Volume Exploration Made Easy Using Feature Maps

Klaus Mueller, Sarang Lakare*, and Arie Kaufman[1]

Center for Visual Computing, Computer Science Department, Stony Brook University, Stony Brook, NY 11794, USA.
{mueller,lsarang,ari}@cs.sunysb.edu

We present a framework that enables an intuitive, feature-centric exploration of segmented volumetric datasets. Our system is geared towards users familiar with the basic elements of volume rendering, but who seek to conduct volume exploration in a guided fashion. It provides the infrastructure to organize features and objects extracted from a volume dataset, via segmentation or otherwise, and provides the functionality to view these features with standard volume rendering tools. A novel aspect of our system is that it does not require a separate binary tag volume to indicate the presence of a feature (or object). Instead, we mark a feature by migrating its density range, including its smooth boundary, to a private interval. This avoids the aliasing problems associated with binary tag volumes as well as the extensive run-time costs incurred to resolve these. In addition, since the smooth boundaries of the features are preserved, any volume renderer can be used for data visualization, without modification.

1 Introduction

The extraction of features in volumetric datasets remains a hard task, and these difficulties are among the main impediments in making volume rendering a main-stream data investigation tool for general users, such as medical doctors and other clinical personnel, computational scientists, and even data miners. A main advantage of volume visualization is that it allows users to "play" with the data, exploring different aspects in an engaging interrogative experience. This is usually done via modifying the transfer functions that map raw volume data to visual attributes, such as color and transparency. Transfer functions give users the flexibility to "sculpt" a visualization from the raw volume data, including densities and their derivatives. Transfer functions allow users to show certain features as soft, semitransparent gel-like materials or

*Currently at Siemens Medical Solutions, Malvern, PA, USA

as accentuated surfaces. A number of elaborate tools have become available that enable users to assist in this endeavor. One such framework is the dual-domain interaction tool by Kniss et al. [8], which builds on earlier work by Kindlmann and Durkin [7]. Here, users can find and accentuate features by probing the volume to find critical points in a 2.5D histogram that plots the magnitudes of first and second derivatives over voxel densities. Users can then place so-called transfer function widgets directly into the histogram to visually accentuate the probed features. Tenginakai and Machiraju extended the range of data signatures from the first and second-order to higher-order moments and their derivatives, such as skew and kurtosis [23], which essentially extends the dimensionality of the transfer functions further. Other related work includes that of Pekar [14] who applied cumulative Laplacian-weighted gray value histograms.

While these tools are undoubtedly extremely valuable for visualization experts, they are likely too involved for users who are only marginally experienced in the theoretical underpinnings of feature exploration using data signatures. Here, it does not help either that the tools become quite difficult, and perhaps even inadequate, to use once the density distributions in the data grow more complex, such as for MRI volumes, fine-scale computational data, and others. An example for such a configuration is illustrated in Fig. 1, which shows the visible human's foot. In this dataset, both the muscle and the bone-marrow have very similar data signatures, and the corresponding region voxels will all fall into the same portion of the density-signature histogram. Thus, their given visual attributes will overlap as well, and as a consequence, the two features will not be visually distinguishable.

It has become a trend, in particular in medical visualization, to create very simple interfaces for clinical practitioners. These applications have just a few buttons and sliders, to allow a fast and target-oriented visualization of the patient data for diagnosis and planning. In the typical case, there are a number of task-specific feature extraction tools, such as a vessel segmentation tool sensitive to tubular structures, or a lung nodule tool sensitive to spherical objects of certain densities [19]. Beyond these capabilities, there are typically only a set of navigation facilities, such as zoom, rotate, tilt, and slice. No transfer functions are usually available to change visual attributes, rather, users can choose among a few provided colormaps to colorize the data.

Thus, these highly-specialized tools are on one end of the spectrum of data visualization frameworks, while the data-signature tools mentioned earlier are on the other. In this paper, we suggest a system that is somewhere inbetween. It is geared towards a user who is comfortable in using low-dimensional transfer functions for volume exploration, and would like to enjoy the benefits of applying density-based histograms to create custom visualizations of a dataset. This user either lacks the expertise, time, or motivation to engage into a session with a complex transfer function-based data explorer that operates directly on the raw volume data. Instead, our system provides what one might call a "groomed" data exploration experience, that is, the data are

Fig. 1. Features with similar data signatures are hard to distinguish since they map to the same visual attributes. (a) Muscle and bone marrow are both mapped to gray when using transfer functions. (b) Our method maps the bone marrow's density interval to a private range, which allows different visual attributes to be assigned for it (compared to muscle), while using the same transfer function interface as in (a).

converted into a representation in which exploration with transfer functions is still possible, yet the transfer functions have low dimensionality and are therefore easy to manage and to interact with.

Our system takes as its input a segmented dataset, which has either been prepared by an automated segmentation method or an experienced "senior-user" with an advanced interactive segmentation tool. Here, the segmentation could have been obtained via seed-growing, watershed algorithms, snakes, balloons, live-wires, statistical methods, level sets, deformable models, feature-tracking, and the like. In this regard, our tool may also proof useful to fine-tune a prior automated pre-segmentation. Our system represents the extracted features as a graph, which allows grouping and selection of individual features. However, contrary to other methods of this nature, it does not represent the features as binary objects captured in a tag volume. The problem with tag volumes is that they must invoke tedious and time-consuming algorithms when a ray sample point falls into the proximity of a boundary interface, to determine the object at that position for visual property look-up [24]. When more naive algorithms are used instead, staircasing artifacts may be visible in the resulting image. Our system, on the other hand, maintains objects in their original fuzzy boundary representation, which avoids all visual artifacts and allows any available volume renderer to be used unchanged to produce the visualization.

Our paper is structured as follows. First, in Section 2, we will overview related work and preliminaries. The following sections will then focus on our new contribution. Here, Section 3 will describe our approach of feature migration, in which we port the density intervals of segmented features to private intervals, and Section 4 will describe how these features can be managed us-

ing transfer functions in conjunction with feature maps. Finally, Section 5 will end with final conclusions and give an outlook onto future work.

2 Preliminaries and Related Work

Our method takes as its input regions of voxels that have been associated with a particular feature or region of interest. Here, a feature may just be one of the skeletonized toes in a foot dataset, the entire skeleton, the muscles, or any other region. This feature-centric approach makes it possible for visualization users to manipulate the feature in isolation of other, previously signature-similar, regions, which couldnt be distinguished with the transfer function interface before.

We refer to the process of extracting the voxels belonging to a particular region of interest as volume extraction. This term has been used previously in the literature [1, 22], but it has always referred to the extraction of 3D surfaces from a volume. In contrast, we actually extract voxels from one volume and port them into another volume, which we call the feature volume. This feature volume is the data, which the end-user interacts with. Here, volume regions, which were not specifically tagged as features may simply be copied into the new volume at verbatim, without change.

One of our main goals is to allow high quality volume rendering of these feature regions. To achieve this, we need to avoid artifacts caused by aliasing, since any aliasing in the data results in subsequent aliasing effects in the volume rendered image. One of the locations where aliasing often occurs in extracted volumes is at the boundary of the segmented region. We shall illustrate this by ways of an example. Consider Fig. 2a (left), where we show a cross-sectional image of the Engine dataset, while Fig. 2b (left) shows the result of a segmentation via thresholding, followed by a simple extraction of the valve in which the intensities of all voxels that do not belong to the valve are set to the air intensity. Consider now Fig. 2a (right) which shows the zoomed-in images of the valve. We observe that the object boundaries were originally fuzzy, and not binary. That is, there is a smooth intensity variation as one moves from one region to another due to the partial volume effect. In the segmented image (see Fig. 2b (left)), however, the fuzziness is not present. Instead there is a sharp contrast between the intensities of the segmented valve and the surrounding air. This sharp contrast causes an aliasing effect, mostly due to the poor gradient estimation it affords (see the rendered images Figs. 2c and 2d (left)). The intensity-flipping algorithm described in [8] restores the fuzziness at the boundaries of an extracted feature, and rendering results are shown in Figs. 2b, c and d (right).

The effect of the intensity flipping algorithm of [11] is illustrated in Fig. 3a. Here, the boundary profile (labelled before flipping) separates two objects. The goal is to remove the object with the higher density (the one on the right), but leave the boundary characteristics (the position of the zero-crossing of

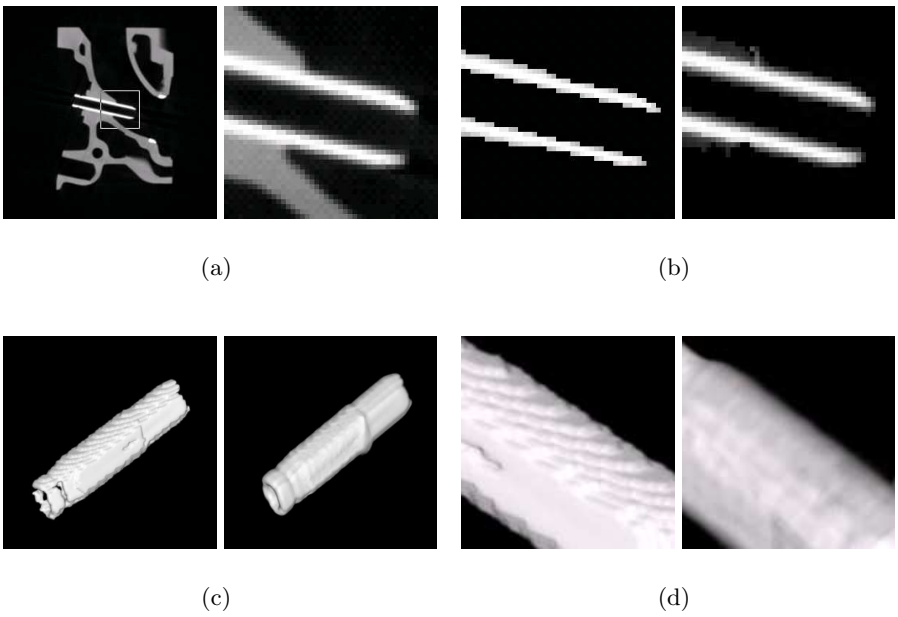
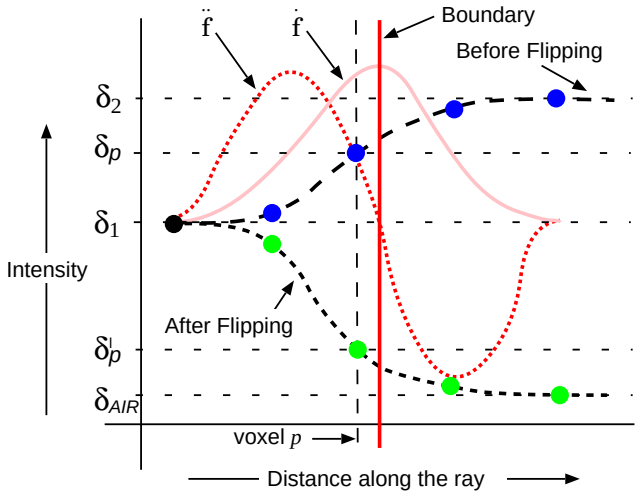


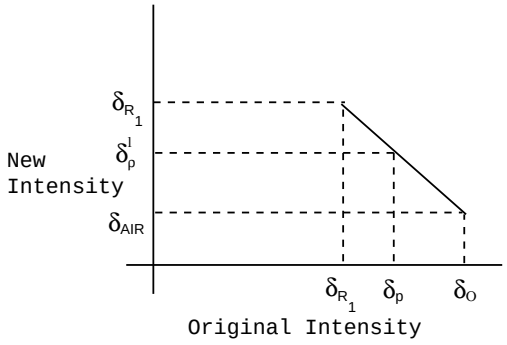
Fig. 2. Feature extraction (the valve) from the Engine dataset: (a) original slice (left) and zoomed into the valve (right), (b) binary extraction (left) and fuzzy extraction (right), (c) and (d) volume renderings of the extracted valve: binary extraction (left) and fuzzy extraction (right).

the second derivative and the location of the maximum of the first derivative) unchanged since these will determine the location of the iso-surface in the volume rendering (while the fuzziness will determine gradients and the overall look of the volume rendered surface). Fig. 3b shows the intensity/density transfer function. Here, δ_1 is the density of the object we would like to keep, while δ_2 is the density of the object we would like to remove. The range $\delta_2 - \delta_1$ is the density range that the boundary bridges. The transfer function maps this density range linearly to the range $\delta_1 - \delta_{AIR}$ such that the new density of the other side of the boundary to δ_1 is set to δ_{AIR}. The algorithm described in [11] performs this mapping using local measures of δ_1 and δ_2, thus the mapping function adapts to the local density statistics of the boundary. The current work borrows from this technique, when merging the extracted features into the newly constructed feature volume, preventing aliasing in the process.

Work related to our intensity-flipping technique includes the smooth boundary enhancement of voxelized geometric objects [1, 21] as well as adaptive distance fields (ADFs) [5, 4]. While the former methods append a smoothly (linearly) decaying intensity seam to the binary density field generated by voxelization algorithms, the latter stores a distance field in the



(a)



(b)

Fig. 3. Effect of the intensity flipping algorithm. (a) The old and new boundary profile with the location of the maximum of the first derivative and the zero-crossing of the second derivative left intact. (b) The corresponding density/intensity transfer function.

non-occupied voxels. Both help alleviate the aliasing problems associated with binary and near-binary objects. In contrast, our method is not geared towards voxelized objects and does not seam an object with a static, artificial function. Rather, it restores the original smooth density-falloff at the boundary of the sampled object, which may vary with spatial location. These boundary effects

may be partially due to the partial volume effect and the lowpassing of the sampling process, but may also be due to the object characteristic itself.

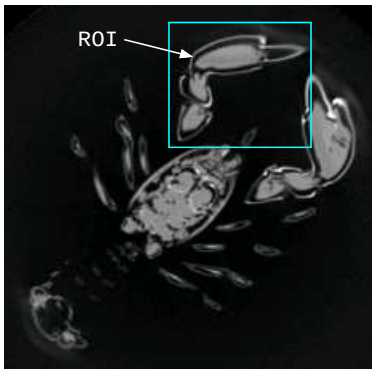
Another approach to reach our goal might be gleaned from the soft-segmentation technique [16]. Soft-segmentation seeks to overcome the fundamental problem in image segmentation. For many images there is no way to uniquely and correctly determine the object boundaries. Instead of the usual crisp segmentation where a fixed boundary is derived for a region, soft-segmentation proposes an approach where a pixel/voxel can belong to more than one region. This results in voxels around the region boundary being assigned partially to each of the neighboring regions. However, such an approach would require either non-scalar volumes for storage of these mixed-membership voxels or an extra set of tags, one for each mixed class. We will achieve similar effects with a single scalar volume, allowing users to enhance and visualize mixed regions by ways of the transfer function.

3 Density Range Migration

In the following, we will use the term region [2] to refer to a connected group of voxels with similar properties (e.g., a bone or a muscle), while we use the notion region of interest (ROI) to describe a group of one or more connected regions that we are interested in. The ROI is the feature to which range shifting is applied so that it can be assigned optical properties without interfering with other objects or features. Our algorithm is composed of four steps. In the first step, we define the ROI. In the second step, we separate the data voxels into different categories in order to perform different actions on them. In the third step we move the scalar value range of the ROI so that it now occupies a different space in the histogram, and in the final step we modify the boundary between the ROI and its neighboring regions to reflect the change in the ROI range. We shall now describe these four steps in turn.

3.1 Defining the ROI

The ROI is defined by a mask volume, which marks all voxels that are part of the ROI, as well as a list of voxels that form the boundary of the ROI. A voxel is said to be on the boundary if one of its neighboring voxels is not part of the ROI. The mask volume, which defines the ROI can be generated using any suitable segmentation algorithm (as mentioned in the introduction). Feature tracking has also been employed for time-varying datasets [13, 20]. Fortunately, the recent advances in computer graphics hardware [10, 18] make interactive, user-assisted methods, such as seed growing [9, 17], level set methods [12], or snakes/balloons [6], a viable solution. Using such an interactive segmentation system with immediate visual feedback on the segmentation result, ROI-mask voxels can be quickly labeled. We shall illustrate our algorithm using the Lobster CT dataset as an example. Fig. 4a show a cross-section of



(a)

(b) (c) (d)

Fig. 4. The Lobster dataset serves as an example to illustrate parts of our algorithm. (a) a slice, (b) segmenting the ROI (the left claw), (c) marking the boundary voxels, (d) categorizing the voxels.

this dataset. The goal of our example application is to apply range shifting to one of the claws, which forms the ROI (shown in a box) in our example. We segment the claw using an interactive seed growing algorithm. The result of the segmentation is shown in Fig. 4b. The voxels which lie on the boundary of the ROI are highlighted in Fig. 4c.

3.2 Categorizing the voxels

After selecting the ROI, we divide the data voxels into separate categories depending on the actions we perform on them in the later stages. From the previous step we have identified the voxels that are part of the ROI and a list of those that lie on the ROI boundary (Fig. 4b, c). Each ROI boundary voxel can be of one of two types: (1) adjacent to at least one non-boundary ROI voxel, and (2) not adjacent to any non-boundary ROI voxel.

In Fig. 4d, the voxels of the first and second have been painted in differnt shades of gray. The significance of the voxels of the second type (shown in light gray on the periphery) is that they are part of a very thin ROI. The ROI is so thin that all ROI voxels are also boundary voxels.

A volumetric dataset obtained from a scanning modality, such as CT, MRI or PET, displays a natural fuzziness at the object boundaries. This can be attributed to the partial volume effect which occurs due to a finite resolution sampling of a continuous signal by the scanners and the reconstruction operators. The ROI boundary voxels are also part of a naturally fuzzy boundary between the ROI and its neighboring regions. The only exception occurs when the ROI boundary voxels are of the second type. In that case, the ROI boundary voxels are not part of the fuzzy boundary, however, the voxels that surround them are.

We now attempt to find all voxels that form this fuzzy boundary. We assume that the width of the fuzzy boundary is between 3-4 voxels. This can be justified by the fact that commonly the radius of the Gaussian kernel used in 3D reconstruction from medical and other image data is about 1.5 times the size of the voxel. To mark the fuzzy boundary voxels, we grow the ROI boundary region once in all directions using an 18-neighbor region grow, which is a 3D version of the 8-neighbor (in 2D) region grow [15]. Next, we categorize the data voxels into the following four categories:

1. Voxels that form the fuzzy boundary between the ROI and its neighboring regions.
2. Voxels of a thin ROI.
3. The remaining voxels of the ROI.
4. The rest of the voxels in the dataset.

We begin with voxels which belong to category 1. In this we include the *ROI* boundary voxels of the first type and the fuzzy boundary voxels we found by region growing. The ROI boundary voxels of the second type are the category 2 voxels. The remaining voxels of the ROI belong to category 3. All remaining voxels in the dataset belong to category 4 (shown in black in Fig. 4).

3.3 Moving the ROI density range

In this step, we change the scalar value of the *ROI* voxels, which effectively moves the density range of the *ROI*. We start by finding the current density range of the *ROI*. We define the range as a pair of the lowest and the highest scalar values present, and the width of the range as their difference. The range can be easily computed by looping over the voxels once to find the highest and the lowest scalar values. Migrating the entire density interval preserves the fine density fluctuations within the object. These can be meaningful in later visualizations. For example, NPR (Non-Photo-Realistic) techniques can

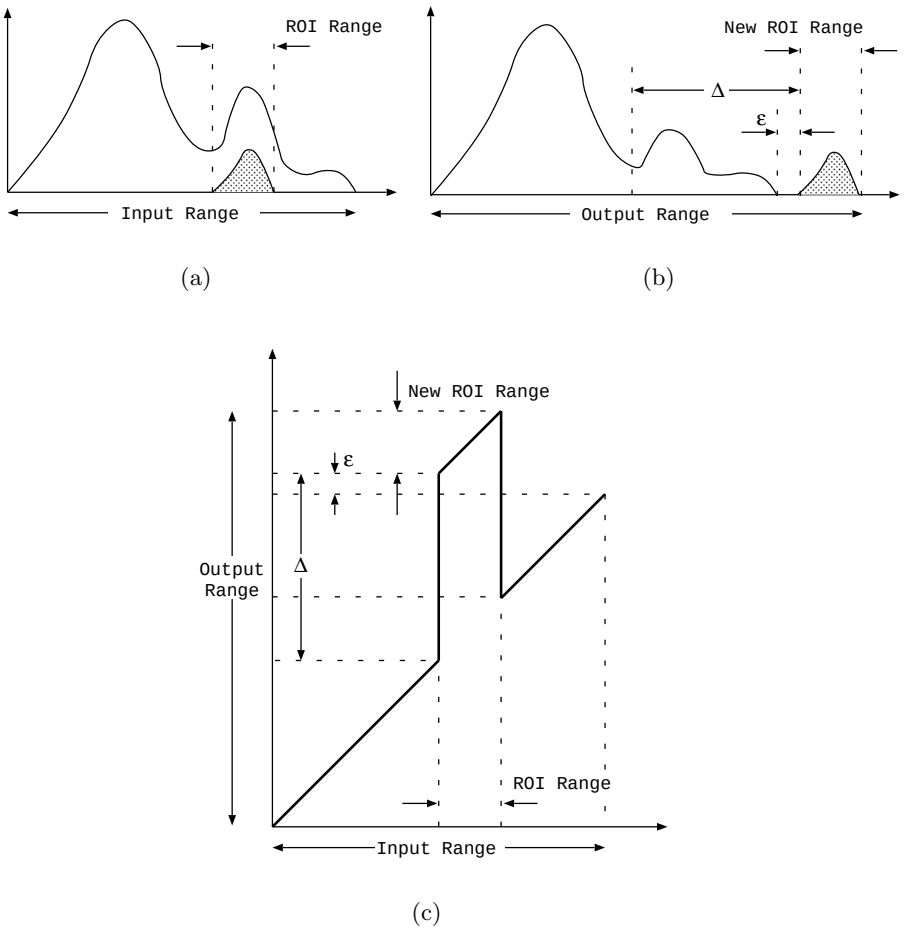


Fig. 5. Migration of the ROI densities: (a) the histogram of the input volume, (b) the histogram of the constructed feature volume, after importing the migrated (shifted) ROI density interval, (c) the density mapping function that achieves this.

be applied to accentutate even minute density variations to produce more insight into the micro-scale definition of an object [3].

In the following, we assume that the range of the input data is: (IN_{low}, IN_{high}). We find the range of the ROI by taking into consideration all category 2 and category 3 voxels. We denote this range by (ROI_{low}, ROI_{high}) and its width by ROI_{width}. Some of the ROI voxels (those on the ROI boundary that now belong to category 1) are not considered when finding the range because they are part of the partial volume and we want to avoid the partial volume being part of the range.

As our goal is to move the range of the ROI, we have to select the new range. The new range should be such that no other voxel in the input data has values in that range. There are two possibilities for the new range. First, the new range can be placed within IN_{low} and IN_{high} if there are ROIwidth consecutive levels of the range that are unused in the input data. In this case, the range of the resultant data remains the same as that of the input data. However, in practice, such cases are rare as the voxel values usually cover the entire range. The second possibility is to place the new range outside the initial range of the input data. This effectively increases the range of the resultant data to:

$$Out = (IN_{low}, IN_{high} + ROI_{width}) \tag{1}$$

We focus on this second case since it is always possible to increase the data range to accommodate the new range of the ROI. Assuming that the ROI range is to be moved beyond the current data range, the new range for the ROI is:

$$ROI' = (IN_{high} + \epsilon, IN_{high} + ROI_{width} + \epsilon) \tag{2}$$

where e is a small number to provide a gap between the ROI and the rest of the data in the histogram (Fig. 5a,b). This gap makes transfer function assignment easier when focussing on the ROI. The starting position of the new ROI range is:

$$ROI'_{low} = IN_{high} + \epsilon \tag{3}$$

Thus, the shift in ROI range from the original position to the new position is given by:

$$\Delta = IN_{high} + \epsilon - ROI_{low} \tag{4}$$

We can now write the equation for moving the ROI range as:

$$\delta' = \begin{cases} (\delta + \Delta) & \text{if category 2 or 3 voxel} \\ \delta & \text{otherwise} \end{cases} \tag{5}$$

where δ is the original scalar value of the voxel and δ' is the new scalar value for the voxel. This transformation for the category 2 and 3 voxels is shown by the graph in Fig. 5c. Fig. 5a shows a hypothetical histogram of a dataset with the ROI histogram shown by a dotted region. The result after applying (5) to the histogram in Fig. 5a is shown in Fig. 5b.

3.4 Reconstructing the fuzzy boundary

The final step consists of reconstructing the fuzzy boundary such that the fuzziness reflects the new region intensities around the boundary. This procedure has two steps. In the first step, we compute a local average intensity

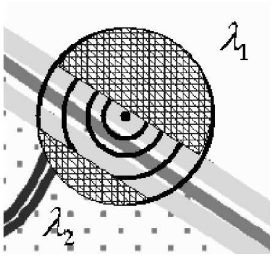


Fig. 6. Region-grow with voxel p at the center.

of the regions surrounding each voxel in the fuzzy boundary. In the second step, we use these local averages to compute the new intensity for the fuzzy boundary voxel. The following two steps are repeated for each voxel p that belongs to category 1.

Step 1. With p as the center, in the original dataset, a region-grow is performed such that the region includes voxels which are nearest to the central voxel and belong to categories 2, 3 or 4. All category 1 voxels are ignored (Fig. 6). The region-grow continues until the following two conditions are satisfied:

- The region includes at least l voxels from categories 2 or 3 OR γ voxels from category 2.
- The region includes at least l voxels from category 4.

where l is a number small enough to compute a reliable local average intensity of the region, and g is a number small enough to compute a reliable local average intensity of the ROI voxels in the thin ROI (category 2). A large value for l or g can result in a wrong average intensity for the surrounding regions. For our experiments, we chose a value of 5 for l and 3 for g. The region-grow is stopped after 3 iterations even if the above conditions are not satisfied. This is done to avoid going past a 3-voxel radius. Voxels beyond a radius of 3 are unlikely to affect the intensity at voxel p and should not be considered. The idea behind the region-grow is to find the voxels that are nearest to voxel p. Some of these belong to categories 2 or 3 and some to category 4. These neighboring voxels represent the regions that surround p, and are responsible for influencing the intensity value at p. We therefore calculate the average intensity of each of these sets of neighboring voxels. We assume that at the end of the region grow, we have li voxels that belong to categories 2 or 3 (or gi voxels that belong to category 2) and lo voxels that belong to category 4. The subscript i denotes that the voxels belong to the ROI and the subscript o denotes that the voxels are outside the ROI.

We compute the average intensity of the ROI voxels nearest to p, denoted by δ_i, using the following:

$$if(\gamma_i = \gamma) \quad \delta_i = \frac{\sum_{k=1}^{\gamma_i} \delta_k}{\gamma_i} \quad \text{else} \quad \delta_\text{i} = \frac{\sum_{\text{k}=1}^{\lambda_\text{i}} \delta_\text{k}}{\lambda_\text{i}} \tag{6}$$

The condition $\gamma = \gamma_i$ checks if the nearest ROI voxels belong to a thin ROI and in that case γ voxels are used to compute the average intensity rather than the l voxels. This change allows our algorithm to work even when the ROI is extremely thin. We similarly compute the average intensity of the non-ROI voxels nearest to p, denoted by do using:

$$\delta_o = \frac{\sum_{k=1}^{\lambda_o} \delta_k}{\lambda_o} \tag{7}$$

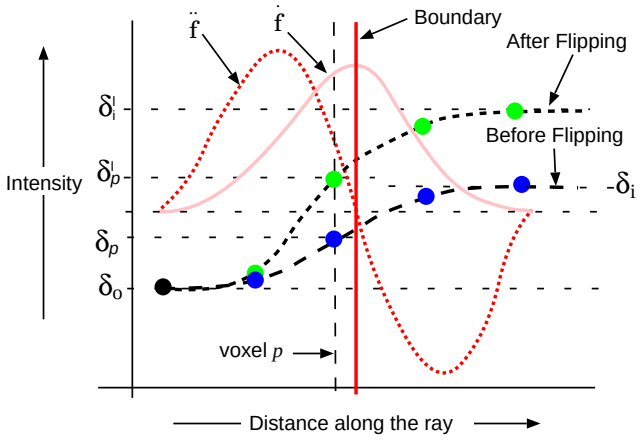
Equations 6 and 7 assume that the values of γ_i, λ_i, and λ_o are non-zero. This is however not always true. An important condition under which one of the values is zero (except γ_i which can be zero when p is not near a thin ROI), is when the region grow stops after going past the 3-voxel radius. At this stage, we make an assumption that since there is no voxel from one of the regions within a 3 voxel radius, the voxel was probably categorized incorrectly as a category 1 voxel and does not belong to the fuzzy boundary. We re-categorize the voxel based on the region whose voxels were included in the region grow. If λ_i is non-zero and λ_o is zero, we consider the voxel as a category 2 or 3 voxel that belongs to the ROI and change its intensity based on 5. In the other case, the voxel is considered part of category 4 and its intensity remains unchanged. Another degenerate case occurs when either λ_i or λ_o are not equal to λ at the end of the region grow. In those cases, we use 6 and 7 to compute the average intensities.

In addition to λ_i and λ_o we also need to compute the new average intensity of the ROI voxels nearest to p after the shifting of the ROI range. We compute the new average intensity without performing another region grow. A region grow performed from p on new values of ROI would include the same voxels that were included in the first region grow performed at p, since we do not move the location of any voxel. Also, all of the ROI voxels included (either γ_i or λ_i) have values shifted by D in the previous step as they belong to category 2 or 3. Thus, the new average intensity of these ROI voxels is shifted by D, and is given by:

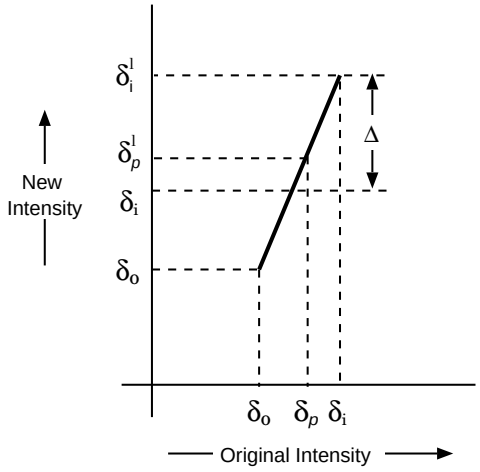
$$\delta_i' = \delta_i + \Delta \tag{8}$$

Step 2. The goal in this step is to find the new intensity at voxel p, after the ROI range has moved. In Fig. 7a we show an intensity profile along a hypothetical ray that would pass through voxel p, moving along the approximate direction of the gradient, but traversing through voxel centers. Before intensity flipping, the ray profile would be as shown in Fig. 7a with the points which are the voxels that the rays traverses. As we go across voxel p (dashed vertical line), we see that the intensity gradually changes from δ_o, to δ_i. This intensity profile basically depicts how the intensity changes at the boundary of the region (solid vertical line), as we move across the boundary.

The goal of intensity flipping is to preserve the boundary location while changing the intensities of the voxels along the ray. The basic idea is that if the



(a)



(b)

Fig. 7. Migrating a density interval. (a) The old and new boundary profile with the location of the maximum of the first derivative and the zero-crossing of the second derivative left intact. (b) The corresponding density/intensity transfer function.

new voxel intensities are all scaled by the same ratio, the gradient magnitude would increase, but the maxima, which defines the boundary, will remain at

the same location. The assumption here is that the voxel intensities along the ray lie inbetween those of the surrounding regions (δ_i and δ_o).

We now present the generic intensity flipping equation when one of the regions around the voxel p has changed its average intensity. Assuming that δ_i is the new average intensity of the region whose original average intensity was δ_i, the intensity flipping equation will be given by:

$$\delta'_p = \delta_o + \frac{\delta_p - \delta_o}{\delta_i - \delta_o}(\delta'_i - \delta_o) \tag{9}$$

where δ'_o is the new intensity for the voxel p. Substituting δ'_o from Equation 8 into Equation 9 we get:

$$\delta'_p = \delta_o + \frac{\delta_p - \delta_o}{\delta_i - \delta_o}(\delta_i + \Delta - \delta_o) \tag{10}$$

The intensity flipping curve corresponding to (10) is shown in Fig. 7b. When this intensity flipping is applied to all the voxels along the ray, the intensity profile of our hypothetical ray changes, and is shown by the curve labelled "after flipping" in Fig. 7a. The points are the intensities of the same voxels along the ray but now with different values than before. As a result of this flipping, the first derivative changes, but the maximum of the first derivative (the boundary) remains at the same spatial location. Similarly, the second derivative zero-crossing remains unchanged. This nice property gives us an unchanged boundary location, even though one of the regions around the boundary has changed.

Equation 10 is based on the assumption that dp is somewhere inbetween δ_i and δ_o. However, there are cases when this is not true. This is either due to noise in the dataset or due to an incorrect location of the fuzzy boundary. We solve this degenerate case by suggesting that whenever such a case occurs, the voxel should not be part of the fuzzy boundary, and was categorized incorrectly to category 1. We compute the new intensity for this voxel by re-categorizing the voxel based on its original intensity δ_p. If the original intensity is beyond the intensity δ_i, then we shift its intensity as if it were a category 2 or 3 voxel using Equation 5. Otherwise, we leave its intensity unchanged as if it were a category 4 voxel.

We apply the previous two steps on all category 1 voxels as mentioned before. All the category 4 voxels are left untouched.

4 Volume Exploration with Feature Maps

The process described above yields the feature volume in which each feature (or ROI) takes up a private density interval (but with the original density statistics), surrounded by a smooth boundary. In essence, the feature volume can also be regarded as a segmentation notebook where features, once captured, are collected in a common environment. In practice, we first copy the

original volume into the feature volume and then perform the segmentation and migration there. In this way, unsegmented volume portions are kept "as-is". To keep track of the extracted objects and their assigned density intervals, we log them in a hierarchical organization, which we call a feature map. Displaying this feature map directly below the transfer function window enables users to easily navigate the volume. Descriptive labels and text may also be associated with each map node, and this labeling may be done either at the segmentation stage or later by the user in the general viewing stage.

A simple example for such a user interface is depicted in Fig. 8a. Here, the fibula of the Visible Human Foot was extracted and its density interval migrated to the upper portion of the density range. The new density layout is communicated below in form of the feature map hierarchy. The user may now use the indicated density intervals (or brackets) as a guide to quickly give each feature the desired look. The color transfer function uses a color palette along with a brightness (luminance) curve, while the opacity transfer function works with the familiar 1D curve. While on the left panel the user decided to show the fibula colored in green and the remaining bones in white, on the right panel he/she chose to eliminate all other bones to reveal the fibula in full view, still colored in green. Finally, Fig. 8b shows the Engine with one of the valves extracted as a separate feature, while Fig. 8c shows the Lobster with one claw separated from the remaining tissue and shell.

5 Conclusions

We have presented a framework that enables an intuitive, feature-centric exploration of (possibly automatically) segmented volumetric datasets. This approach is geared towards users who are familiar with the basic elements of volume rendering, but who seek to conduct volume exploration in a guided fashion. These users could be scientists, medical personnel, or students. Our system provides the infrastructure to organize features and objects extracted from a volume dataset, via segmentation or otherwise, and provides the functionality to view these features with standard volume rendering tools. Our method does not require a separate binary tag volume to indicate the presence of a feature (or object). Instead, we migrate the entire density range of the feature to a private interval, including its smooth boundary. There are several advantages to this approach. First, the aliasing problems associated with binary tag volumes are avoided, as well as the run-time costs incurred to resolve these. In essence, we defer these costs to the density migration stage, which, however, typically runs in a matter of seconds. Since in our method the smooth boundaries of the features are preserved, any volume renderer can be used for dataset visualization, without modification. On the other hand, since the density statistics of the objects are preserved as well, any sophisticated volume visualizer, such as an NPR renderer, can be employed to enhance these small-scale fluctuations.

Another highlight of our system is the hierarchical feature map that captures and organizes the structural knowledge that has been gathered about the dataset. While hierarchical feature maps are also possibly used in conjunction with tag volumes, the density migration makes it possible to use them within the familiar transfer function interface to control color and opacity as a function of density. This enables users to apply the typical volume rendering effects, such as smooth semi-transparencies and gradient modulation. This luxury, however, does not come for free. In all but the simplest cases, the density range provided by 8-bit datawords will not be sufficient and longer datawords (in most cases 16-bit) will be needed to house the extended density intervals. This, however, may not be a huge problem since even GPUs can nowadays process volumes as large as 32 bits per voxel. On the other hand, tag volumes double the memory consumption as well, in addition to requiring algorithmic changes to the renderer. But in any case, future work will seek to investigate methods that can provide a dynamic range compression of the density intervals, in order to make the extended range fit into the 8-bit range. Another notable point is that due to the range migration and the associated steeper density curves at boundaries, care must be taken when rendering with gradient modulation. We have resolved this issue by using a modulation curve that saturates gradients above a certain threshold.

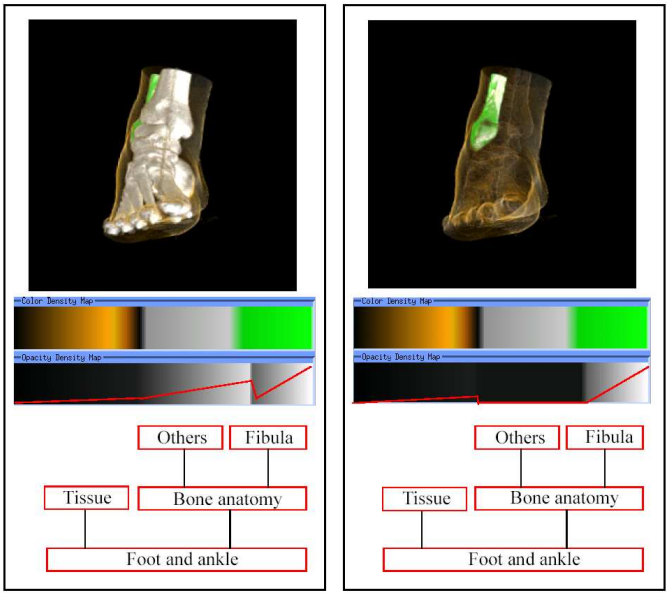
6 Acknowledgements

This work has been supported by grants from NIH #CA82402, NSF Career grant ACI-0093157, NSF CCR-0306438, CAT Biotechnology, NYSTAR, and ONR # N000110034. The engine dataset is courtesy of GE. The Visible Human foot dataset is courtesy of the Visible Human project. The authors wish to thank Manjushree Lakare and the visualization lab members for their help.

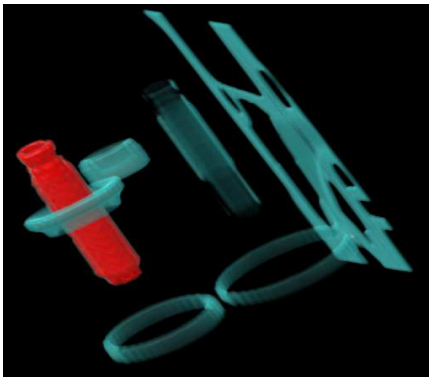
References

1. F. Dachille and A. Kaufman. Incremental Triangle Voxelization. In *Proc. Graphics Interface*, pages 205–212, May 2000.
2. E. R. Dougherty and C. R. Giardina. *Matrix Structured Image Processing*, chapter Topological Operations, pages 140–148. Prentice-Hall, 1987.
3. D. Ebert and P. Rheingans. Volume Illustration: Non-Photorealistic Rendering of Volume Models. In *Proc. IEEE Visualization*, pages 195–202, 2000.
4. S. Frisken, R. Perry, A. Rockwood, and T. Jones. Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics. In *Proc. SIGGRAPH*, pages 249–254, 2000.
5. S. Frisken Gibson. Using Distance Maps for Accurate Surface Representation in Sampled Volumes. In *Symposium on Volume Visualization*, pages 23–30, 1998.
6. M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *International Journal of Conputer Vision*, 1(4):321–331, 1988.

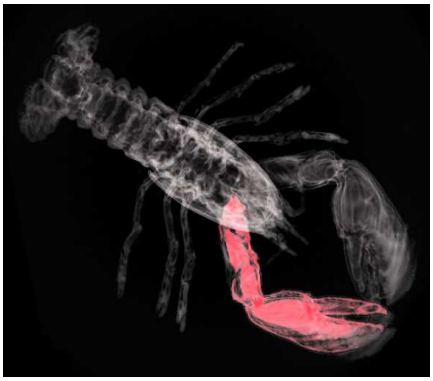
7. G. Kindlmann and J. Durkin. Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering. In *Proc. of Symposium on Volume Visualization*, pages 79–86, 1998.
8. J. Kniss, G. Kindlmann, and C. Hansen. Multi-Dimensional Transfer Functions for Interactive Volume Rendering. *IEEE Trans. on Visualization and Computer Graphics*, pages 270–285, 2002.
9. Kevin Kreeger and Arie Kaufman. Interactive Volume Segmentation with the PAVLOV Architecture. In *IEEE Parallel Visualization and Graphics Symposium*, pages 61–68, October 1999.
10. J. Krüger and R. Westermann. Acceleration Techniques for GPU-Based Volume Rendering. In *Proc. IEEE Visualization*, pages 287–292, 2003.
11. S. Lakare and A. Kaufman. Anti-Aliased Volume Extraction. In *Data Visualization 2003, Proc. of Eurographics/IEEE TCVG Visualization Symposium*, pages 113–122, May 2003.
12. A. Lefohn, J. Kniss, C. Hansen, and R. Whitaker. Interactive Deformation and Visualization of Level Set Surfaces Using Graphics Hardware. In *Proc. IEEE Visualization*, pages 75–82, 2003.
13. J. Ming, R. Machiraju, and D. Thompson. A Novel Approach to Vortex Core Detection. In *Proc. of Euro./IEEE Visualization Symp.*, pages 217–225, 2002.
14. V. Pekar, R. Wiemker, and D. Hempel. Fast Detection of Meaningful Isosurfaces for Volume Data Visualization. In *Proc. IEEE Visualization*, pages 223–230, 2001.
15. W. K. Pratt. *Digital Image Processing.* A Wiley-Interscience, second edition, 1991.
16. D. Prewer and L. J. Kitchen. Soft Image Segmentation by Weighted Linked Pyramid. *Pattern Recognition Letters*, 22(2):123–132, 2001.
17. K.-L. Ma R. Huang, P. McCormick, and W. Ward. Visualizaing Industrial CT Volume Data for Nondestructive Testing Applications. In *Proc. IEEE Visualization*, pages 547–554, 2003.
18. C. Rezk-Salama, K. Engel, M. Bauer, G. Greiner, and T. Ertl. Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage-Rasterization. In *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 109–118, 2000.
19. Y. Sato, C. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S Tamura, and R. Kikinis. Tissue Classification Based on 3D Local Intensity Structures for Volume Rendering. *IEEE Tran. on Visualization and Computer Graphics*, 6(2):160–180, 2000.
20. D. Silver and X. Wang. Tracking Scalar Features in Unstructured Datasets. In *Proc. IEEE Visualization*, pages 79–86, 1998.
21. M. Sramek and A. E. Kaufman. Alias-Free Voxelization of Geometric Objects. *IEEE Trans. on Visualization and Computer Graphics*, 5(3):251–267, July 1999.
22. Ikuko Takanashi, Shigeru Muraki, Akio Doi, and Arie Kaufman. 3D Active Net - 3D Volume Extraction. *Journal of the Institute of Image Information and Television Engineers*, 51(12):2097–2106, 1997.
23. S. Tenginakai, J. Lee, and R. Machiraju. Salient Iso-Surface Detection with Model-Independent Statistical Signatures. In *Proc. IEEE Visualization*, pages 231–238, 2001.
24. U. Tiede, T. Schiemann, and K. H. Höhne. High Quality Rendering of Attributed Volume Data. In *Proc. IEEE Visualization*, pages 255–262, 1998.



(a)



(b)



(c)

Fig. 8. (a)A feature volume of the Visible Human Foot with opacity and color transfer functions and a relatively simple hierarchical feature map for navigation. On the left, the user was painted the fibula green, and the rest of the skeleton white. On the right, the user decided to eliminate the occluding bone structures and just show the fibula, along with a faint outline of the tissues. (b) The Engine with one valve separated from the rings. (c) The lobster with one claw separated from the remaining body structures.