Feature-Preserving Volume Data Reduction and Focus+Context Visualization

Yu-Shuen Wang, Chaoli Wang, Member, IEEE, Tong-Yee Lee, Member, IEEE, and Kwan-Liu Ma, Senior Member, IEEE

Abstract—The growing sizes of volumetric data sets pose a great challenge for interactive visualization. In this paper, we present a feature-preserving data reduction and focus+context visualization method based on transfer function driven, continuous voxel repositioning and resampling techniques. Rendering reduced data can enhance interactivity. Focus+context visualization can show details of selected features in context on display devices with limited resolution. Our method utilizes the input transfer function to assign importance values to regularly partitioned regions of the volume data. According to user interaction, it can then magnify regions corresponding to the features of interest while compressing the rest by deforming the 3D mesh. The level of data reduction achieved is significant enough to improve overall efficiency. By using continuous deformation, our method avoids the need to smooth the transition between low and high-resolution regions as often required by multiresolution methods. Furthermore, it is particularly attractive for focus+context visualization of multiple features. We demonstrate the effectiveness and efficiency of our method with several volume data sets from medical applications and scientific simulations.

Index Terms—Data reduction, focus+context visualization, interactive visualization, mesh deformation, transfer functions, volume rendering.

1 INTRODUCTION

In visualization, the sizes of volume data we have been dealing with have increased dramatically over the years from 128³ to 1,024³ voxels or larger. The ever-increasing data size poses a great challenge to visual analysis in terms of both storage and rendering costs. To reduce storage cost, data compression may be used. However, the complication of runtime decompression could dramatically slow down rendering. To enable interactive visualization, rendering a reduced resolution of the data is commonly done before a desirable view and transfer functions are derived. Conventional multiresolution methods must cope with the costs of added storage space and removing cross-resolution boundary artifacts. In our work, we have developed a feature-preserving approach to volume data reduction that supports focus+context visualization. Our approach avoids runtime decompression while offering high rendering quality.

In this paper, we present the design and evaluation of our feature-preserving volume data reduction method. In contrast to downsampling, which uniformly discards information oblivious to the data content, our method more intelligently reduces data. The volume data are partitioned into cubic regions and each region is assigned an importance value based on the importance of the voxels in the region, which is hinted by the user-specified color and opacity transfer functions. As illustrated in Fig. 1, the regions with higher importance values, thus, containing features of interest, are magnified and other regions are compressed to retain the original volume boundary. As a result, the regions corresponding to the features of interest are populated with more samples whereas other regions are sparsely sampled.

With our approach, the original volumetric mesh is deformed and voxels are repositioned. The varying importance values introduce various region deformations, which could cause region intersections and distortion of large features. We apply edge flipping constraints to avoid region intersections, and introduce Laplacian smoothing to minimize the distortion around the features of interest. All these constraints are formulated into energy functions which are minimized using a global optimization system. After deforming the grid space, voxels within each region are trilinearly resampled on the GPU.

Our feature-preserving data reduction method has several advantages. First, the continuous space deformation guarantees similar resolutions along neighboring grid boundaries. This seamless deformation obviates the need of extra data packing or blending [35] to avoid gaps or to handle unmatched data resolutions among neighboring blocks, which are problems commonly found in multiresolution methods. Second, our method outputs a very simple data format: only a reduced data set and a deformed grid for shape recovery. At runtime, the full size volume can be directly rendered via texture lookup on the GPU. Since no data reconstruction is needed, significant memory saving is achieved. This approach, therefore, provides a cost-effective way for representing and rendering large volume data. Third, our solution may reduce the original volume to various sizes in a continuous fashion. The flexible
choice of volume data size draws a clear difference from building a multiresolution data hierarchy which often has the 2:1 ratio for each data dimension between two neighboring levels. Besides data reduction, our real-time voxel reposition technique also achieves focus+context visualization by rendering the features as the focus and the rest as the context. Unlike many previous methods [5], [31], [33] based on a magnifying lens which only allows a single focus at a time, our method automatically recognizes all features of interest for focus+context visualization. This characteristic is particularly useful for volumetric feature specification since features in volume data usually have complicated shapes and are surrounded by homogenous materials. In all, our method enhances our ability to visualize large, complex volume data.

2 RELATED WORK

2.1 Volume Data Reduction

Data reduction remains one of the important themes in the field of visualization as the size and complexity of data continue to increase at rapid rates. The simplest way of data reduction is uniform subsampling. Another way is to build a multiresolution data hierarchy and compress the data associated with each node in the hierarchy. This allows us to visualize data at different levels of detail and trade image quality for interactivity. Many algorithms have been developed to provide hierarchical data representation for volumetric data. Examples include the Laplacian pyramid [9], multidimensional trees [36], and octree-based hierarchies [2], [16]. The use of wavelet transform and compression for volumetric data was also introduced [24] and coupled with octree-based data compression and rendering [11]. Besides the octree structure, other researchers took a simpler scheme that constructs a flat block-based hierarchy [18], [20]. In that scenario, the entire volume is first subdivided into smaller subvolumes. Then, for each subvolume, coarser levels of the data are created by repeatedly filtering the data. In Section 5, we will compare this flat block-based multiresolution method with our data reduction method.

One complication with multiresolution schemes is the management of individual data blocks. It requires extra effort to either pack them into a single volume or render them separately in multiple passes and then compose all partial images to generate the final image. Another complication is rendering the transition between two neighboring blocks with different resolutions. To avoid seams between adjacent blocks, data along the block boundaries need to be replicated for correct interpolation [35]. In contrast, our deformation-based data reduction is much simpler. It generates a reduced volume along with a grid for shape recovery. There is no need for data block management and no special handling of mismatched resolutions. Moreover, our solution does not require explicit data reconstruction before rendering, which is necessary for many multiresolution methods.

There are several research efforts in volume visualization that incorporate transfer functions into multiresolution data compression and rendering [10], [20], [30], [37]. Their common goal was to adapt data precision and resolution according to the visualization content so that a better trade-off between data compression and rendering quality can be achieved. Our work shares similar ideas in terms of utilizing the transfer function to determine important regions. Unlike previous work, we magnify important regions to allocate more samples for preserving greater details of the features. We also allow the user to reverse this deformation process either directly or progressively at runtime. The reversal is very fast since the interpolation of each voxel value is independent and can be accelerated using the GPU.

2.2 Focus+Context Visualization

Focus+context techniques have long been used in visualization for interactive data exploration. In information
visualization, researchers have introduced fisheye view methods to magnify the focal area and either distort or overlay the neighboring regions to highlight the region of interest. For example, Carpendale et al. [5] presented several distortion patterns, such as stretch orthogonal and nonlinear radial, to demand more space for the focal region to achieve 3D viewpoint-independent distortion. Keahey et al. [12], [13], [14] deformed texts or 2D images by a transformation grid which is determined by nonlinear magnification fields. In volume visualization, Viola et al. [29] presented importance-driven volume rendering for automatic focus+context display of objects. They used predetermined object importance to encode voxel visibility priority. This information is used to guide rendering so that important regions are not occluded by unimportant ones. They also presented a technique for automatic focusing on features [28]. A focus is selected from a set of predefined features and their algorithm automatically determines the most expressive view of the features.

Researchers have also developed solutions for highlighting features in 3D volumetric data sets through deformation. McGuffin et al. [23] applied deformation techniques for volumetric data browsing. Their techniques allow the user to open up, spread apart, or peel away the outer layers to reveal hidden structures. Correa et al. [7] used physical and optical illustration operators to manipulate the geometry of data objects. Wang et al. [33] presented a focus+context technique based on an energy optimization model to magnify a region of interest for closer examination while deforming other regions without perceivable distortion.

Other researchers attempted volume deformation through the manipulation of optical attributes with a magnifying lens. For instance, LaMar et al. [17] deformed rendered 2D images or 3D volumes using hardware acceleration. Their approach dynamically computes texture coordinates for grid vertices and renders the texture with coordinates projected onto a homogeneous space to ensure desirable results. Wang et al. [31] presented an interactive focus+context technique for rendering volumetric models according to the optical lens theory. Their approach simulates the ray direction that is determined by the position of the focal point. The expanded image is displayed within the magnification lens. Although both [17] and [31] provide different shapes of bounded lenses for the user to magnify regions of interest, there is no guarantee of a nice fit into local feature shapes. In particular, the specification could be very inconvenient for interior features. This is not an issue for our approach with the assumption that features of interest are already well defined by the given transfer function.

3 Voxel Reposition

We deform the volume space to reposition the voxels. Important regions are magnified while unimportant regions are shrunk to maintain the original volume size. This idea is inspired by the resizing technique proposed by Wang et al. [32], [33], [34], and we revise it to meet our application needs. Specifically, the given volumetric data set is partitioned using a uniform grid \( G = \{ V, E, C \} \), where \( V = \{ v_0, v_1, v_2, \ldots, v_v \} \) denotes the vertex positions, and \( E \) and \( C \) are the set of edges and space cubes, respectively. We attempt to analyze the content of each local region and assign importance values to space cubes. While deforming the volume space, cubes with higher importance values deserve more samples to keep more information when the whole data set is downsampled or is rendered within a low-resolution display. In other words, cubes are resized according to their importance values. Unfortunately, the variation of cube deformations leads to some problems such as space intersection and shape variation between neighboring cubes. In contrast, the method presented in [33] does not have this problem since it handles polygonal models and cubes are partitioned into only two categories, making cube deformation relatively simple. In this paper, we smooth the deformation of neighboring cubes by reducing the change of the Laplacian coordinate of their shared grid vertex. We also introduce inequality constraints to prevent edges from flipping to avoid space intersection. These constraints are formulated into energy terms to serve the requirements and we strive to search for a deformed grid that minimizes our objective function. Finally, the voxels within the space cubes are trilinearly resampled based on space deformation [3].

3.1 Feature Specification

In volume visualization, the most common way to present features for a data set is through specification of the transfer function. For example, the user can specify color and opacity to highlight voxels of interest with distinct colors and high opacity values. As such, we use the following equation to compute the importance of a voxel \( p \):

\[
F(p) = \alpha(p) \cdot \| \nabla g(p) \|_F, \tag{1}
\]

where \( \alpha(p) \) is the opacity and \( \| \nabla g(p) \|_F \) is the Frobenius norm (magnitude) of the color gradient. To make this closer to human perception, we adopt the perceptually adapted CIELab color model for importance evaluation. We notice that certain saliency measures such as [15] can better identify features from the volume data set. However, our interactive system requires efficient computation and we found this simple method works well in most situations. Finally, each cube importance value is determined by averaging the interior voxel importance values and then normalized to \([0, 1]\) as a weighting factor used in (2).

3.2 Grid Space Deformation

3.2.1 Weighted Space Cube Expansion

We magnify the cubes \( c_k \) to emphasize the content within. Namely, \( c'_k = s c_k \), where \( s \) is a scaling factor specified by the user and \( c'_k \) is the deformed version of \( c_k \). To better utilize less important space and to minimize the distortion of important regions (see Fig. 2), we allow cube rotation by embedding the rotation matrix \( R' \) into the energy term

\[
D_f = \sum_{c_k \in C} \left( \lambda + w_k \right) D_f(c_k), \quad \text{where } D_f(c) = \| c' - s R' c \|^2, \tag{2}
\]

where \( w_k \) is a weighting factor representing the cube importance, \( \lambda \) is a small constant to avoid instability when \( w_k \) is very close to zero. With the fixed volume space, cubes with larger \( w_k \) are expanded due to the larger forces of magnification. Similar to the method presented in [33], we represent each local space cube with a set of vertices and edges. We then solve for each vertex position to fit the deformations of the cubes which share it. Specifically, we transform (2) into the following form:

\[
D_f(c) = \sum_{(i,j) \in E} | e'_{ij} - s R' e_{ij} |^2, \tag{3}
\]
achieve this constraint by the following energy term:

$$D_i = \sum_{v \in V} |L(v_i') - s_i R'_i(L(v_i))^2|,$$

$$L(v_i) = \frac{1}{|E(i)|} \sum_{\{i,j\} \in E(i)} v_i - v_j,$$

$$s'_i = \frac{1}{|C(i)|} \sum_{c \in C(i)} s'_c, \quad R'_i = \frac{1}{|C(i)|} \sum_{c \in C(i)} R'_c.$$

where $s'_i$ is the average of scaling factors and $R'_i$ is the rotation matrices averaged from the deformed cubes that share $v_i$. Note that the Laplacian coordinate $L(v_i)$ is rotation variant. The deformed cube sizes vary due to different weighting factors, even though cubes are scaled using the same transformation.

We constrain the boundary vertices moving along their respective planes in order to retain the size and shape of bounding space. Let $\partial V_x, \partial V_y$, and $\partial V_z$ be the boundary vertices in the $yz, xz$, and $xy$ planes, respectively. We solve for the deformed vertex positions by minimizing the energy terms $D_f + D_t$ subject to the volume boundary constraints

$$\begin{cases} v'_{i,x} = v_{i,x} & \text{if } v_{i} \in \partial V_x, \\ v'_{i,y} = v_{i,y} & \text{if } v_{i} \in \partial V_y, \\ v'_{i,z} = v_{i,z} & \text{if } v_{i} \in \partial V_z. \end{cases}$$

We also preserve neighboring cubes from intersecting each other using the edge flipping constraints. The deformed edge $e'_{ij}$ is required to have a similar direction to the original edge $e_{ij}$. Written in equation, each edge $\{i, j\}$ should satisfy

$$e'_{ij} \cdot e_{ij} > 0.$$

Since (3), (4), and (5) can be written as the linear combination of unknown vertex positions $V'$, we transform the objective function into a linear system $AV' = b(V')$, where $A$ represents the coefficients of unknown vertex positions, and $V'$ and $b(V')$ are the vectors in the right-hand side of simultaneous equations. We solve for the unknown vertex positions in a least square sense because the number of equations is much larger than that of the unknown vertex positions. Since the scaling factor $s'_i$ and the cube rotations $R'_i$ are unknown, we apply the Gaussian-Newton method [21], [22] to minimize this constrained nonlinear objective function in an iterative way. We solve for three coordinates of the vertex positions separately since the boundary constraints applied to the $x$, $y$, and $z$ coordinates are different.

Specifically, we consider the deformed scaling factor $s'_i$ and the rotation matrix $R'_i$ as additional unknown variables and update $V'$, $s'_i$, and $R'_i$ alternatively. The system starts from considering $s'_i = 1.0$ and $R'_i$ as an identity matrix and solves for vertex positions only. It then keeps updating $R'_i$, $S'_i$, and $V'$ until the system converges. To determine the unknown scaling factor $s'_i$ and the rotation matrix $R'_i$, we place the original and deformed cubes at the origin and compute a matrix that approximately transforms vertices of the original cube to those of the deformed cube [1]. The resulting transformation matrix is then decomposed into a rotation matrix $R$ and a shear matrix $S$ using polar decomposition. We set $R_e$ to $R$ and $s_e$ to the average of diagonal elements in $S$.

The edge flipping constraints are inequalities which make it impossible to solve using a linear system. Therefore, we detect if there are edges conflicting with (6) once the vertex positions $V'$ are updated. The flipped edges are enforced to be in the directions of their original versions. Specifically, we add the constraint

$$D_e(i, j) = \gamma |e'_{ij} - \delta e_{ij}|^2,$$

if the edge $\{i, j\}$ is flipped. Here, $\delta \geq 0$ is a parameter for preserving the space cube from being shrunk to zero size or even being negatively scaled. We set $\delta = 0.1$ and $\gamma = 10$ in our implementation. For those legal edges (i.e., $e'_{ij} \cdot e_{ij} > 0$), the edge flipping constraints are not necessary.

Equation (7) enforces the flipped edges to lie in their original directions. However, it is not always appropriate to hold these constraints since flipped edges may become legal if the user decreases the scaling factor or changes the transfer function. Constraining these edges to have nearly zero lengths would abnormally deform the cube, thus, resulting in more distortion to the data set. To avoid this, we check the edge length of $e'_{ij}$ and remove its flipping constraint when $|e'_{ij}| > \delta |e_{ij}|$. This is because we solve for the vertex positions using soft constraints, which means that we can only obtain the deformed edge $e'_{ij}$ with $0 \leq |e'_{ij} - \delta e_{ij}| \leq \mu$, where $\mu > 0$ is a very small number. $|e'_{ij}| > \delta |e_{ij}|$ indicates that other energy terms are squeezing the edge while $|e'_{ij}| > \delta |e_{ij}|$ indicates that they are stretching it. Since only squeezing may flip the edge, the flipping constraint is not needed in the stretching case.
rendering is a texture lookup to figure out where the voxel is stored. Specifically, in this example, the voxels A and B are stored in the positions of A’ and B’, respectively. The only additional computation of rendering is a texture lookup to figure out where the voxel is stored.

4 Applications

The voxel reposition technique brings us several applications in data reduction and rendering.

4.1 Data Reduction

We achieve data reduction through downsampling the deformed volume data set. The extracted voxels are trilinearly interpolated. Since features are magnified, more samples from those important regions would be picked up and then stored in the reduced data set. During rendering, the corresponding deformed grid is used to locate the queried voxel in the reduced data set. We show a 2D illustration to explain how our 3D recovering grid works in Fig. 3.

Compared to the direct downsampling method, our algorithm preserves features better since it does not simply uniformly discard voxels. Homogeneous regions are reduced more to leave more samples for features. Compared to multiresolution techniques, our system provides a much simpler data format for rendering. Rather than packing a sequence of downsampled textures, rendering our reduced data set requires only an additional texture lookup for obtaining the corresponding voxels. This fits nicely into the rendering pipeline since only the regular 3D data format is used. Furthermore, with continuous deformation between neighboring cubes, there is no need to smooth the boundaries between low and high-resolution regions. The above two advantages enable us to render the reduced data set efficiently. The computational cost is only slightly higher than that of rendering downsampled data sets. As illustrated in Fig. 4, the sampling resolutions along the boundary of more and less important space cubes are similar due to the shared boundary constraint.

4.2 Focus+Context Visualization

Our interactive deformation technique enables focus+context visualization. When visualizing a high-resolution data set on a low-resolution display, this interactive deformation system allows the user to take a closer look of selected features in context. Given the user-specified transfer function, our system magnifies regions with higher color gradient magnitudes and shrinks the rest so that the whole data set can be rendered within its original volume boundary. Our system also minimizes distortions of features, similar to the work in [33]. A significant limitation of [33] is that their discrimination of the focus and context regions is binary and, thus, is not sufficient to magnify features to different extents. Another major advantage of our system is that it magnifies the whole features without explicit feature specification. This is because transfer functions usually provide sharp color boundaries around features for effective visualization. Our system, thus, sets high importance values to space cubes covering the features. This distinction makes our feature specification easy and intuitive, especially for the interior and nonregular features (e.g., vessels) that are surrounded by homogeneous materials.

Our system is flexible with different kinds of transfer functions. For example, the applied transfer functions may consider voxel intensity, size [6], shape [25], etc. In addition, the system can take the combined multidimensional transfer function for better feature specification since the mapping to color and opacity from only a scalar value (usually intensity) may fail to capture features of interest. For focus+context visualization, since feature recognition is based on the perceptual color difference, our system responds to other types of criteria, such as color, opacity, and frequency, that highlight the focal regions. By employing these criteria, we emphasize the matched focal regions to attract attention.

4.3 Direct/Progressive Recovering

In addition to recovering a deformed data set back to its original shape in one step, we give the user an option to observe how the deformation is applied to the reduced data set. Let \( G^0 \) and \( G^f \) be data sets with the original and deformed shapes, respectively. The recovering grid \( T_{t\rightarrow0} \) transforms the deformed data set \( G^f \) back to its original shape \( G^0 \) (\( G^0 = T_{t\rightarrow0}G^f \)). By providing a list of grids \( T_{t\rightarrow0} \), where \( 0 \leq k \leq t \), we are able to show snapshots of the deformation process in a progressive manner. In this way, the user can intuitively understand how the deformation progresses. These grids also allow the user to achieve focus+context visualization with the specified transfer function.

5 Results

We have implemented our algorithm and show some of the test results in Figs. 1, 5, 6, 8, and 9 to demonstrate the effectiveness of our approach. The magnification results show focus+context visualization. They also show our
method can better preserve features and confirm the sampling between neighboring blocks is continuous. The features of interest are magnified so it is possible to visualize them clearly on a small display.

5.1 Data Reduction

We downsampled the deformed data set to preserve features. In Figs. 1, 5, and 8, we can observe that the data sets reduced using pure downsampling suffer from aliasing artifacts due to uniform loss of data content. In contrast, our method produced smoother results and preserved feature details better. We further compare our method with the Daubechies D4 wavelet transformation (Fig. 8) and multiresolution method [18], [20] (Fig. 9) for reducing large data sets to demonstrate the effectiveness of our method. To make a fair comparison, all methods reduce the data set to 1/64 of the original size. In addition, we determine the sampling rate of each cube in the multiresolution method according to its importance values. Clearly, wavelet reduction overly smoothes the features (see Fig. 8) while the multiresolution method produces discontinuity artifacts (see Fig. 9). Therefore, our method achieves the best results.

In this paper, we do not compare our method with widely used multiresolution methods since many of them have the same discontinuity problem due to their discrete nature. While interblock interpolation [19] can interpolate between sample values within and between blocks of arbitrary resolution levels to avoid discontinuity, its implementation is not trivial. If an additional smoothing process is applied as usual, it requires a large amount of additional space [8], [35]. Based on the formula given in [8], smoothing the boundaries for a $21 \times 21 \times 64$ resolution block requires nearly 11 percent of the original data set. In contrast, rendering our reduced data sets requires only the 3D positions of $22 \times 22 \times 65 = 30,976$ grid vertices for shape recovery. Specifically, in Fig. 9, additional 217 MB of data are required to fix the boundary discontinuity while our coarse deformed grid needs only 368 KB.

To measure data loss using different algorithms, we upsampled the reduced data sets to their original sizes by trilinear interpolation, followed by the comparison of color difference between corresponding voxels. Namely, the distortion at voxel $p$ is computed using the following equation:

$$\alpha(p)|C_o(p) - C_r(p)|^2,$$

where $\alpha(p)$ denotes the opacity of $p$, $C_o(p)$ and $C_r(p)$ are the colors of $p$ obtained from the original and reduced data sets, respectively. In the above equation, we multiply the color difference by the voxel opacity since visual distortion of a transparent voxel is unnoticeable. As shown in Table 1, with respect to the measured root mean square distortions, the multiresolution method has the minimum data loss. This is because the multiresolution method optimizes the available space to store important content. On the other hand, our method wastes some homogeneous regions to satisfy the space continuity. Although the distortions of the reduced multiresolution data sets are the minimum, they are gathered around block boundaries, which are sensitive to human perception. The artifacts can be smoothed but it requires extra effort to fix the boundary problem.

It can be observed that the overall distortion in our reduced hurricane data set is high due to continuous resampling on
highly squeezed regions, although the hurricane’s eye is well preserved. In Fig. 5, we show that the distortions of our reduced data set are mostly distributed further away from the hurricane’s eye, which is not important, and thus less noticeable. We do not measure the distortion of wavelet transformed data using root mean square since scalar data values are transformed to frequency values.

5.2 Focus+Context Visualization

Our voxel reposition system magnifies the focal region based on transfer functions. In Fig. 6, we adopt the size evaluation method presented by Correa and Ma [6] to magnify different parts of vessels using a 2D transfer function. By changing the vessel opacity according to the evaluated size, our system recomputes cube importances on the fly and magnifies specified features. From our accompanying video, it should become clear that our transfer-function-based feature recognition approach is effective and much easier to use than techniques based on magnifying lenses.

5.3 Performance

We run our system on a PC with a dual core 3.0 GHz CPU, 4 GB RAM, and an nVidia GTX 295 video card. Although the objective function is nonlinear and there are lots of unknown variables to be solved, our unoptimized code still achieves interactive performance. The timing result is reported in Table 2. As we can see, the computational cost of space deformation depends on grid complexity, where the slowest part is the minimization of the objective function. To trade quality for interactivity, we apply coarse grids for volume deformation. The grid resolutions we use are good enough for generating desirable results. On the other hand, the determination of cube importance depends on the data set itself. We can apply a downsampled data set to reduce the computation cost since only a scalar value from each cube is needed. This simplification brings little side effects to the results. Here, we render the volume data sets on an image of 640 $\times$ 480 with 500 samples along each pixel. Although the actual speed also depends on the viewing direction, we found that the rendering achieves 14 frames per second on average.

In Table 2, we list the computation time of space deformation for each iteration. In our experiment, the iteration number of minimizing the nonlinear objective function is usually less than five. It will get larger when more edges are flipped. However, we seldom give such a high scaling factor to the system since edge flipping implies that less important regions are squeezed dramatically. Increasing

<table>
<thead>
<tr>
<th>Table 1</th>
<th>The Root Mean Square Distortions of Downsampling, Multiresolution, and Our Reduced Data Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>data set</td>
<td>down sampling</td>
</tr>
<tr>
<td>aneurism (Fig. 1)</td>
<td>0.0222</td>
</tr>
<tr>
<td>aneurism (Fig. 6)</td>
<td>0.0201</td>
</tr>
<tr>
<td>hurricane (Fig. 5)</td>
<td>0.0009</td>
</tr>
<tr>
<td>viswoman (Fig. 8)</td>
<td>0.0348</td>
</tr>
<tr>
<td>plume (Fig. 9)</td>
<td>0.0186</td>
</tr>
</tbody>
</table>

The error is measured in the RGB color space.
the cube distortion: nonfeature regions, we apply the following term to measure unacceptable. To balance the quality between feature and zero size because totally missing that information is also also want to avoid squeezing all unimportant regions into since the bounding space is fixed. On the other hand, we ever, applying a very large scaling factor is not meaningful larger scaling factor would better preserve features. How-

improves the quality of rendering. This strategy also apply denoising methods such as bilateral filtering [27] to handle boundary discontinuity. We admit that multiresolu-

interesting features in data reduction, it obviates the need to system does not optimize the available space to preserve differences to measure the cube distor-

The price of preserving space continuity is not to squeeze all unimportant regions. This situation usually occurs between the focus and context regions. Although our system does not optimize the available space to preserve interesting features in data reduction, it obviates the need to handle boundary discontinuity. We admit that multiresolu-

We implemented our algorithm using the GPU to leverage its inherent parallelism. The tasks include the determination of cube importance, minimization of objective function, and raycasting of volumetric data sets. Even though the current generation of the GPU is already very powerful, handling all of the tasks still entails a heavy workload. With our current implementation, the CPU idles during the deformation process. In the future, we would like to balance the workload between the CPU and GPU so that better overall efficiency can be achieved.

6.1 Limitation

Our deformation technique magnifies each local region based on its importance value. This implies that the deformation fails to magnify the features if they are everywhere. In this scenario, our data reduction method would degenerate to pure downsampling and focus+context visualization becomes invalid. In addition, to preserve space continuity and uniform feature expansion, some homogeneous cubes would

\[
\frac{1}{n} \sum_{c \in C} w_k \| c_k - (s^t R^t)^{-1} c_k' \|^2, \tag{9}
\]

where \( n \) is the number of cubes and \( w_k \) denotes the cube’s importance value. We transform the deformed cube \( c_k' \) to have the same size and orientation as the original cube \( c_k \) and compare their differences to measure the cube distor-

Namely, the distance of corresponding vertices is accumu-

lated. We measure the distortion whenever the volume data set is further magnified and stop the magnification when the distortion increases rapidly. Instead of transforming the original cube to its deformed version, this strategy ensures that the measurement is determined with the same size. Due to the embedded weighting factor \( w_k \), the measured distortion would rapidly increase when the important cubes are distorted. This means most of the less important regions are already squeezed and we should stop the magnification. The diagrams in Fig. 7 show the relations between the distortion and the scaling factor for different data sets. The distortion \( \Psi \) always increases as the scaling factor \( s \) increases, while the partial derivative \( \partial \Psi / \partial s \) may decrease because the deformation is nearly terminated.

In this paper, we utilize a GPU implementation of the concurrent number cruncher (CNC) sparse solver [4] to solve the linear system. Given that the cube importance would be changed whenever the transfer function is adjusted, as well as new constraints would be added in when the edges are flipped, the coefficient matrix \( A \) may change frequently. The commonly used Cholesky direct solver is not sufficient in such a scenario. This is because the factorization step is very expensive and is necessary whenever the matrix \( A \) has been changed. Although the CNC solver is implemented with the conjugate gradient method, the computation is still efficient with the GPU speedup. Hence, the grid space deformation can be performed in real time, which is critical for interactive focus+context visualization.

6 DISCUSSION

Our system allows the user to adjust the scaling factor \( s \) to magnify interesting regions to different degrees. This interactive operator helps the user balance the importance of the focus and context regions. The scaling factor is usually larger than 1.0 since our goal is to magnify the features.

The cube importance is determined based on the color gradients of interior voxels, which highly depends on the input transfer function. That is, features with smooth boundaries may not be considered as important. Fortunately, in this scenario, blurred features are usually less important. Thus, reducing or distorting them is acceptable. On the other hand, if the data contain noise, the importance values of data cubes would increase since they have high color contrasts. To handle this, an easy and intuitive way is to apply denoising methods such as bilateral filtering [27] before computing the cube importance. This strategy also improves the quality of rendering.

For feature-preserving data reduction, it is clear that a larger scaling factor would better preserve features. How-

ever, applying a very large scaling factor is not meaningful since the bounding space is fixed. On the other hand, we also want to avoid squeezing all unimportant regions into zero size because totally missing that information is also unacceptable. To balance the quality between feature and nonfeature regions, we apply the following term to measure the cube distortion:
not be shrunk, which means that our system may not be able to fully utilize available regions for feature preservation (see Fig. 8, left). We will investigate how to fill out the unimportant regions with features as much as possible.

Our system deforms the input data set according to local information, i.e., the importance of space cube and the smoothness between neighboring cubes. Therefore, the system does not guarantee to preserve the global structure such as straight lines or symmetry. Although this issue does not affect data reduction, visualization of focus+context information may require this global structure to be retained. Fortunately, this is not a critical issue since our interactive system allows the user to see the change on the fly. The user can go back and forth to observe spatial structures and relationships. We will develop an automatic method to detect global structures and enforce space cubes covering the corresponding features to have similar deformations.

### 7 Conclusion

We have introduced the concept of feature-directed voxel repositioning for interactive focus+context volume visualization. Our design not only enables better utilization of both screen and storage spaces but also offers rendering quality
higher than that of previous methods. Features of interest are suggested by a user-defined transfer function. No additional operations by the user are required. To examine different aspects of the data, the user may change feature specification any time by interactively modifying transfer functions. No expensive data preprocessing and waiting are needed. According to the feature specification, our system can automatically magnify the feature to show greater details in context, which is convenient and intuitive to the user. We achieve high interactivity with GPU acceleration of compute-intensive steps, and high quality with continuous sampling a deformation grid of the original volume. Our test results clearly verify our design.

We have shown a prototype system of our design to a group of surgeons, and they confirmed the value of the quality and interactivity of focus-context visualization offered by our system. We, therefore, plan to continue our work and involve these surgeons in the development and evaluation of a production-level system.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments. They also thank Jeremiah Caron for the video narration. Yu-Shuen Wang and Tong-Yee Lee are supported by the Landmark Program of the NCKU Top University Project (contract B0008) and the National Science Council (contracts NSC-97-2628-E-006-125-MY3, NSC-96-2628-E-006-200-MY3, and NSC-99-2221-E-006-066-MY3), Taiwan. This research was supported in part by the US National Science Foundation through grants OCI-0325934, OCI-0749217, CNS-0515727, CCF-0811422, OCI-0749227, OCI-0950008, CCF-0938114, and OCI-0850566, and the US Department of Energy through the SciDAC program with Agreements No. DE-FC02-06ER25777 and DE-FG02-08ER54956.

REFERENCES


