NeuralMapInitialization:

CalculateInterval ( ) {

NumberofMAPNodes = multiply (map_width,
map_length)

SmallestAvailableColorValue = get (min (R, G, B))

LargestAvailableColorValue = get (max(R, G, B))

Range = subtract (LargestAvailableColorValue,
SmallestAvailableColorValue)

Interval = divide (Range, subtract (Numberof
MapNodes, 1)
}
initialize (CurrentMapNodeValue,
SmallestAvailableColorValue)

For each (MapNode) {CurrentMapNodeValue =
CurrentMapNodeValue + Interval  }
```

Figure 8. Pseudo-code of an interval computation for map node initialization.

After initialization, training starts by randomly selecting pixels from all over the image, storing their spatial positions and passing them to input nodes one by one in order to decide the best match unit. BMU is the map node that is most similar to the input values as compared to other nodes. As it can be seen in figure 7 part (b), once the best matching node is selected, its feature vectors and neighborhood are adjusted to most closely resemble the input values. The amounts of adjustments are determined by the learning rate and neighborhood size factors in two stages. In the first stage, the learning rate begins at 0.9 and gradually reduces to 0.1, and the neighborhood size begins by half of the map size and gradually decreases to 1. The second stage is more of a refinement. At this stage, the feature vectors of the nodes are stabilized and would further be refined by a learning rate of 0.01. The neighborhood size remains at 1, meaning that only the BMU node is adjusted. For ease of calculations, the feature vectors are normalized

from [0, 255] to [-1, 1]. Upon the end of training and convergence numbers, locations and values that cluster centers should possess are approximated. Equation 1 formulates a model for node adjustment.

$$w_v(s+1) = w_v(s) + \theta(u,v,s)\alpha(s)\big(D(t) - w_v(s)\big) \quad (1)$$

Wv, s, u, v, t, $\alpha(s)$, and $D(t)$ refer to the weight vector, step index, BMU index for $D(t)$, neurons, training sample index, monotonically decreasing learning coefficient, and input vector, respectively. $\theta(u,v,s)$ is the neighborhood function. It is calculated based on the lattice distance between BMU (u) and neuron v.

3.1.3 K-means clustering

K-means is a statistical clustering algorithm. Equation 2 formulates how K-means commonly works. x, m, d^2, and min refer to the data point (pixel), average of a data points set (cluster center), Euclidian distance, and minimum value, respectively.

$$\frac{1}{n}\sum_{i=1}^{n}\left[min\ d^2\ (x_i, m_j)\right] \quad (2)$$

A homogenous cluster is identified when a particular cluster center possesses the minimum distance (mean squared error) to several data points as compared to other cluster centers. In this work, few rounds of K-means clustering were used to produce the preliminary classification results or, in other words, elementary clusters. The initial cluster centers required by K-means algorithm are provided by Phase 1.2. At this stage, to avoid the ambiguity caused by compromise over features, the dimensions are limited to merely three color bands and a diagonal busyness factor (DF). DF is used to probe texture around each pixel. Texture is an intrinsic property of virtually all surfaces. It relays senses such as regularity, coarseness, fitness, directionality, and granularity. Textures, in general, grant a directional sense to the spatial tendency of image intensities. It helps in clarifying ambiguous situations. For instance, hatches of a brownish chair could be used to tell it apart from its brown background. As it may be noticed in Equation 3, DF is calculated as the sum of color differences among the central and diagonal pixels. C and x denote the central and diagonal pixels, respectively. Figure 9 demonstrates the diagonal directions around a central pixel.

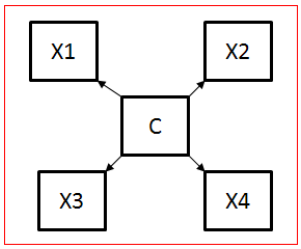

Figure 9. Diagonal directions including top left, top right, bottom left, and bottom right.

$$DF = \sum_{i=1}^{4} |C - x_i| \quad (3)$$

3.1.4 Statistical optimization

Fisher discriminant ratio is used for the refinement of preliminary and coarse clusters produced in the previous phase. K-means clustering originally requires many iterative rounds for a valid clustering outcome. However, a Fisher discriminant ratio could perform a stable clustering using only few iterative rounds. In contrast to K-means, a Fisher discriminant ratio enables uniformity across the entire space. A balanced distribution of homogeneous clusters is obtained by including variances within ratio calculations. Equation 4 formulates how Fisher discriminant ratio works. $Jp, m, and\ v$ refer to ratio, mean, and variance, respectively. The discriminant ratio is calculated as the sum of differences in means over sum of variances. Larger differences in means or smaller variances obviously lead to a higher ratio.

$$J_p = \frac{\sum_{i=1}^{k}\sum_{j>i}^{k}\left(m_{ip} - m_{jp}\right)^T\left(m_{ip} - m_{jp}\right)}{\left(\sum_{i=1}^{k} v_{ip}^T v_{ip}\right)} \quad (4)$$

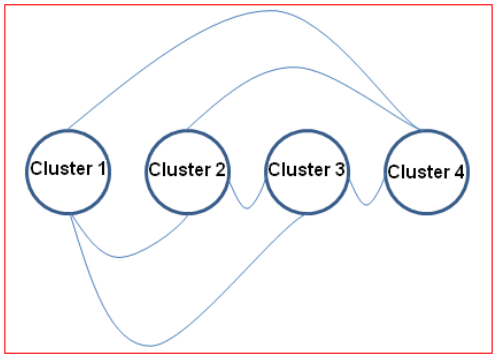

Figure 10. Graphical representation of ratio's nominator.

In an iterative round, each pixel (data point) is temporarily removed from its home cluster and assigned to each one of the other clusters one by

one. Every time a temporary assignment occurs, the differences in means and variances are renewed and the discriminant ratio is recalculated. Figure 10 indicates the routes among the four clusters required for calculation of differences in means. A pixel is permanently detached and reassigned to another cluster only if such reassignment causes an increase in ratio, thus promoting a larger inter-cluster distance and a smaller intra-cluster distance. Statistical optimization ends when ratio alteration becomes ignorable.

3.1.5 Edge detection and analysis

In this section, edge treatment is considered as a prerequisite to recombination (performed in Section 3.1.6). It directs the spatial recombination task by discerning potential object boundaries that may be undermined during spatial recombination if not specifically exempted. The sobel algorithm is used to identify and filter important edge pixels. Long and continues edges are extracted, while short and discrete edges are ignored. In the cases where there are multiple discrete and short but closely adjacent edges, an aggregation operation is performed in order to produce a wholesome and continues edge. As it can be noticed from Equations 5 and 6, two 3 × 3 masking kernels are employed to approximate the derivatives along the horizontal and vertical directions. Throughout the image masking, kernels perform convolution operations and highlight the edge pixels.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \times A \tag{5}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & 2 & 1 \end{bmatrix} \times A \tag{6}$$

Horizontal and vertical masks are denoted by matrices at G_x and G_y, respectively. The multiplication operation with A represents convolution. The values used in masking kernels are not limited to Equations 5 and 6. They could be changed depending on the image at hand. Diverse images may require different filter values.

Equations 7 and 8 formulate gradient magnitude and gradient direction calculations in the sobel algorithm. The parameters such as G_x and G_y are derived from Equations 5 and 6.

$$G = \sqrt{G_x^2 + G_y^2} \tag{7}$$

$$\theta = arctan\left(\frac{G_y}{G_x}\right) \tag{8}$$

3.1.6 Spatial adjustment

Often few irrelevant fragments remain within clusters even upon multiple rounds of refinement. In order to reduce these artifacts, spatial recombination is employed. To carry out recombination, the image is traversed from top to bottom, and each pixel is scanned in eight connected directions. For each pixel, the cluster to which the majority of pixels in the eight connected neighborhood belong is determined. If the central pixel (pixel whose neighborhood is being scanned) belongs to the same cluster as the majority of its eight connected neighborhood pixels, its cluster will not change; otherwise, its cluster will change to the cluster that the majority of its eight connected neighborhood pixels belong. As it can be noticed in figure 11 part (a), the majority of pixels around the central pixel are classified under cluster number 4. Consequently, the central pixel's cluster is changed to 4 (figure 11 part (b)). Given the nature of color and texture-based clustering, the objects with similar color and pattern characteristics but dissimilar spatial properties may be classified under the same group, rendering them impossible to be distinguished. To tackle this problem, an optional tolerance-oriented spatiality factor referred to as epsilon is devised. If the sum of spatial distances of pixels belonging to a particular cluster is greater than the epsilon value, then those pixels that are further away are identified, and if their density is great enough, they are grouped under a different cluster.

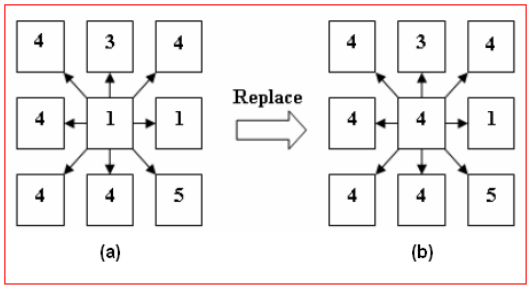

Figure 11. (a) A 3×3 neighborhood before recombination (b) After recombination.

3.2 An enhanced rendering architecture

Figure 12 provides a flow chart, representative of this work's rendering architecture. A sub-numbering scheme is used in order to indicate the connection between figure 12 and phase 2 of figure 4.

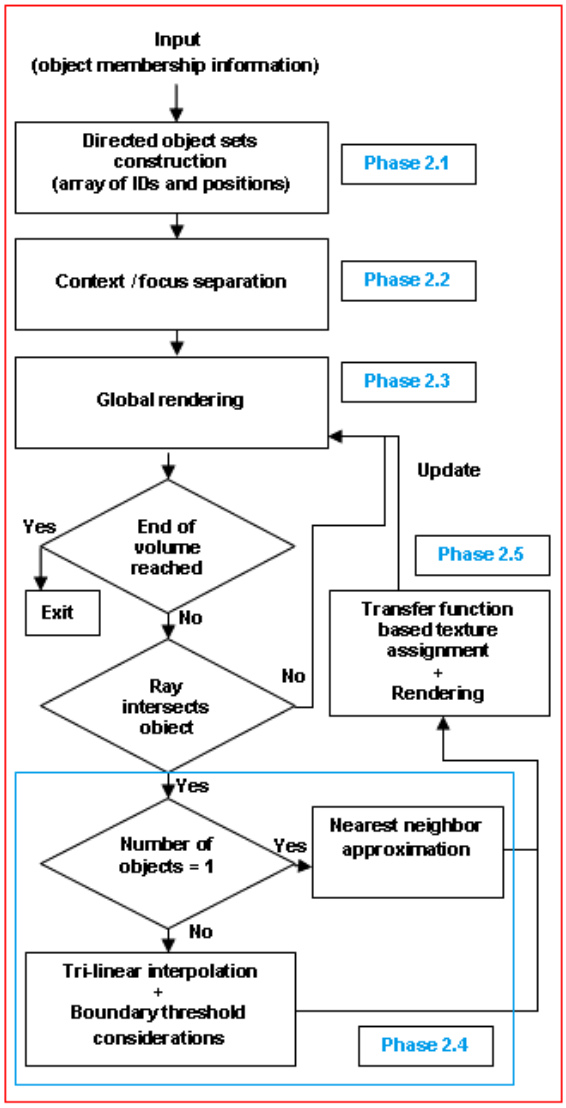

Figure 12. An enhanced rendering architecture.

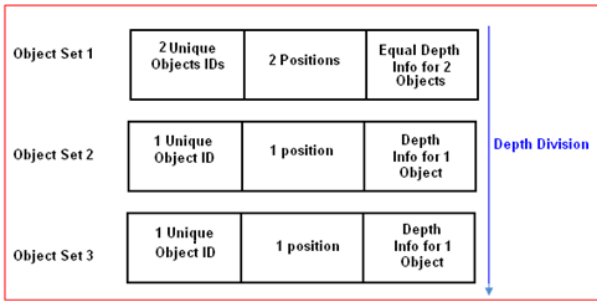

Figure 13. A typical object set array.

Following the flow chart, in phase 2.1, the object membership information generated by the K-means-based hybrid segmentation algorithm is used to produce a unique ID tag for each particular object. An ID tag is accompanied by an array of object depth and position. Depending on the axis along which projection takes place, the depth properties are subject to change. However, volumetric position is indifferent to any changes. As it may be evident in figure 12, the input of each phase is dependent on its prerequisite phase. Therefore, the output of phase 2.1 is passed to phase 2.2 as the input. In phase 2.2, a rendering focus is made possible by assigning the ID tags produced in phase 2.1 to the desired cubical voxels. The non-tagged regions are set to anonymous and moved to the context. In phase 2.3, global rendering is performed in a manner different from the original ray casting technique. In ray casting, sampling and composition are carried out from the very beginning to the end at a constant pace. However, in our proposed method, global rendering is simplified. The global module starts traversal, and does not perform any composition until it reaches a particular volumetric depth, where there is an object of interest. Upon reaching an object, global module is suspended, and instead, a local module is activated. The local module's job is to perform the interpolation and composition tasks and generate an interim rendering result. Once the local module is finished, global module takes over again, and by performing only a single composition operation, incorporates the interim rendering result into its buffer. Thus the above-mentioned loop continues until the global module reaches the end of volume. In phase 2.4, we demonstrate how an interpolation task is handled within the local rendering process. The choice of the interpolation method depends on the number of objects available. If there is only one object, then the nearest neighbor approximation method is used. The nearest neighbor approximation is a primitive method and does not require calculations. It merely picks the sample value from any of the vertices of the representative cubical voxel(s). Now, if there is more than one object, then tri-linear interpolation along with threshold considerations form a mechanism for distinctive sample value generation. In standard ray casting, tri-liner interpolation is performed after texture assignment, which leads to vector-based ambiguity. However, in our proposed solution, tri-linear interpolation is performed using IDs prior to texture assignment that in combination with threshold considerations lead to clear and distinctive results.

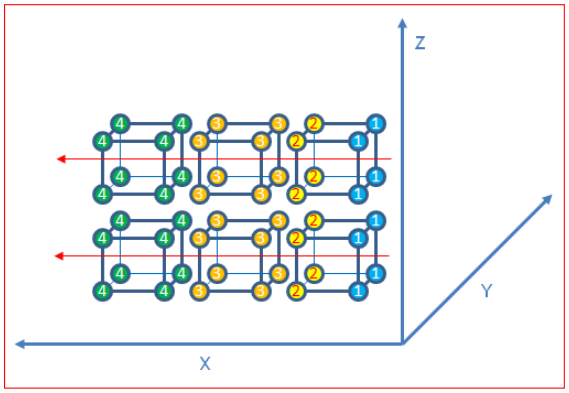

Figure 14. IDs 1 and 2 represent two objects at the same depth, while IDs 3 and 4 represent two objects at different depths.

In phase 2.5, the texture assignment task within the local rendering process is demonstrated. Texture assignment takes place based on a one to one lookup table. The lookup table is actually a simplified and precise transfer function that has come about thanks to ID tags. A sample uses its unique ID to search for the lookup table for a relevant texture (color and opacity) value.

3.2.1 Directed objects set construction

Directed object set construction in its brief and simple form refers to building an array of voxels representing an object. It is directed because the position and membership ID of voxels are determined at vertex level on each 2D slab by a K-means-based hybrid segmentation system (please refer to Section 3.1). As demonstrated in figure 13, an object set should at least include an array of membership ID, position, and depth along the viewing plane. Depth information is crucial in identifying the order with respect to which an object is rendered as compared to other objects in volumetric space. For instance, if two objects are equal in depth, they stand in the same place on the rendering queue; otherwise, the object in front would have the primitive turn in a typical front to back composition. Figure 14 demonstrates a case in which the viewing plane is along the X axis, and composition is in front to back order. Following the ray traversal (depicted by crossing arrows), objects 1 and 2 coexist at the same depth, while objects 3 and 4 are located at different depths. Objects 1 and 2 should, therefore, be grouped under the same object set array, while objects 3 and 4 should each be grouped under different arrays.

3.2.2 Focus/context separation

Inspired with the empty space skipping technique, an effective solution is proposed. Directed or guided rendering could accomplish focus on the desired regions of the volume dataset by assigning IDs only to preferred voxels, leaving the remaining/contextual voxels unassigned. Transfer function texture assignment is denied to anonymous set of voxels. In other words, unwanted areas are removed, assisting effective management of LOD required for fine and correct visualization results. The overall interpolation and composition efficiency are also improved since fewer voxels are naturally less exhaustive.

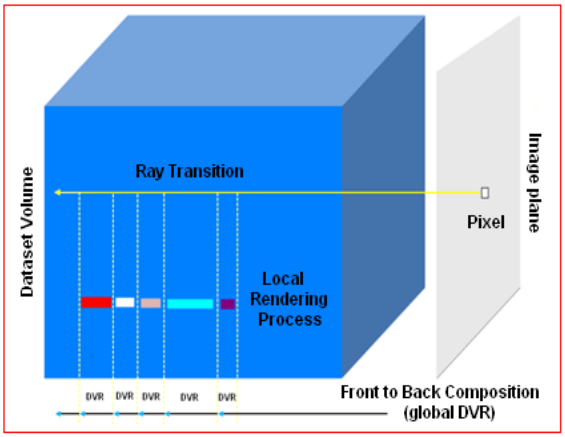

Figure 15. Double rendering design.

3.2.3 Global composition mode

Ray casting is a direct volume rendering technique. As it may be noticed in figure 15, ray casting is broken into a global traversal and local traversals as two complementary rendering components. Starting by a global traversal, as the casted ray traverses the volume, it updates (renders) the projection plane buffer (global buffer) only after passing through an object and not during the intersection course. Instead, intra-object rendering is handled locally. Occurrence of an intersection is determined by the depth at which the ray is traversing and whether there are any objects available at this depth on the object set list. Upon intersecting an object, a secondary but complete rendering process is initiated; this time is given a temporary local buffer. Given the ID(s) of the intersected object(s), a relevant interpolation solution is carried out (please refer to Section 3.2.4), and the local buffer is updated. The contents of the local buffer are then summed into the projection plane buffer as part of the overall composition mode.

3.2.4 Interpolation design and high resolution boundaries

First, selection of an appropriate interpolation technique depends on the number of objects a local ray casting process ought to render. If there is only

one object, no ambiguity could be imposed by other objects, and, therefore, a fast nearest neighbor approximation should be used to yield an unsurprisingly precise sample value. However, if there are more than one objects, then a tri-linear interpolation is used. Second, interpolation takes place based on object IDs rather than RGBA values. This is to reduce errors raised from blended color bands. IDs provide a single representative value for each vertex, while RGBA is an elemental factor of four values per vertex that could lead to an obscured interpolation outcome. Therefore, interpolation is initially performed in terms of IDs and only then mapped to relevant textures for rendering. Third, having more than one object at the same depth necessitates tri-linear interpolation among two or more objects. Combination among the boundaries' voxels and reduced visualization resolution may be inevitable in this situation. A threshold value solution could be set to separate two object borders clearly such that the mixed points in between are reassigned to either one of the objects. However, extending this solution to three or more objects is not readily possible because one could not set a single threshold value for iso-separation of more than two object IDs. To resolve this issue, a down scaling mechanism is used to produce a universal threshold. As it can be noticed in figure 16, all IDs could be normalized to the [0, 1] range. The universal threshold could then be set as 0.5, and voxels above and below threshold could be distinctively reassigned to relevant IDs. Equation 9 provides a typical normalization formulation. In a case where we use the values given in figure 16 as the Equation 9 parameters, $Max, Min, a, b, and\ x$ refer to the actual upper limit (7), actual lower limit (5), lowest normalized value (0), highest normalized value (1), and actual value that has to be normalized (5.4, 5.8, and so on), respectively.

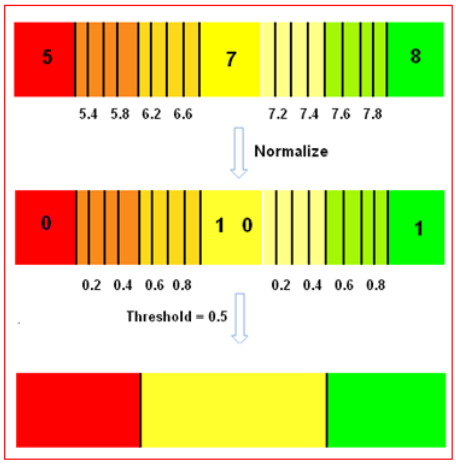

Figure 16. Universal threshold composition.

$$f(x) = \frac{(b-a)(x-Min)}{Max-Min} + a \tag{9}$$

3.2.5 Texture assignment and local rendering

The standard volume rendering techniques use global and single dimensional transfer functions for the object distinction process. However, due to the existing complexity and twist within volumetric datasets, even global and multi-dimensional (dimension refers to distinguishing factors) transfer functions are not entirely enough for a proper voxel classification. This work develops the notion of local and singular transfer functions. A lookup table is constructed to assign the texture values (color and opacity) to interpolated points (IDs) for the local rendering process prior to global composition. As it can be noticed in figure 17, there is a one to one relationship between ID and texture such that for each group of voxels that form an object, a local transfer function is allocated.

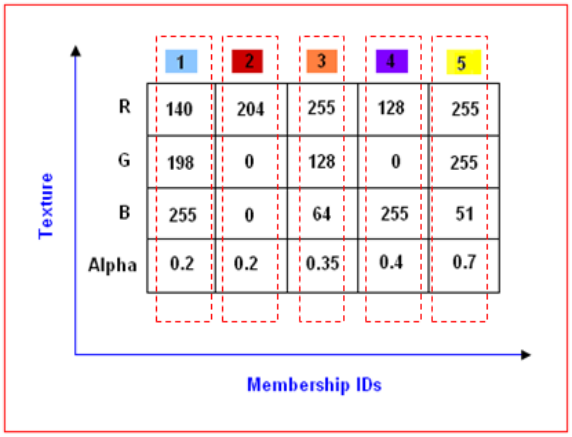

Figure 17. Texture assignment table. Each number and its respective dashed rectangle represent an object ID and its relevant RGBA values.

4. Experimental results

In this section, two experimental cases are presented, one signifying the segmentation solution of this work, and the other illustrating the combined solution.

4.1 Generic segmentation case

This work offers rather a handy generic segmentation solution that could be used independently. To validate the segmentation outcome, a comparative demonstration is provided.

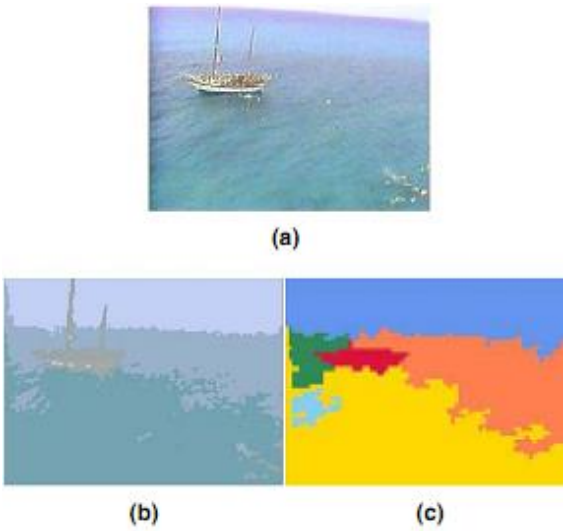

Figure 18. (a) Original image (b) This work's segmentation (c) Reference 10's segmentation.

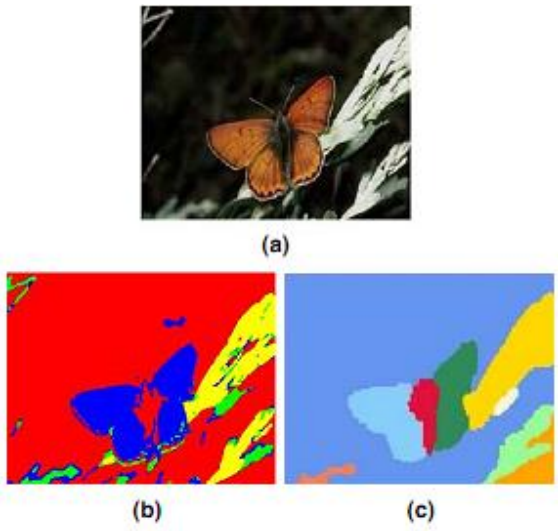

Figure 19. (a) Original image (b) This work's segmentation (c) Reference 10's segmentation.

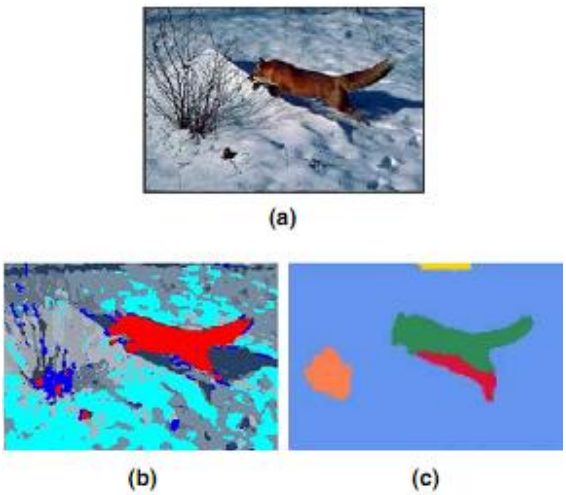

Figure 20. (a) Original image (b) This work's segmentation (c) Reference 10's segmentation.

4.2 Chest and abdomen – pelvis case

This sub-section allows a comparison between the solution proposed in this work and the standard ray casting technique. Figure 21 presents segmentation of chest and lung. Here, chest is pseudo-colored in light blue, lung in red, surrounding tissue in yellow, air way tissue in green, and skin in dark blue. A visual indication of neural map construction for cluster center initialization is provided as well.

Figure 22 deems pre-processing as an important preliminary step for an effective segmentation. It infers how ineffective could the segmentation result for low-quality slabs turn out to be if no pre-processing is performed. Figure 22 parts (a) and (c) use histograms to clarify this claim by comparing the harmonic balance of intensities with respect to the pixel density. The horizontal axis refers to intensities (small bins correspond to larger intensity intervals) and the vertical axis refers to pixel congestion.

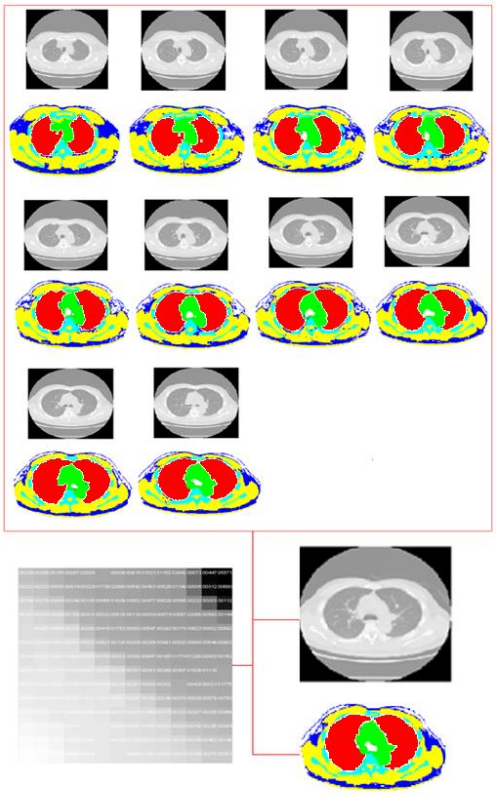

Figure 21. Chest and lung gray level slabs and their corresponding segmentations. A representative segmentation along with its 14 × 14 neural map.

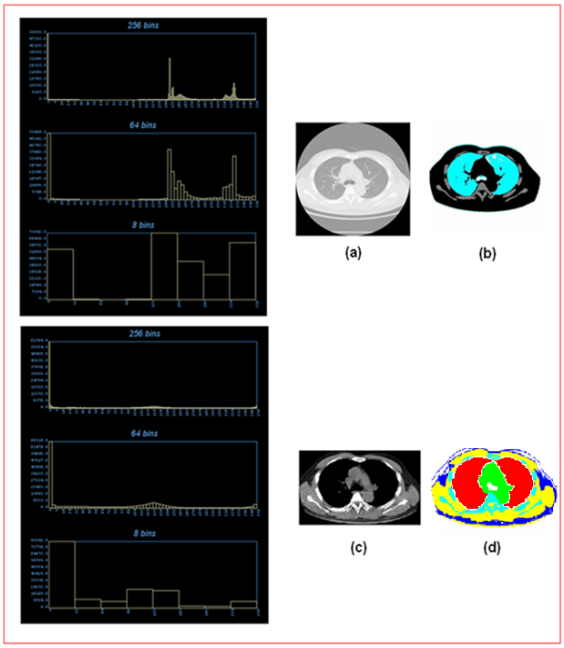

Figure 22. (a) Raw MRI data and its histogram before pre-processing (b) Segmentation of part a (c) Raw MRI data and its histogram after pre-processing (d) Segmentation of part c.

Figures 23 and 24 both represent the volumetric visualization of the same raw dataset. Figure 23 is based on figure 21, while figure 24 is merely produced by standard ray casting. At first glance, it seems that figure 24 offers more detailed visualization outcome compared to figure 23, which is true, but such advantage is limited to simple slabs (in this case, chest and lung), which originally possess fine intensity distinction between the different organs. Please notice that even in a perfect situation such as figure 24, there is still edge artifacts within the narrow object regions. Figure 25 compares figures 23 and 24 at a magnified level. As it can be noticed, in figure 25 part (a), the ribs are mixed and there is no clear edge distinction among them, while in figure 25 part (b), ribs are clearly distinguished.

Figure 23. Volumetric visualization of chest and lung based on Figure 21.

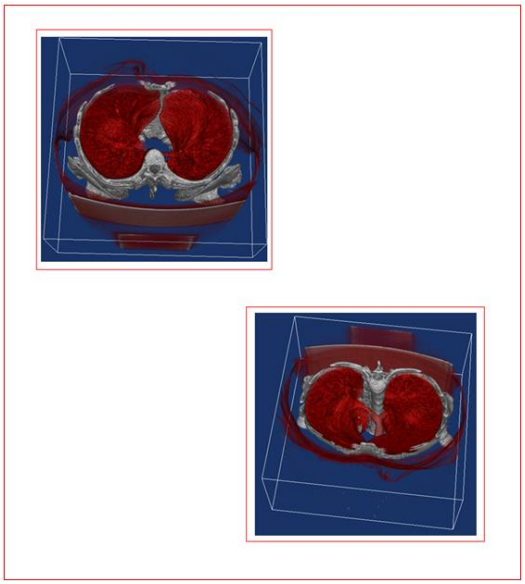

Figure 24. Volumetric visualization of chest and lung via standard ray casting.

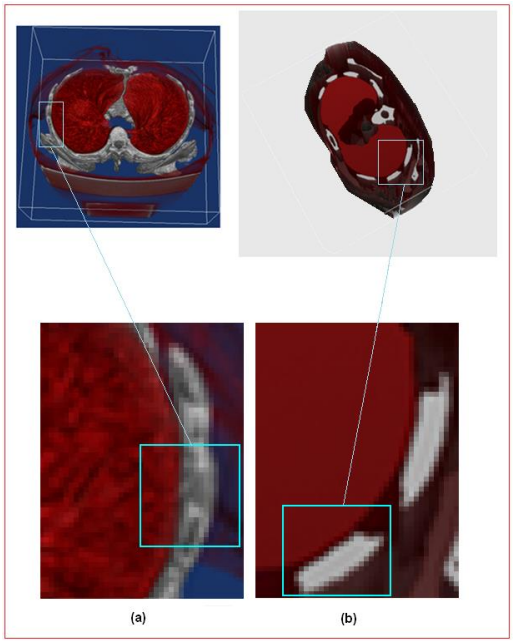

Figure 25. Chest and lung at 600% magnification (a) Ray casting (b) this work.

Figure 26 follows a similar path with figure 21 in the sense that segmentation is performed over several slices and distinct details of the desired organs are produced. Such distinctions are then utilized by rendering pipeline to generate volumetric tags. Figure 27 further clarifies the limitation of ray casting and justifies the importance of a tagged mechanism for complex slabs. As it may be noticed, compared with part (a), in part (b), kidney is properly identified, and there is a clear visualization of relevant tissues. In part (a), there are many overlapping fragments, leading to unnecessary clutter and obstructed visualization. Thus such overlaps are due to rudimentary transfer function design of the ray casting technique.

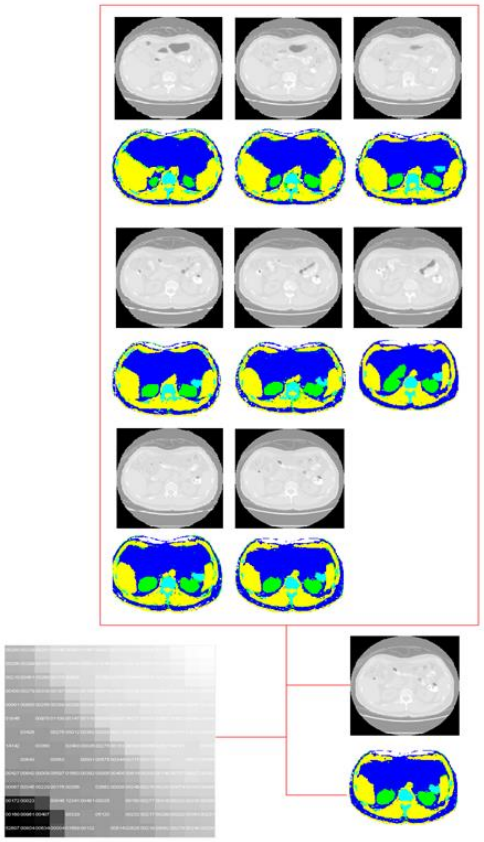

Figure 26. Kidney and liver gray level slabs and their corresponding segmentations. A representative segmentation along with its 14 × 14 neural map.

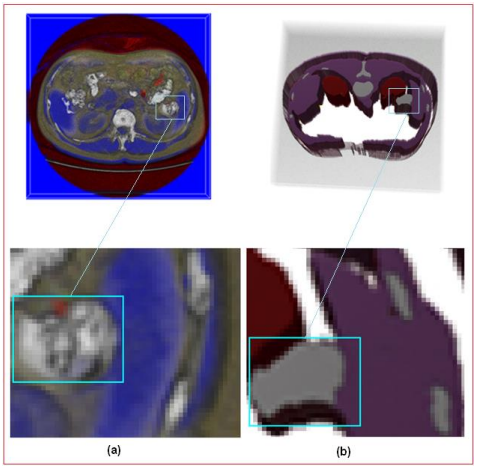

Figure 27. Bone and liver at 600% magnification (a) Ray Casting (b) This work.

Figure 28 compares the tri-linear interpolation method used by ray casting with our proposed interpolation method. As it may be noticed, the average of relative errors for this work is relatively lower for an equivalent number of randomly selected samples. Given the scalar value for each vertex, one could perform a tri-linear interpolation to obtain a representative sample for each voxel. In

the case of our method, we used an ID value at each vertex. Then we performed interpolation based on IDs. A comparison between the tri-linear interpolation method and our interpolation method is based on the relative error. Since a sample point is interpolated using values of eight cubical vertices, we can calculate the relative error between each vertex and the interpolated point with:

$$E = \frac{|A-B|}{|B|} \quad (10)$$

where A, B, and E refer to the interpolated value, vertex value, and relative error, respectively. Now, in order to obtain a single measurement of error for all vertices of a voxel, we can calculate an average error as:

$$A = \frac{\sum_{i=1}^{n} E_i}{n} \times 100 \quad (11)$$

where A, $\boldsymbol{n}$, and E refer to the average of relative errors, number of vertices, and relative error, respectively.

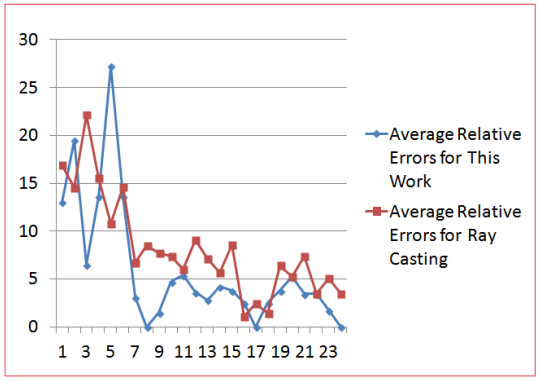

Figure 28. Ray casting's interpolation error verses this work's interpolation error. The horizontal axis refers to samples (multiples of hundred) and the vertical axis refers to average relative errors.

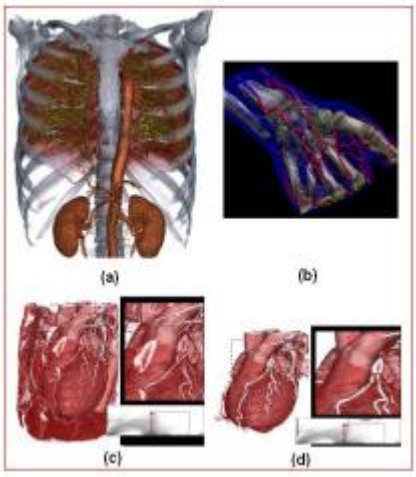

Figure 29. (a) Skeleton-based segmentation [4] (b) Manual and explicit dissection [1] (c) and (d) Heart visualization after fast marching active contour segmentation.

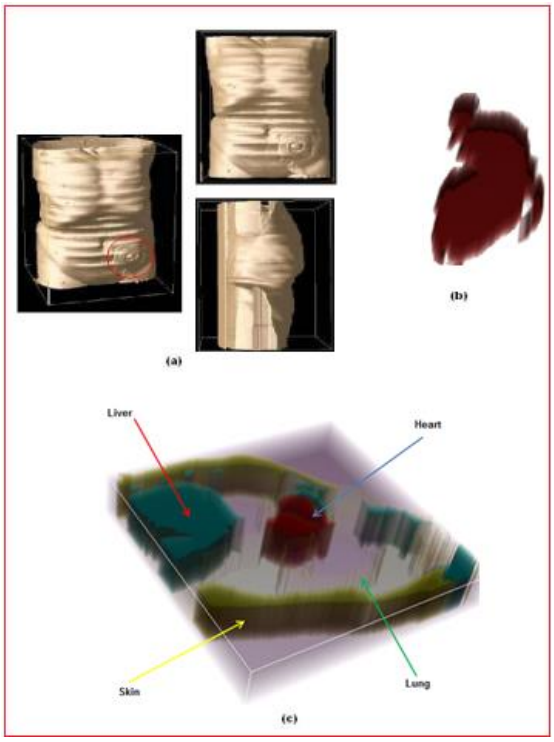

Figure 30. (a) This work's skin visualization (b) This work's heart visualization (c) This work's abdomen visualization.

As it may be noticed, figure 30 insinuates the implementation of the methodology of this work. Figure 29, on the other hand, enables a comparison by demonstrating the results from other articles. Depending on the application (inclusive/exclusive) at hand, the nature of segmentation and the relevant rendering design may differ. For instance, in figure 29 part (a), an implicit segmentation design takes advantage of the skeletal structure to visualize abdomen section. Organs closer to main bones are identified clearly. In figure 30 part (c), organs are

less cluttered. In figure 29 part (b), the segmentation aspect is insignificant but an elaborate rendering pipeline is proposed. The blue skin area is comparable to figure 30 part (a). Figure 29 parts (c) and (d) focus on the segmentation, and perform the rendering task via a rather openly available package, while figure 30 part (b) undertakes both segmentation and rendering in detail.

References

[1] Hadwiger, M. Berger, C. & Hauser, H. (2003). High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. International conference on visualization, USA, pp. 301-308.

[2] Mueller, D. & Oshea, P. (2007). Tagged volume rendering of the heart. International conference on medical image computing and computer assisted intervention, pp. 194-201.

[3] Hauser, H. Morz, L. Bischi, G. I. & Groller, M. E. (2001). Two level volume rendering. International conference on visualization and computer graphics, pp. 242-252.

[4] Xiang, D. Tian, J. Yang, F. Yang, Q. Zhang, X. Li, Q. & Liu, X. (2011). Skeleton cuts-an efficient segmentation method for volume rendering. Journal of visualization and computer graphics, vol. 17, no. 9, pp. 1295-1306.

[5] Prassni, J. Ropinski, T. & Hinrichs, K. (2010). Uncertainty aware guided volume segmentation. Journal of visualization and computer graphics, vol. 16, no. 6, pp. 1358-1365.

[6] Jeong, W. Beyer, J. Hadwiger, M. Vazquez, A. Pfister, H. & Whitaker, R. T. (2009). Scalable and interactive segmentation and visualization of neural processes in EM datasets. Journal of visualization and computer graphics, vol. 15, no. 6, pp. 1505-1514.

[7] Pfister, H. Lorensen, B. Bajaj, C. Kindlmann, G. Shroeder, W. Avila, L. Raghu, K. Machiraju, R. & Lee, J. (2001). The transfer function bake-off. Journal of computer graphics and applications, vol. 21, no. 3, pp. 16-22.

[8] Kniss, J. Kindlmann, G. & Hansen, C. (2002). Multi-dimensional transfer functions for interactive volume rendering. Journal of visualization and computer graphics, vol. 8, no. 3, pp. 270-285.

[9] Meissner, M. Huang, J. Bartz, D. Mueller, K. & Crawfis, R. (2000). A practical evaluation of popular volume rendering algorithms. International conference on volume visualization, pp. 81-90.

[10] Luo, M. Fei, Y. Hong, M. Jiang. & Zhang. (2003). A spatial constrained k-means approach to image segmentation. International conference on information, communication and signal processing, Singapore, pp. 738-742.

[11] Shafiee, A. M. & Latif, A. M. (2014) Modified CLPSO-based fuzzy classification System: Color Image Segmentation. Journal of AI and datamining, vol. 2, no. 2, pp.167-179.

نشریه هوش مصنوعی و داده کاوی

بهینه‌سازی نمایش سه بعدی اشیاء به کمک تکنیکهای پردازش تصویر

آرش عظیم زاده ایرانی و رضا پورقلی*

دانشکده ریاضی و علوم کامپیوتر، دانشگاه دامغان، دامغان، ایران.

ارسال ۲۰۱۸/۰۶/۲۵؛ بازنگری ۲۰۱۸/۰۸/۱۲؛ پذیرش ۲۰۱۹/۰۴/۰۷

چکیده:

ری کستینگ یک روش رنگ آمیزی مستقیم مکعب است که برای نمایش سه بعدی استفاده می شود. این روش کاربردهای مهمی در زمینه تصاویر پزشکی و بیولوژیکی دارد. نقطه ضعف این روش مکانیزم کلاسه‌بندی ضعیف آن است. توابع تیدیل در این روش همپوشانی داشته و اشیاء را بدرستی جدا نمیکنند. ما یک روش بر اساس تکنیکهای پردازش تصویر ارائه میکنیم که این نقصان را بر طرف کرده و یک مکانیزم بهینه ارائه میکند.

کلمات کلیدی: روش سگمنتیشن هیبرید، رنگ آمیزی مکعب، بهینه‌سازی نمایش.