Visualization-Specific Compression of Large Volume Data*

Chandrajit Bajaj
Dept. of Computer Sciences
The Univ. of Texas at Austin
U.S.A.

Insung Ihm
Dept. of Computer Science
Sogang University
Korea

Sanghun Park
TICAM
The Univ. of Texas at Austin
U.S.A.

Abstract

When interactive real-time applications are developed with very large volume data, the use of lossy compression is often inevitable. Lossy compression schemes generally encode data without consideration of the purpose of visualization that is actually performed, which often results in inefficient compression. In this paper, we present a new method for classifying voxels according to their importance in visualization, and assigning appropriate weights to them. The associated weight information can be combined with lossy compression schemes to reduce the visual degradation of reconstructed images, resulting in higher compression rates and visual fidelity. Test results demonstrate that the proposed technique improves both the amount of compression and the quality of visualization significantly.

1. Introduction

Volume visualization is a research area that deals with various techniques for extracting meaningful and visual information from abstract and complex volume data sets. Effective representation, whether it is exact or approximate, of various volume data sets has been one of the most critical research problems in volume visualization, and a wide variety of approaches have been developed.

When volume data are very large, as in many recent applications, the use of compression techniques is often inevitable. Many conventional two-dimensional image compression schemes can be effectively extended to three- or higher-dimensional volume data sets. When interactive real-time applications are developed, however, the compression scheme should be chosen carefully. For instance, the methods based on such filters as discrete cosine transform or Daubechies wavelets show excellent performance in both compression rates and image fidelity. On the other hand, they are often inappropriate for real-time applications that require fast decompression for random data access.

In this paper, we focus on compression schemes suitable for real-time applications, although the ideas presented here can be extended naturally to other compression methods. Vector quantization has been very popular in designing real-time applications, as it provides probably the fastest decoding speed. Some recent applications of vector quantization in the computer graphics field include compression of CT/MRI datasets [14], light fields [8], and 2D textures [2].

The idea of using wavelets, initiated by Muraki [12], has been applied to efficient representation and approximation of volume data by many researchers, including [18, 13, 3, 16, 20]. Recently, a new wavelet-based compression scheme for 3D RGB and gray-scale images, called *zerobit encoding*, was developed for applications wherein data items are accessed in an unpredictable manner, and real-time performance of decoding is required [1]. The encoding scheme, based on 3D Haar wavelets, naturally offers a multi-resolution representation for 3D images, and experimental results on medical images, light- field images, and 3D textures show that it provides fast random access to compressed data, in addition to achieving fairly high compression rates. In spite of the fast random access ability of these vector quantization and wavelet-based compression schemes, they have the disadvantage that their filtering capability is not as good as other frequently used compression techniques, which may cause error-prone visual information.

In many visualization applications, one often wishes to visualize some fixed set of coherent features that are concentrated in a few regions. For instance, isosurfaces with a few isovalues are built during isosurface extraction. In direct volume rendering, regions of voxels with density values in a range of interest are usually rendered. If the features to be examined are known beforehand, it is possible to utilize the relevant information during compression to preserve them as precisely as possible. Lossy compression can

*C. Bajaj's and S. Park's research was supported in part by the National Science Foundation under grants ACI-9982297 and SBR-9873326. I. Ihm's research was supported in part by the the Ministry of Information & Communication of Korea under the University Foundation Research Program 2000 grant, and by the Sogang University Research grants in 2001.

be viewed as allocating a limited amount of resources to voxels, and it is the main goal of this work to distribute those resources wisely, according to the specific types of visualization tasks to be performed.

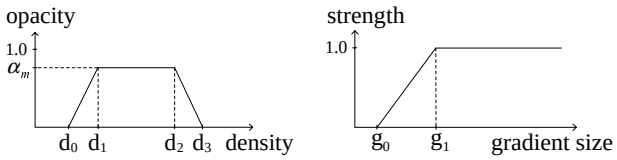
In this paper, we present a technique that classifies voxels in accordance with the purpose of visualization, and that assigns appropriate weights to voxels. The associated weights are then combined with compression algorithms so that important features remain as correct as possible after reconstruction. Contrary to the standard lossy compression schemes, in which information on features is *uniformly* lost regardless of the visualization to be performed, the new scheme enables one to focus more on important features that are more frequently used during the visualization. This approach is different from other enhancement techniques such as the classified vector quantization [15], the classification algorithm [6], or the multi-level techniques [4, 19, 10, 17] that attempt to improve the approximation or classification quality based on spatial properties of images themselves.

This paper is organized as follows: In Section 2, two popular visualization methods that are considered in this paper are summarized. In Section 3, we explain how voxels are classified and weighted according to their importance in interesting visualization. The compression scheme [1] based on the 3D Haar wavelet transform is then taken as an example to explain how the simple Haar filters are enhanced without any tradeoff in Section 4. Experimental results are shown in Section 5, and the paper is concluded in Section 6.

2. Isosurface extraction and direct volume rendering

In this paper, two fundamental types of visualization algorithms are considered. First, we deal with isosurface extraction based on the marching-cube algorithm [9]. During construction of isosurfaces, vertices of generated triangles are computed by determining all edges of cells in volume data which intersect the isosurfaces. The edge intersections are usually interpolated linearly from the densities of two incident voxels of the edges. Furthermore, normal vectors are also linearly interpolated from the voxels incident to the edges. While there are other better approximation methods [11], we assume that gradients at voxels are approximated using computationally simple central differences. Thus the correctness of isosurfaces with normals are dependent on the densities of incident voxels of all intersecting edges, and their 6-neighbors.

In addition to the indirect volume rendering, we take into account a direct volume rendering. In particular, we focus on the ray-casting algorithm [7]. While our method is designed for ray-casting, it is easily modified for other popular direct volume rendering methods such as splatting. When a ray is cast, the shaded color-opacity pairs are computed at resampling points on the ray, and are accumulated in front-to-back order. In classifying materials, we use a model similar to one employed in [5]. The opacity transfer function for a material m is defined by its opacity α_m and a quadruple $[d_0, d_1, d_2, d_3]$. A voxel v has positive opacity value $T_{opac}(v)$, determined by a ridge-shaped function, only when its density is between d_0 and d_3 (Figure 1(a)). Gradient at v is also an important factor in classification, and its magnitude indicates how close the voxel is to boundaries. In order to emphasize boundaries of materials in rendering, we use $\alpha(v) = T_{opac}(v) \cdot T_{grad}(v)$ as its computed opacity at v, where $T_{grad}(v)$ is a *normalized* magnitude of gradient. The gradient transfer function is defined on rescaled gradient magnitude of voxels, and its function value is determined by a pair of numbers $[g_0, g_1]$ as in the figure (b). When a point in a cell is resampled, its color and opacity affect accumulation only when the computed opacity of at least one voxel is greater than zero. In this case, the densities of its eight voxels and all their 6-neighbors are used in ray-casting, hence they need to be preserved well when a lossy compression is applied to volume data.



(a) Opacity function (b) Gradient function

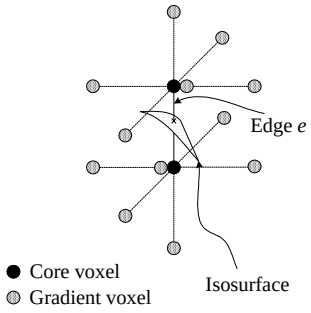
Figure 1. Transfer functions

3. Voxel classification and weights

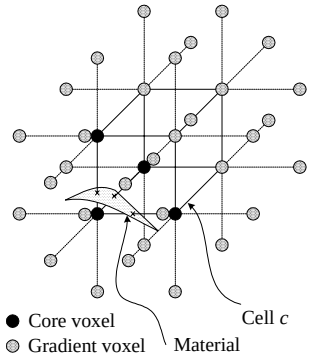
3.1. Core voxels and gradient voxels

Consider a 3D volume data set, defined on a rectilinear Cartesian grid. Let V be the set of all voxels v in the data on which a real-valued scalar density field $d(v) : V \rightarrow R$ is defined. The rectilinear grid can be regarded as a 3D graph where voxels form a vertex set V, in which each voxel is connected to its 6-neighboring voxels. All the unordered pairs (u, v) of adjacent voxels are *edges* whose set is denoted by E. A cell c is a cube of 8 adjacent voxels, and the cell set is represented by C.

Let $D_{is} = \{d_{is}^i \mid i = 0, 1, \cdots, p-1\}$ be a set of p density values. Also, define $D_{rc} = \{[d_{rc}^{i0}, d_{rc}^{i3}] \mid i = 0, 1, \cdots, q-1\}$ to be a set of q intervals of density values. D_{is} contains p interesting isovalues whose corresponding isosurfaces may be examined during visualization. On the other hand, D_{rc} includes density ranges of q interesting materials that may be rendered in ray-casting. A set of q opacity functions $\alpha_i(v)$,



(a) Isosurface extraction



(b) Ray-casting

Figure 2. Core voxels and gradient voxels

$i = 0, 1, \cdots, q-1$ are defined for the q materials, respectively, as explained in Section 2.

For isosurface extraction, define a voxel set $V_{is}^c(D_{is}) = \{v \in V \mid d_{is}^i \in I_d(v,w)$ for some $d_{is}^i \in D_{is}$ and $w \in N_6(v)\}$, where $I_d(v,w) = [min(d(v), d(w)), max(d(v), d(w))]$, and $N_6(v)$ consists of all 6-neighbors of v. A voxel is contained in this set only if at least one interesting isosurface intersects any of its 6 incident edges. Similarly, we define the corresponding voxel set for ray-casting as $V_{rc}^c(D_{rc}) = \{v \in V \mid \sum_{i=0}^{q-1} \alpha_i(v) \geq \alpha_\epsilon$ for some $\alpha_\epsilon > 0\}$, where α_ϵ is a small positive threshold such as, for instance, 0.05. A voxel is included in this set only if at least one interesting material passes through any of its incident cells, and its computed opacity, added for all specified materials is large enough to impact on color accumulation in ray-casting. The two sets $V_{is}^c(D_{is})$ and $V_{rc}^c(D_{rc})$ contain all the voxels that directly affect isosurface extraction and/or ray-casting, represented by D_{is} and D_{rc}. That is, the densities of voxels in their union $V^c(D_{is}, D_{rc}) = V_{is}^c(D_{is}) \cup V_{rc}^c(D_{rc})$ greatly influence the fidelity of isosurface extraction and/or ray-casting, hence must be given highest priority in any lossy compression. We call these *core voxels*.

In addition to core voxels, we consider *gradient* voxels. Figure 2 depicts how the two visualization schemes handle gradient voxels separately. First, when an isosurface crosses an edge *e* as in (a), gradients at its two incident core voxels must be computed to approximate the gradient at the intersection point. Since the central difference formula are used in gradient approximation, the densities of all 6-neighbors of the two voxels need special treatment during lossy compression. Let $V_{is}^g(D_{is}) = \{v \in V \mid w \in V_{is}^c(D_{is})$ for some $(v,w) \in E\}$. This is the set of all voxels that are adjacent to some core voxels, and are used in approximating gradients at core voxels during isosurface extraction.

For ray-casting, more voxels are usually put into its gradient set. Consider a cell c contains at least one core voxel as in (b), which means that an interesting material having nontrivial computed opacity, penetrates c. In this case, gradients at arbitrary points inside c are necessary for resampling shaded colors and opacities. They are generally approximated by tri-linearly interpolating the gradients at c's eight voxels, which are, in turn, approximated using the central differences. That is, densities of all 6-neighbors of the eight voxels are used in computing gradients inside c. The set of all such gradient voxels can be expressed as $V_{rc}^g(D_{rc}) = \{v \in V \mid v \in N_6(w)$ for some $w \in V(c)$ such that $u \in V_{rc}^c(D_{rc})$ for some $u \in V(c)\}$, where $V(c)$ is a set of eight voxels of cell c. Since the direction and magnitude of approximated gradients play important roles in shading and classification, the density values of voxels in the gradient set $V^g(D_{is}, D_{rc}) \equiv V_{is}^g(D_{is}) \cup V_{rc}^g(D_{rc})$ must also be handled carefully.

Lastly, the voxels that are neither core nor gradient, are classified as *unimportant*. This term is conveniently used to indicate that these voxels are simply not used in visualization, expressed by D_{is} and D_{rc}. For completion of definitions, we define the unimportant voxel set as $V^u(D_{is}, D_{rc}) = V - (V^c(D_{is}, D_{rc}) \cup V^g(D_{is}, D_{rc}))$. When the context is clear, we drop the arguments (D_{is}), (D_{rc}), and (D_{is}, D_{rc}) in denoting the various sets defined in this subsection.

3.2. Definition of weight function $\phi_v(v)$ for voxels

When D_{is} and D_{rc} are given, voxels in the volume data belong to one or more of V^c (core), V^g (gradient), and V^u (unimportant). Notice that V^c and V^g are not mutually exclusive, and a voxel can be in more than one core or gradient voxel sets. The classified voxels are weighted according to their possible usage, or importance in visualization. The idea is to regard voxels in either V^c or V^g as important, and to keep the fidelity of their densities as much as possible during any lossy compression.

First, let's consider the voxels in V_{is}^c. As mentioned before, they are incident to at least one edge that is intersected by any isosurfaces being considered. Our main concern is to be able to compute intersection points along crossing edges as correctly as possible from reconstructed volume data. At first, it might seem to be a good idea to assign a larger weight to a voxel closer to intersection point. In extracting isosurfaces, however, the intersection along an edge is usually approximated using linear interpolation. This implies that both densities of incident voxels are equally important. Instead of assigning a uniform weight to core voxels, we give larger weights to voxels whose incident edges are cut through by more isosurfaces. Notice that densities of such voxels are used more frequently in computing intersection points, and contribute to building isosurfaces to a larger extent. Since the correctness of isosurface models mainly depends on the position of vertices, it is reasonable to emphasize them to build isosurfaces with higher fidelity from compressed volume data.

Let $(V_{is}^c)^i$, $i = 0, 1, \cdots, p-1$, be the subsets of V_{is}^c, made of core voxels relative to the ith material only. For a voxel $v \in (V_{is}^c)^i$, the number of its incident edges crossed by isosurface of the ith isovalue is expressed by the following function:

$$\delta_{is}^i(v) = \begin{cases} |\{w \,|\, w \in N_6(v) \cap (V_{is}^c)^i\}|, & \text{if } v \in (V_{is}^c)^i, \\ 0, & otherwise. \end{cases}$$

Then the weight function $\phi_{is}^c(v)$ for $v \in V_{is}^c$ is defined as

$$\phi_{is}^c(v) = \max_{0 \le i \le p-1} \delta_{is}^i(v).$$

We take a conservative approach in defining the weight functions by selecting the maximum of weights rather than adding them. That is, if a voxel is important to at least one isosurface or material, it is considered to be a significant voxel. The weight indicates how much a voxel can affect the correctness of an isosurface of D_{is} across its vicinity, and its maximum possible value is 6.

Next, consider how the core voxels in V_{rc}^c are weighted for ray-casting. Their densities are used in direct volume rendering, and we intend to preserve well the voxels' densities that actually contribute to color-opacity accumulation during rendering. Since the function $\alpha_i(v)$ shows how greatly v's density may impact on image composition relative to the ith material, the weight function $\phi_{rc}^c(v)$ is naturally defined as follows:

$$\phi_{rc}^c(v) = \max_{0 \le i \le q-1} \alpha_i(v).$$

Notice that it has values at most $\max_{0 \le i \le q-1} \alpha_{mi}$, where α_{mi} is the maximum opacity of the ith material.

The gradient voxels are also weighted by their usage. Consider a voxel $v \in V_{is}^g$. Its density is used when the gradients at v's 6-neighbors need to be approximated. Since any reconstruction error, associated with v, could be propagated in computing normal vectors near v, the density becomes more significant as more isosurfaces pass through the edges incident to the neighbors. Note that the usage of v for the ith isosurface is expressed by $\sum_{w \in N_6(v)} \delta_{is}^i(w)$. Hence, we define the weight function $\phi_{is}^g(v)$ for gradients voxels of isosurface extraction by taking the maximum usage for all isosurfaces:

$$\phi_{is}^g(v) = \max_{0 \le i \le p-1} \sum_{w \in N_6(v)} \delta_{is}^i(w).$$

Lastly, the weight function $\phi_{rc}^g(v)$ for ray-casting is similarly defined. Recall the definition of gradient voxels (Figure 2(b)). A voxel $v \in V_{rc}^g$ is possibly used in the gradient computation for all cells incident to the 6-neighbors of v. Its density value is used only in the cells that have at least one voxel with nontrivial computed opacity. If $C(v)$ is all eight cells incident to v, the number of cells that need the density of v during gradient approximation for the ith material can expressed by another function $\delta_{rc}^i(v)$, where $\delta_{rc}^i(v) = |\{c \in C \,|\, c \in C(w) \text{ for some } w \in N_6(v) \text{ such that } u \in V_{rc}^c \text{ for some } u \in V(c)\}|$. Then the importance of a gradient voxel v is defined as follows:

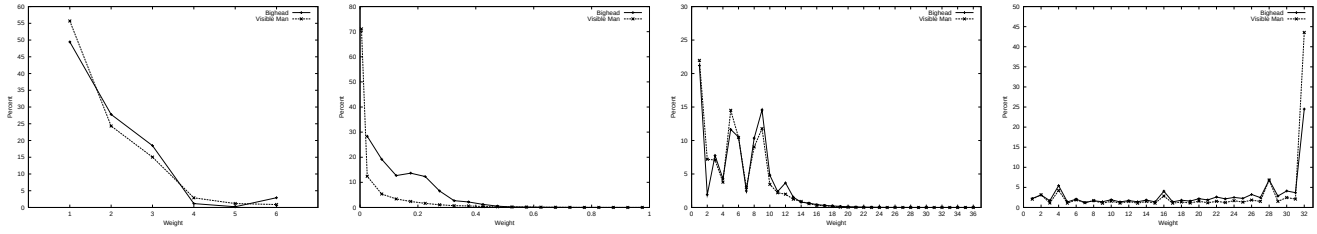
$$\phi_{rc}^g(v) = \max_{0 \le i \le q-1} \delta_{rc}^i(v).$$

It is not hard to see that $\phi_{is}^g(v)$ and $\phi_{rc}^g(v)$ have values at most 36 and 32, respectively.

Now the weight function $\phi_v(v)$ for all $v \in V$ is a function $f(\phi_{is}^c(v), \phi_{rc}^c(v), \phi_{is}^g(v), \phi_{rc}^g(v))$ of the above four functional values. Notice that the functions $\phi_{is}^c(v)$, $\phi_{rc}^c(v)$, $\phi_{is}^g(v)$, and $\phi_{rc}^g(v)$ are defined in the domains that are different to each other. So, they must be normalized properly when combined into one function. One problem to be solved here is how to choose a proper set of the four constant values. Since they are dependent on volume data, and a variety of visualization parameters such as D_{is} and D_{rc}, the data must be analyzed to determine them empirically.

3.3. A sample example of weight functions

We tested with the UNC Bighead data ($256 \times 256 \times 225$) in order to illustrate how voxels are weighed in our scheme (Figure 8(a)). In this example, we are interested in visualizing both bone and skin, where $D_{is} = \{66, 160\}$ and $D_{rc} = \{[40, 90], [95, 255]\}$ include one element per material, respectively. For ray-casting, we used opacity transfer functions $[40, 45, 88, 90]$ and $[95, 135, 253, 255]$ for the two materials, respectively, and a gradient transfer function $[0, 95]$ for rescaled gradients (Refer to Section 2.). Figure 8(b) to (e) show weighted images for the 114th slice that are colored using a color map where the maximum weights in the slice are normalized to 1.0.



(a) $\phi^c_{is}(v)$ (b) $\phi^c_{rc}(v)$ (c) $\phi^g_{is}(v)$ (d) $\phi^g_{rc}(v)$

Figure 3. Distribution of weights for Bighead and Visible Man data

An analysis of the weight distribution reveals some typical patterns of the four weight functions as shown in Figure 3(solid lines), where the percentages in the captions are the ratios of voxels having nonzero weights for the corresponding functions. In combining weights, we first rescale them, according to their distributions, using a function $s[x_0, x_1, f_0, f_1](x)$ defined as follows:

$$s[x_0, x_1, f_0, f_1](x) = \begin{cases} 0.0, & \text{if } x < x_0, \\ f_1, & \text{if } x \geq x_1, \\ f_0 + \frac{(f_1 - f_0)\cdot(x - x_0)}{x_1 - x_0}, & \text{otherwise.} \end{cases}$$

In combining the rescaled weights, we use a heuristic method where the core and gradient weights are added for the two visualization techniques, respectively, and then their maximum is taken as the computed weight of a voxel. The exact equation we used in this example is as follows: $\phi_v(v) = max\{\kappa_{is} \cdot (s[1, 4, 0.7, 1.0](\phi^c_{is}(v)) + s[1, 20, 0.2, 0.5](\phi^g_{is}(v))),\ \kappa_{rc} \cdot (s[0.05, 0.6, 0.4, 1.0](\phi^c_{rc}(v)) + s[1, 30, 0.1, 0.5](\phi^g_{rc}(v)))\}$. Here, κ_{is} and κ_{rc} control the relative strength of isosurface extraction and ray-casing, respectively. Notice that $V^c_{is} \subset V^g_{is}$ and $V^c_{rc} \subset V^g_{rc}$. The idea we rely on is to emphasize core voxel regions more over gradient voxel regions. Another natural way of combining them is to add them rather than taking their maximum. For the Bighead data, the maximum weight performs slightly better. Figure 8(f) and (g) show the sample slices for both cases with $\kappa_{is} = \kappa_{rc} = 1.0$.

4. Application to compression scheme

In the previous section, we discussed the function $\phi_v(v)$ that weighs voxels according to their importance. The weight information can be utilized by lossy compression schemes so that the loss of information is minimized with respect to a specified visualization task. As an application, we combine the weighting mechanism with a compression scheme that are based on the simple 3D Haar wavelets [1]. The Haar filters are attractive in developing real-time applications since they are computationally very cheap. However, it suffers from their poor filtering capability. In this section, we explain how the weights of voxels can enhance the 3D Haar wavelets without any tradeoff in the process of visualization.

4.1. Definition of weight function $\phi_w(w)$ for wavelet coefficients

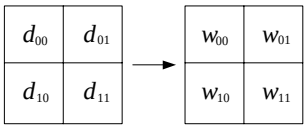
For the sake of simplicity of explanation, consider the 2D Haar transform as given in Figure 4(a). When densities of voxels in a 2×2 area are decomposed into the wavelet space by the Haar filters, the same number of wavelet coefficients are generated. We adopt the convention that the produced wavelet coefficients are stored in the same region in the image (Figure 4(b)). In this figure, w_{00} is the average of voxels, and w_{01}, w_{10}, and w_{11} are the detail coefficients. To reconstruct the original voxels, the inverse Haar transform is applied to these coefficients.

For a 4×4 square, the Haar transform is first applied to its four 2×2 subsquares, resulting in four sets of wavelet coefficients. Then the same transform is repeatedly applied to the four averages where the transformed coefficients are stored as depicted in Figure 4(c). The 16 coefficients in the region after two consecutive applications of the forward transform can be organized in a hierarchy, called a *decomposition tree*, depicted in Figure 4(d), which consists of an average w_{00}, one set of detail coefficients $\{w_{0j} \mid j = 1, 2, 3\}$ on level 0, and four additional detail sets $\{w_{ij} \mid j = 1, 2, 3\}$, $i = 1, 2, 3, 4$ on level 1 that correspond to the four 2×2 subsquares.

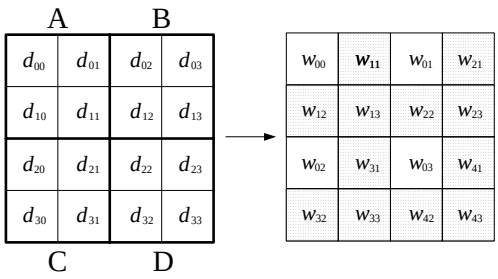
Next consider how weights assigned to voxels are associated with wavelet coefficients. The function $\phi_v(v)$ indicates the importance of a voxel in the data space and its meaning must be translated into the wavelet space properly. When four voxel densities are decomposed, the wavelet coefficients, stored in the same area, contain the necessary information for reconstructing the original voxels. The trans-

$$
\begin{aligned}
w_{00} &= (d_{00} + d_{01} + d_{10} + d_{11})/4 \\
w_{01} &= (d_{00} + d_{01} - d_{10} - d_{11})/4 \\
w_{10} &= (d_{00} - d_{01} + d_{10} - d_{11})/4 \\
w_{11} &= (d_{00} - d_{01} - d_{10} + d_{11})/4
\end{aligned}
$$

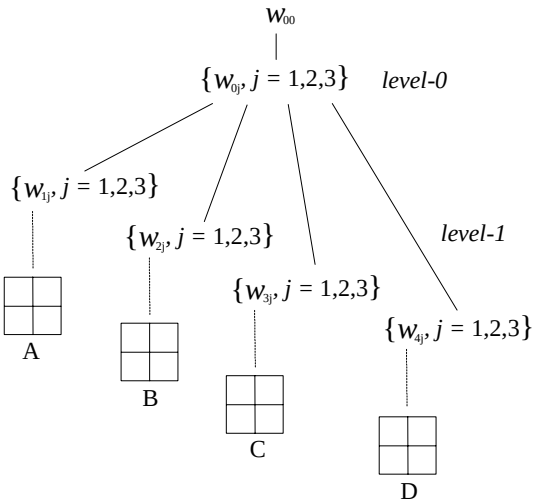
(a) 2D Haar transform



(b) Rearrangement of coefficients



(c) two level applications



(d) Decomposition tree

Figure 4. 2D Haar decomposition

formed coefficients are important if the original densities in the region are important. One possible way to connect the importance measures of voxels and wavelets is that the four wavelet coefficients have the same weight which is the average of the four voxels' weights. This seems natural, however, notice that all the four wavelet coefficients are used simultaneously when any voxel in the area is reconstructed. That is, the accuracy of reconstructed voxels is determined by the accuracy of all the four wavelet coefficients. In order to guarantee the accuracy of voxel with maximum weight, it is more reasonable to conservatively assign the largest weights of the four voxels to the four wavelet coefficients.

When the Haar transform is applied twice as in Figure 4(d), the weights of 16 coefficients are determined by traversing the tree in bottom-up fashion. First, the largest of the four voxels in a 2×2 area becomes the weights of wavelets in the corresponding level-1 detail node. Then, their maximum value which indicates the importance of the entire 4×4 region is assigned to the average and detail wavelets on level 0.

The basic idea is naturally extended into 3D space. Consider a $4 \times 4 \times 4$ region, called a *cell*. Voxel densities in a cell are decomposed by the 3D Haar transform. The transformed cell is represented similarly in a decomposition tree that are made of one average node w_{00}, one detail node $\{w_{0j} \mid j = 1, 2, \cdots, 7\}$ on level 0, and 8 detail nodes $\{w_{ij} \mid j = 1, 2, \cdots, 7\}$, $i = 1, 2, \cdots, 8$. Note that the averages of the 8 $2 \times 2 \times 2$ subregions are implicitly represented in the decomposition tree, and they can be reconstructed using the average node and the detail node on level 0. The entire voxel densities are then reconstructed using the eight averages and eight level-1 detail nodes.

Now we are ready to define the weight function $\phi_w(w)$ for a transformed coefficient w in the wavelet space. Let v_{ik}, $k = 1, 2, \cdots 8$, be the 8 voxels in the ith $2 \times 2 \times 2$ region ($i = 1, 2, \cdots, 8$). The same weight is assigned to all wavelets in the corresponding ith detail node as follows:

$$\phi_w(w_{ij}) = \max_{k=1,2,\cdots,8} \phi_v(v_{ik}) \text{ for } j = 1, 2, \cdots, 7.$$

Then, they are propagated upwards to the detail coefficients on level 0 in the following way:

$$\phi_w(w_{0j}) = \max_{k=1,2,\cdots,8} \phi_w(w_{k1}) \text{ for } j = 1, 2, \cdots, 7.$$

Finally, this weight is assigned to the average node w_{00}, that is, $\phi_w(w_{00}) = \phi_w(w_{01})$.

4.2. Truncation of wavelet coefficients

The theory behind wavelet compression tells that the best way to pick a fixed number of wavelet coefficients when an orthogonal basis is used, making the resulting error in the

L^2 norm as small as possible, is simply to select coefficients with the largest norms, and replace the rest by null values. The original information is thus approximated by a smaller number of nonzero wavelet coefficients. In the compression scheme [1], the 3D Haar transform is applied twice only since a smaller number of applications of inverse transform results in a faster reconstruction, and most of the data $\frac{63}{64}$ $(= 1 - (1/8^2))$ is already decomposed into wavelets. In the process of truncation, we modulate the magnitude of wavelets using their weights so that the important regions can be reconstructed faithfully. For a wavelet w_{ijk}, we use $\mid \nu(\phi_w(w_{ijk})) \cdot w_{ijk} \mid$ instead of $\mid w_{ijk} \mid$ as its modified norm. Here the function $\nu(*)$ rescales the wavelet's weight based on the distribution of wavelet coefficients. In our work, we use a function in the form of $\nu(\phi_w(w)) = \gamma_0 + \gamma_1 \cdot \phi_w(w)$, where γ_0 and γ_1 are determined experimentally.

5. Results

5.1. Test with the Bighead data set

We continue to illustrate the effectiveness of the proposed method on the UNC Bighead data. We produced two sets of reconstructed volumes using 2%, 3%, 5%, and 7% of wavelet coefficients, respectively. In generating the first set, the magnitude of the wavelets was used without alteration in the truncation step. The second set was generated by applying weights, as explained in the previous section, with rescaling factors $\gamma_0 = 1.0$ and $\gamma_1 = 10.0$ that are chosen empirically through the analysis of distribution of wavelets coefficients. Statistics on our objective fidelity criteria are summarized in Table 1(a) for the two cases. The root mean squared error (RMSE) is the square root of the average of the squared error measure, and it is one of the most often used average measure. As another indicator of image quality, we also present the peak-signal-to-noise ratio PSNR (dB) that measures the size of the error relative to the peak value of the signal. In calculating them, we considered only the important voxels that have positive weights, in other words, that are possibly used in the visualization specified by D_{is} and D_{rc}. The rate of important voxels is 19.89% for the Bighead data. It is clearly seen that the quality of the weighted Haar is significantly better than the quality of the unweighed Haar.

In Figure 5, we enlarged a portion of the 114th slice images reconstructed with two wavelet ratios. The quality turns out to be always superior when wavelets are weighted. When the ratio is less than 5%, blocky artifacts are clearly visible in both methods. When more than 5% of wavelets are used, reconstructed images using weights are almost free of aliasing artifacts in the important regions. We observe that the weighting method is very effective in improving the poor filtering quality of the simple 3D Haar transform.

		Ratio of the used wavelets			
		2%	3%	5%	7%
Unweighed Haar	RMSE*	10.78	8.76	6.38	4.91
	PSNR	27.48	29.28	32.04	34.31
Weighted Haar	RMSE*	6.78	4.86	2.82	1.79
	PSNR	31.50	34.40	39.12	43.06

(a) Bighead (*: density range $0 - 255$)

		Ratio of the used wavelets			
		2%	3%	5%	7%
Unweighed Haar	RMSE**	66.61	51.55	36.02	27.10
	PSNR	35.77	38.00	41.11	43.58
Weighted Haar	RMSE**	46.92	34.21	20.88	13.92
	PSNR	38.82	41.56	45.85	49.37

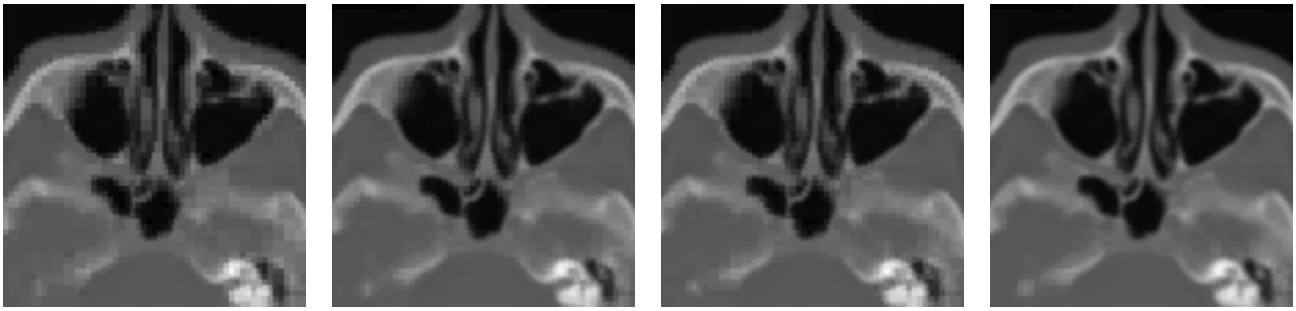
(b) Visible Man (**: density range $0 - 4095$)

Table 1. Quantitative analysis of reconstruction errors

During visualization, small changes of voxel densities in the critical regions can result in serious artifacts. For instance, trivial compression errors in the density of voxels that are very close to an isosurface may change severely its local topology as well as geometry. Furthermore, the gradient approximation methods such as central differences are numerically ill-conditioned. That is, the accuracy of approximated gradients can be easily deteriorated by tiny errors. Inadequate shading caused by distortion of the direction of normal vectors produces very unpleasant artifacts in ray-cast images. We observe this in Figure 6 where enlarged portions of images, generated by the two visualization methods, are shown. The figures (a) to (h) compare polygon-rendered images generated from extracted isosurfaces of isovalue 66. The effect of weighing the Haar filters are prominent as shown in the images. When weighted with only 7% of wavelets, the rendered isosurface is visually almost identical to the isosurface from the uncompressed data. The figures (i) to (p) show the difference in the quality of ray-cast images rendered with semitransparent skin and opaque bone. When voxels are not weighted, the blocky artifacts are annoyingly visible for all the four tested ratios. On the other hand, such artifacts are significantly reduced when our method is applied. When more than 5% of wavelets are used, the ray-cast images are almost free of aliasing artifacts.

5.2. Test with the Visible Man data set

We have also experimented with a larger volume data set of resolution $512 \times 512 \times 512$ by taking 512 slices from



(a) Unweighed: 3% (b) Weighted: 3% (c) Unweighed: 5% (d) Weighted: 5%

Figure 5. Comparison of quality of reconstructed images

the preprocessed fresh CT data of the Visible Man (Figure 8(h)). Two bytes are allocated per voxel, hence the data set takes up 256 MBytes. In visualizing this data, we are again interested in both bone and skin, where $D^{'}_{is} = \{880, 1800\}$ and $D^{'}_{rc} = \{[320, 992], [1120, 2400]\}$. The opacity transfer functions, defined by $[320, 800, 960, 992]$ and $[1120, 1300, 2300, 2400]$ for the two materials, and the gradient transfer function, determined by $[0, 1120]$ for rescaled gradients were used in the test. It is very interesting to see that the weight distribution of the Visible Man data (dotted lines in Figure 3) is very similar to the distribution of the Bighead data. The only major difference in ϕ^c_{rc} ((b)) is due to the use of the different α_ϵ and transfer functions. Figure 8(i) shows the 270th coronal slice colored according to the maximally combined weights. 23% of voxels have positive weights, and the reconstruction errors over these voxels are compared in Table 1(b).

From the through examination of visualized images, we find that the patterns of performance in visualization for both the Visible Man data and the Bighead data are very similar to each other. For ray-casting, the visual quality of images rendered with 4% of weighted voxels compares favorably with the quality of images visualized with 7% of unweighed voxels (Figure 7(a) and (b)). This is true in most cases of tested visualization, and indicates that about half the wavelets are necessary when weighed to achieve the same degree of visual quality. According to the analysis of compression rates in Table 2, this implies that the same quality of visualization is achieved using almost half of volume data.

We also tested with another case where $D^{''}_{is} = \{\, 600 \,\}$ and $D^{''}_{rc} = \emptyset$. We find that only 5.27% of voxels are important. When 3% of weighted wavelets are used, the compressed volume takes only 5,948 KBytes (compression rate = 44.07), and the rendered isosurface with isovalue 600 is visually identical to the isosurface from the uncompressed data of size 256 MBytes (Figure 7(c)).

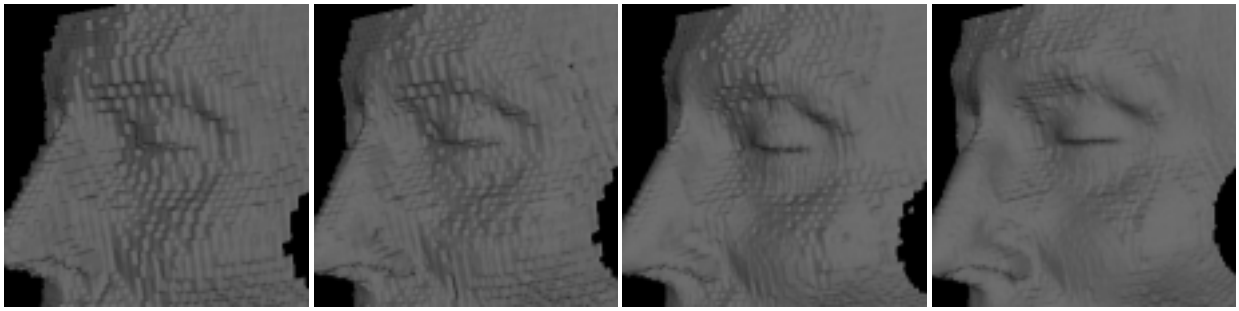
		Ratio of the used wavelets			
		2%	3%	5%	7%
Unweighed Haar	Size (KB)	6744	8780	12592	16340
	Rate	38.87	29.86	20.82	16.04
Weighted Haar	Size (KB)	6860	8256	10680	12872
	Rate	38.21	31.75	24.55	20.37

Table 2. Compression rates for Visible Man

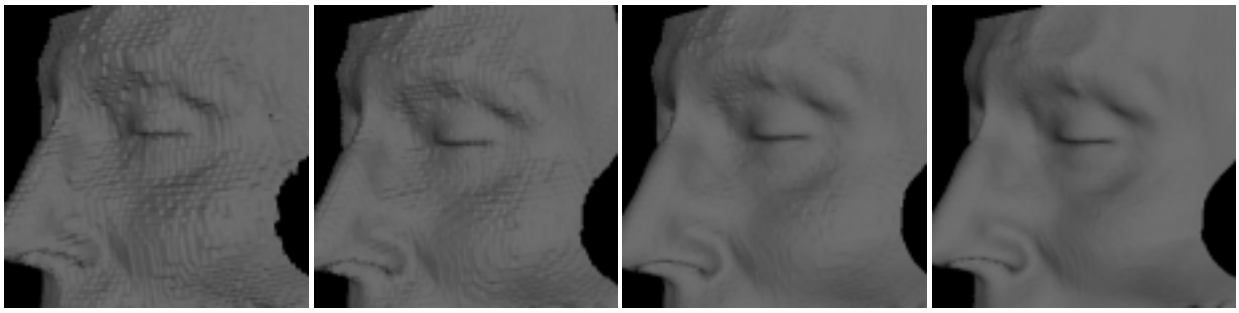
When we use only 2% of weighted wavelets in compression, the resulting size of compressed data is 5,552 KBytes (compression rate = 47.22). Aliasing artifacts start to appear in this case, but they are still trivial (Figure 7(d)). This is an extreme case where only one isosurface is considered. When a goal of visualization is to examine isosurfaces of, say, both bone and skin, a set like $D_{is} = \{556, 557, \cdots, 564, 1220, 1221, \cdots, 1228\}$ will offer enough flexibility in visualization. With such a set, we expect that the size of large volumes such as the Visible Human data sets still can be reduced significantly without harming the visual quality.

6. Concluding remarks

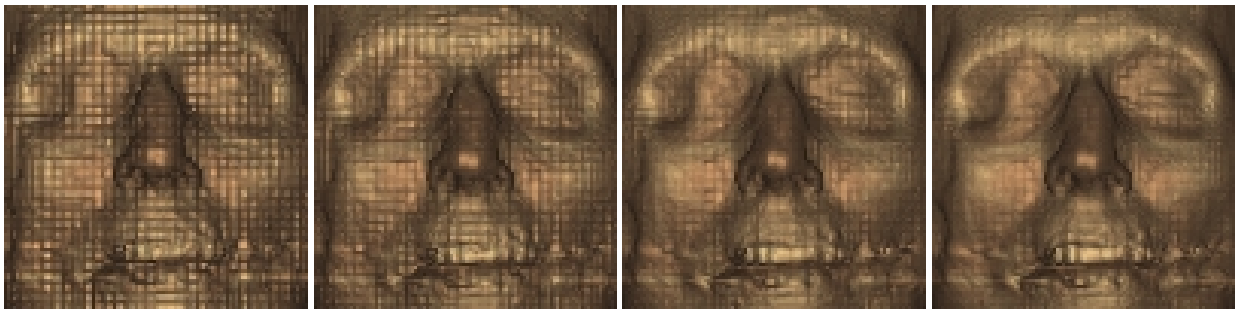
In this paper, we have presented a new method for classifying voxels according to their importance in visualization. To illustrate the effectiveness of our technique, we have also applied weight information to a lossy compression scheme. The new approach is different from the previous enhancement techniques in the sense that it judges the importance of voxels based upon their usage in actual visualizations to be performed as well as the spatial properties of data sets. Application of the weighting mechanism to a compression method was shown to be successful in overcoming the visual degradation of reconstructed images, which, as a consequence, leads to higher compression rates and visual fidelity.



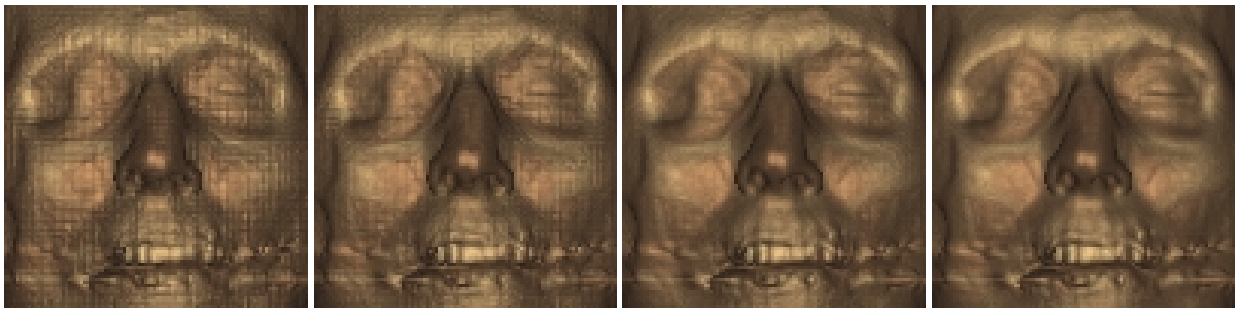
(a) Unweighed: 2% (b) Unweighed: 3% (c) Unweighed: 5% (d) Unweighed: 7%



(e) Weighted: 2% (f) Weighted: 3% (g) Weighted: 5% (h) Weighted: 7%



(i) Unweighed: 2% (j) Unweighed: 3% (k) Unweighed: 5% (l) Unweighed: 7%



(m) Weighted: 2% (n) Weighted: 3% (o) Weighted: 5% (p) Weighted: 7%

Figure 6. Comparison of visualized images ((a)-(h): isosurface rendering, (i)-(p): ray-casting)

(a) Unweighed: 7% (b) Weighted: 4% (c) Weighted: 3% (d) Weighted: 2%

Figure 7. Comparison of visualized images ((a),(b): ray-casting with $D^{'}_{is}$ and $D^{'}_{rc}$, (c), (d)): isosurface rendering with $D^{''}_{is}$)

As another application, we are currently implementing a new vector quantization algorithm where the weight information is utilized in designing enhanced codebooks. In conclusion, we expect that the new technique will allow larger volumes to be used in more compact form in interactive real-time volume visualization, and it will be very effective in achieving the dual purposes of combined isosurface extraction and ray-casting.

References

[1] C. Bajaj, I. Ihm, and S. Park. 3D RGB image compression for interactive applications. *ACM Transactions on Graphics*, 20(1):1–30, March 2001.

[2] A. Beers, M. Agrawala, and N. Chaddha. Rendering from compressed texture. *Computer Graphics (Proc. SIGGRAPH '96)*, pages 373–378, 1996.

[3] R. Gross, T. Ertl, and J. Aschoff. Efficient data structures for volume rendering of wavelet-compressed data. In *WSCG 96: The Fourth International Conference in Central Europe on Computer Graphics and Visualization*, Feburary 1996.

[4] R. Grosso, C. Lürig, and T. Ertl. The multilevel finite element method for adaptive mesh optimization and visualization of volume data. In *Proceedings of IEEE Visualization '97*, pages 387–394, 1997.

[5] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. *Computer Graphics (Proc. ACM SIGGRAPH '94)*, 28(4):451–458, 1994.

[6] D. Laidlaw, K. Fleischer, and A. Barr. Partial-volume bayesian classification of material mixtures in MR volume data using voxel histogram. *IEEE Transactions on Medical Imaging*, 17(1):74–86, Feb. 1998.

[7] M. Levoy. Display of surface from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[8] M. Levoy and P. Hanrahan. Light field rendering. *Computer Graphics (Proc. SIGGRAPH '96)*, pages 31–42, 1996.

[9] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. *Computer Graphics (Proc. ACM SIGGRAPH '87)*, 21(4):163–169, 1987.

[10] R. Machiraju, Z. Zhu, B. Fry, and R. Moorhead. Structure-significant representation of structured datasets. *IEEE Transactions on Visualization and Computer Graphics*, 4(2):117–132, April - June 1998.

[11] T. Möller, R. Machiraju, K. Mueller, and R. Yagel. Evaluation and design of filters using a taylor series expansion. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):184–192, April 1997.

[12] S. Muraki. Volume data and wavelet transforms. *IEEE Computer Graphics and Applications*, 13(4):50–56, 1993.

[13] S. Muraki. Multiscale volume representation by a DoG wavelet. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):109–116, June 1995.

[14] P. Ning and L. Hesselink. Fast volume rendering of compressed data. In *Proceedings of IEEE Visualization '93*, pages 11–18, San Jose, October 1993.

[15] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Transactions of Communications*, COM-34(11):1105–1115, November 1986.

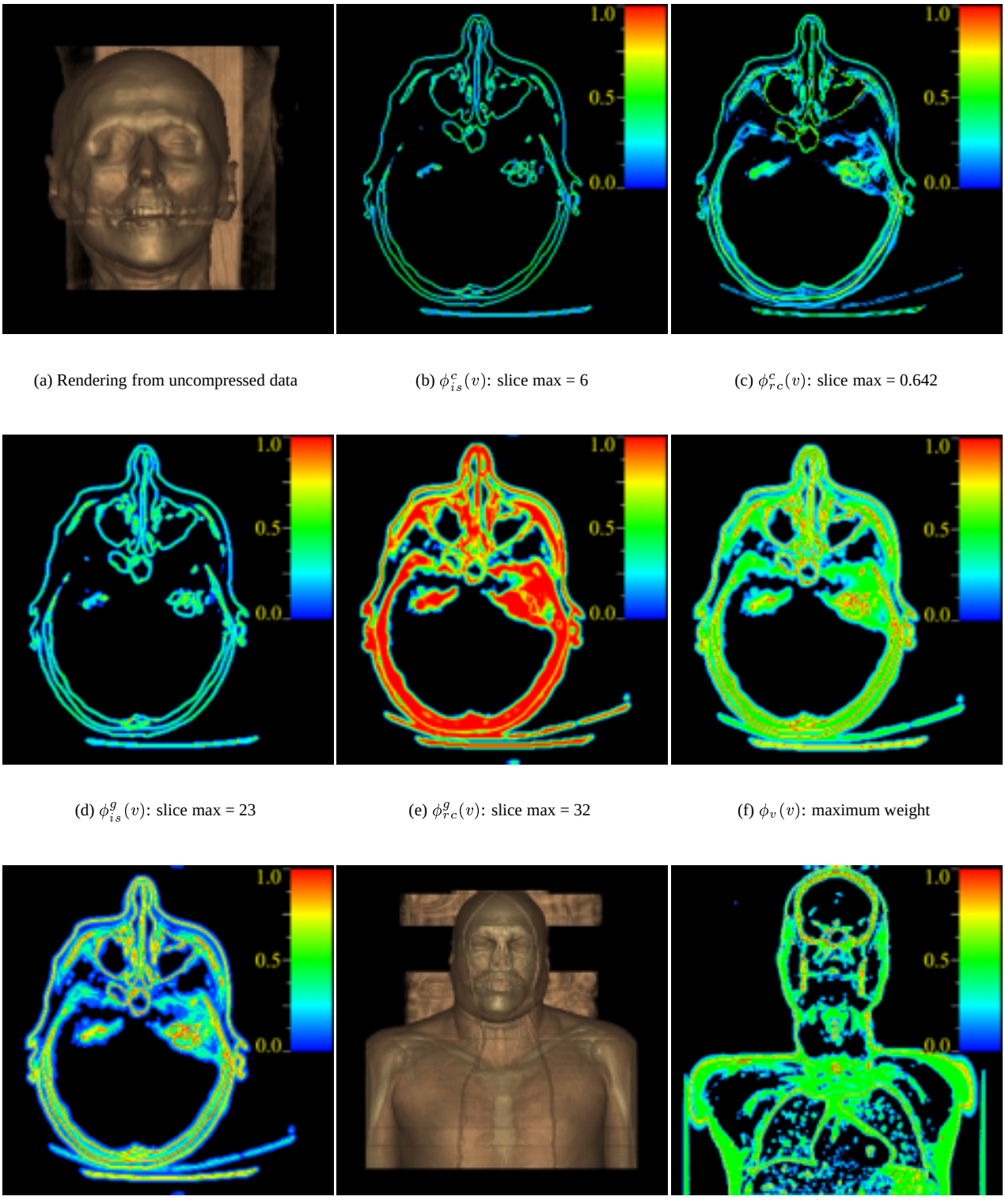
[16] A. Trott, R. Moorhead, and J. McGinley. Wavelets applied to lossless compression and progressive transmission of floating point data in 3-D curvilinear grids. In *Proceedings of IEEE Visualization '96*, pages 385–388, 1996.

[17] I. J. Trotts, B. Hamann, and K. I. Joy. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):224–237, Jul–Sep 1999.

[18] R. Westermann. A multiresolution framework for volume rendering. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 51–58, Washington, D.C., October 1994.

[19] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *Proceedings of IEEE Visualization '97*, pages 135–142, 1997.

[20] Z. Zhu, R. Machiraju, B. Fry, and R. Moorhead. Wavelet-based multiresolutional representation of computational field simulation datasets. In *Proceedings of IEEE Visualization '97*, pages 151–158, 1997.



(a) Rendering from uncompressed data

(b) $\phi^c_{is}(v)$: slice max = 6

(c) $\phi^c_{rc}(v)$: slice max = 0.642

(d) $\phi^g_{is}(v)$: slice max = 23

(e) $\phi^g_{rc}(v)$: slice max = 32

(f) $\phi_v(v)$: maximum weight

(g) $\phi_v(v)$: summed weight

(h) Rendering from uncompressed data

(i) $\phi_v(v)$: maximum weight

Figure 8. Ray-cast images and color-mapped weight images for sample slices ((a)-(g): Bighead, (h),(i): Visible Man)