# Multi-Dimensional Trees for Controlled Volume Rendering and Compression

Jane Wilhelms and Allen Van Gelder*
University of California, Santa Cruz

## Abstract

This paper explores the use of multi-dimensional trees to provide spatial and temporal efficiencies in imaging large data sets. Each node of the tree contains a model of the data in terms of a fixed number of basis functions, a measure of the error in that model, and a measure of the importance of the data in the region covered by the node. A divide-and-conquer algorithm permits efficient computation of these quantities at all nodes of the tree. The flexible design permits various sets of basis functions, error criteria, and importance criteria to be implemented easily.

Selective traversal of the tree provides images in acceptable time, by drawing nodes that cover a large volume as single objects when the approximation error and/or importance are low, and descending to finer detail otherwise. Trees over very large datasets can be pruned by the same criterion to provide data representations of acceptable size and accuracy. Compression and traversal are controlled by a user-defined combination of modeling error and data importance. For imaging decisions, additional parameters are considered, including grid location, allowed time, and projected screen area. To analyze results, two evaluation metrics are used: the first compares the hierarchical model to actual data values, and the second compares the pixel values of images produced by different parameter settings.

## 1 Introduction

As computers and algorithms improve so do our expectations of the kind and quality of images that can be produced. In scientific visualization, many data sets are larger than can be interactively visualized or even can be read into the available memory. Our research explores the use of multi-dimensional trees to deal with both the spatial and temporal aspects of this problem.

*Computer Science Dept, Room 225AS, Santa Cruz CA 95064, USA. E-mail `wilhelms@cs.ucsc.edu`, `avg@cs.ucsc.edu`

Our particular problem area is visualization of sampled $k$-dimensional scalar data arranged on a regular grid using direct volume rendering. However, we believe the data representation paradigm we use is applicable to more general multivariate and non-rectilinear data sets, and can provide useful insights into the imaging of any large database.

We build a space-efficient hierarchy over the data, each node of which contains three types of information: a model of the data below it; error and evaluation information for selective traversal; and structural information. Selective traversal for imaging or compression is controlled by user-defined tolerances.

We have found that selective traversal produces images that are subjectively and quantifiably very close to those produced using the entire data set, and significantly faster. Also, by storing the selected portion of the tree, the method can provide data compression. This article is necessarily abbreviated; a more detailed discussion is given in a technical report [19].

## 2 Background and Related Work

Work most closely related to ours is that concerned with hierarchical data structures for controlled imaging, algorithms for fast volume rendering, and methods for dealing with large data sets.

Meagher did some of the earliest work in representing 3D data using octrees [11]. Levoy used a binary octree to avoid transparent regions [7]. We used a max-min octree to avoid regions not intersecting the desired isosurface, and presented a space-efficient subdivision strategy, called *branch on need* (BON) [18]. This paper extends octrees and the BON strategy to $k$ dimensions.

Laur and Hanrahan build an octree over voxels, and compute the data mean and root mean square error (RME2) at each node [6]. This permits splat volume rendering by progressive refinement using the user-specified error tolerance. Our work extends that paper in several ways. Data models other than the mean are supported, and computed in constant time per node (Section 3.3). We support voxel and cell conventions (Section 4.2), and error metrics based on $L_q$ norms (Section 4.3); RME2 corresponds to the $L_2$ norm. Errors can be weighted by an "importance" function (Section 5). We also quantify image differences (Section 8.2).

Funkhouser and Sequin used a hierarchy for gaining consistent frame rate for complex viewing environments [4]. They also used a weighted combination of parameters to control imaging.

Speed gains for direct volume rendering have also been achieved by using voxel splatting [16], hardware-assisted projection [14, 6, 17], and preprocessing [1, 3, 21, 15]. Preprocessing may create large auxiliary data structures, which we particularly try to avoid in the research presented here.

Our method of hierarchical data representation has some similarities to wavelets and multi-resolution analysis [9, 2, 12, 5], but it has several significant differences. For example, Muraki used multi-resolution analysis to represent 3D volumes [12]. In contrast to our method, basis functions overlapped, and the calculation of one function value involved as many as 2000 basis functions.

Malzbender described efficient volume rendering using Fourier transforms [10]. Levoy described a variation that included a lighting model [8]. Neither method can model opacity.

Ning and Hesselink [13] used vector quantization to compress data sets for direct rendering. While this approach gives very good compression, it is not as flexible as a hierarchical model for imaging.

# 3 Hierarchical Data Models

This section describes the techniques to compute data models and approximation errors at all nodes of a multi-dimensional tree. Efficiency is achieved by computing the model and error at each node in terms of those values for the node's children.

## 3.1 Notation

The $k$-dimensional space of reals is denoted $\mathbf{R}^k$. Bold face letters represent $k$-D vectors. *Location* in $k$-D space is denoted by $\mathbf{x} = (x_1, \ldots, x_k)$. In 3-D and 4-D we will often use $(x, y, z)$ and $(x, y, z, t)$. A *volume* in $\mathbf{R}^k$ is a rectangular $k$-D parallelepiped, or closed *interval* denoted $[\mathbf{x}_{min}, \mathbf{x}_{max}]$. The *width* in dimension $j$ is denoted by $w_j = x_{max,j} - x_{min,j}$ (width vector $\mathbf{w}$).

The *volumetric data* is given as discrete samples on a regular $k$-D grid of (integer) resolutions $\mathbf{r} = (r_1, \ldots, r_k)$. Sample data points are indexed by a $k$-D index $\mathbf{p}$, where $p_j$ runs from 1 to $r_j$. The *spacings* of the data are $\Delta\mathbf{x}$. The relationship of the grid to $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ depends on whether we are using the *voxel* convention, or the *cell* convention (see Figure 1).

In the *voxel* convention, $w_j = r_j \Delta x_j$, $x_{min,j} = -\frac{1}{2} r_j \Delta x_j$, and $x_{max,j} = \frac{1}{2} r_j \Delta x_j$, centered around a sample point. In the *cell* convention, $w_j = (r_j - 1)\Delta x_j$, $x_{min,j} = -\frac{1}{2}(r_j - 1)\Delta x_j$, and $x_{max,j} = \frac{1}{2}(r_j - 1)\Delta x_j$, with sample points at each corner. Sample data values are denoted by $g(\mathbf{p})$, and the data viewed as a function throughout the volume is $g(\mathbf{x})$.

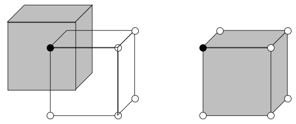


Figure 1: A *voxel* is the region surrounding a data point, whereas a *cell* is the region between data points.

## 3.2 Inner Products and Orthogonality

The derivation and error analysis of the hierarchical representation rest upon the concept of inner product. Typical inner products of interest are integrals over the volume and sums over the grid points.

For two functions $f$ and $g$, defined on $\mathbf{R}^k$, and belonging to a suitable function space, let $\langle f, g \rangle$ denote their inner product. An inner product on a volume induces inner products on subvolumes by restriction.[1] Also, an inner product induces a *norm*, which is often used as a distance measure: $\|g - f\| = \sqrt{\langle g-f,\, g-f \rangle}$. Our implementation uses an integral-based inner product.

Assume a set $\mathcal{B}$ of *basis functions*, $\{b_i\}$, such that distinct functions in $\mathcal{B}$ are orthogonal w.r.t $\langle\rangle$ ($\langle b_i, b_j \rangle = 0$). For the purpose of approximating $g$, let us require $f$ to be a weighted sum of $\mathcal{B}$, $f = \sum_i a_i b_i$, where the $a_i$'s are real numbers. $f$ is an *optimal* approximation ($\|g - f\|$ is minimized), if and only if

$$\langle g-f,\, b_i \rangle = 0 \qquad \text{for all } b_i \in \mathcal{B}$$

An important property resulting from this is that the coefficients of $f$ are given by $a_i$ (scaled) or $A_i$ (unscaled):

$$a_i = \frac{\langle g, b_i \rangle}{\langle b_i, b_i \rangle} \qquad\qquad A_i = \langle g, b_i \rangle$$

## 3.3 Divide-and-Conquer Approximation

We are interested in deriving an optimal approximation $f$ to a given function $g$ and *error bounds* (w.r.t. an inner product $\langle\rangle$), over the rectilinear $k$-D volume $V$. We seek an expression for $f$ in terms of the optimal approximation and error bounds for a *left* ($V_L$) and *right* ($V_R$) subvolume partitioning $V$ in dimension $J$.

By using the divide-and-conquer approach, we will be able to compute the optimal coefficients and errors of approximation for all nodes in the tree in constant time per node, and with only one pass through the data. [2]

[1] I.e., set the function to 0 outside the subvolume.

[2] The computation is also more accurate numerically than summing in a "for loop" through the data.

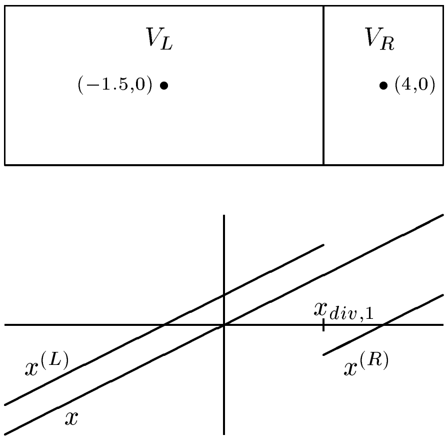


Figure 2: Basis function $x$ for $V$, $V_L$ and $V_R$ in Example 3.1. Boundary is at $x_{div,1}$.

---

Let $V$ be the closed $k$-D interval $[-\frac{1}{2}\mathbf{w}, \frac{1}{2}\mathbf{w}]$ (i.e., the volume is centered). Let the width $w_j = w_L + w_R$, where $w_L > 0$ and $w_R > 0$. Define

$$x_{div,j} = w_L - \tfrac{1}{2}w_j = \tfrac{1}{2}w_j - w_R = \tfrac{1}{2}(w_L - w_R)$$

Let $V_L$, $V_R$ be the $k$-D intervals $[(\mathbf{x}_{min})_L, (\mathbf{x}_{max})_L]$ and $[(\mathbf{x}_{min})_R, (\mathbf{x}_{max})_R]$, respectively. In dimension $j$, the only dimension partitioned, the center of $V_L$ is at $-w_R/2$, and the center of $V_R$ is at $+w_L/2$.

The desired approximation $f$ over $V$ will use the basis set $\mathcal{B}$. The approximations over $V_L$ and $V_R$ are in basis sets $\mathcal{B}_L$ and $\mathcal{B}_R$, which we assume are closely related to $\mathcal{B}$, but apply to their respective domains. Let $b^{(L)}$ denote basis function $b \in \mathcal{B}_L$, expressed in the coordinate system of $V$. Also, denote the restriction of a function $b$ to a subvolume $V_L$ by $b|_L$; that is, $b|_L$ is equal to $b$ in $V_L$ and is zero outside $V_L$. Use similar notations for $R$.

**Example 3.1:** Consider a 2-D "volume" $V$ with $\mathbf{w} = (11, 4)$ and $j = 1$ (see Figure 2). Let the set of basis functions be $\mathcal{B} = \{1, x, y, xy\}$. Suppose the desired partition is $w_L = 8$ and $w_R = 3$. The centroid of $V_L$ in the coordinate system of $V$ is at $(-1.5, 0)$. $\mathcal{B}_L$ is similar. Then we have:

$$\begin{array}{rcl rcl}
1^{(L)} &=& 1|_L & 1^{(R)} &=& 1|_R \\
x^{(L)} &=& x|_L + 1.5|_L & x^{(R)} &=& x|_R - 4|_R \\
y^{(L)} &=& y|_L & y^{(R)} &=& y|_R \\
xy^{(L)} &=& xy|_L + 1.5y|_L & xy^{(R)} &=& xy|_R - 4y|_R
\end{array}$$

Observe that equations for $\{1|_L, x|_L, y|_L, xy|_L\}$ can be solved quickly in terms of $\{1^{(L)}, x^{(L)}, y^{(L)}, xy^{(L)}\}$. The idea extends to any number of dimensions. □

To find $A_i$ for the optimal approximation, decompose $g$ into $g|_L + g|_R$, the restrictions to $V_L$ and $V_R$ (i.e., zero outside their respective domains).

$$A_i = \langle g|_L, b_i \rangle + \langle g|_R, b_i \rangle = \langle g|_L, b_i|_L \rangle + \langle g|_R, b_i|_R \rangle$$

But $b_i|_L$ is a linear combination of certain $b_n^{(L)}$, and $b_i|_R$ is a linear combination of certain $b_n^{(R)}$. We will recursively derive optimal approximations $f_L$ to $g|_L$ and $f_R$ to $g|_R$. Therefore,

$$(A_L)_n = \langle g|_L, b_n^{(L)} \rangle \quad \text{and} \quad (A_R)_n = \langle g|_R, b_n^{(R)} \rangle$$

are known when the two subproblems are completed. These values can be used to compute the $A_i$. The general description above will now be illustrated for some common basis sets. See [19] for a discussion of quadratic basis functions.

**Example 3.2:** For the voxel mean model, the basis set $\mathcal{B}$ consists of just the constant function, 1, with one unscaled coefficient, $A = A_L + A_R$. (This case is essentially the one considered by Laur and Hanrahan, using the voxel model [6].) The scaled coefficient $(a_i)$ is the (possibly weighted) mean value of $g$, and is given by

$$a = \frac{A_L + A_R}{\langle 1, 1 \rangle} = \frac{\langle 1|_L, 1|_L \rangle a_L + \langle 1|_R, 1|_R \rangle a_R}{\langle 1, 1 \rangle} \qquad \square$$

**Example 3.3:** Suppose $k = 3$ or $4$ and $\mathcal{B}$ is the trilinear or quadlinear basis. We shall index $b_i \in \mathcal{B}$ as follows:

$$\begin{array}{rcll}
b_0(\mathbf{x}) &=& 1 & \text{for } \mathbf{x} \in V \\
b_1(\mathbf{x}) &=& x_1 b_0(\mathbf{x}) & \\
b_{i+2}(\mathbf{x}) &=& x_2 b_i(\mathbf{x}) & \text{for } i = 0, 1 \\
b_{i+4}(\mathbf{x}) &=& x_3 b_i(\mathbf{x}) & \text{for } i = 0, \ldots, 3 \\
b_{i+8}(\mathbf{x}) &=& x_4 b_i(\mathbf{x}) & \text{for } i = 0, \ldots, 7
\end{array}$$

Thus the bits of the index indicate whether the corresponding linear factor is present, and $b_i$ is independent of $x_j$ just when $\lfloor i/2^{j-1} \rfloor$ is even. If $b_i$ is not independent of $x_j$, it is linear in $x_j$ in this k-linear case. We identify $(x_1, x_2, x_3, x_4)$ with $(x, y, z, t)$. For example, the trilinear basis functions would be $(1, x, y, xy, z, xz, yz, xyz)$.

Similarly to Example 3.2, for decomposition in direction $j$, we have

$$A_i = (A_L)_i + (A_R)_i \qquad \text{for } \lfloor i/2^{j-1} \rfloor$$

For $\lfloor i/2^{j-1} \rfloor$ odd, we note (see Figure 2) that

$$b_i|_L = b_i^{(L)} - \tfrac{1}{2} w_R b_{i-2^{j-1}}^{(L)} \qquad b_i|_R = b_i^{(R)} + \tfrac{1}{2} w_L b_{i-2^{j-1}}^{(R)}$$

From this it follows that, for $\lfloor i/2^{j-1} \rfloor$ odd,

$$A_i = (A_L)_i + (A_R)_i - \tfrac{1}{2} w_R (A_L)_{i-2^{j-1}} + \tfrac{1}{2} w_L (A_R)_{i-2^{j-1}}$$

□

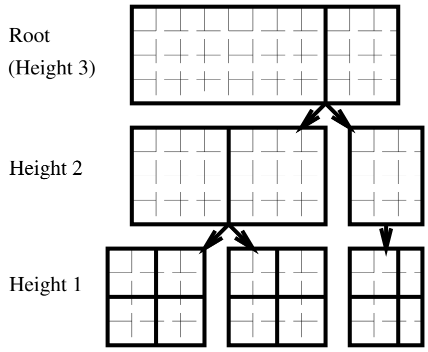


Figure 3: The BON strategy on an 11x4 2D "volume". Solid lines show how nodes subdivide. Dashed lines are voxel or cell boundaries.

---

**Example 3.4:** Continuing with Example 3.1, assume the indexing: $b_0 = 1$, $b_1 = x$, $b_2 = y$, $b_3 = xy$. Here $j = 1$, so $b_i$ is linear in $x$ when $i$ is odd. Working up the tree from below, we have already calculated coefficients for the optimal approximations to the two subvolumes $V_L$ $((A_L)_0, (A_L)_1, (A_L)_2, (A_L)_3)$ and $V_R$, $((A_R)_0, (A_R)_1, (A_R)_2, (A_R)_3)$. From the specified $\mathbf{w}$, we know that $\langle 1|_L, 1|_L\rangle = 32$ and $\langle 1|_R, 1|_R\rangle = 12$ and $\langle 1, 1\rangle = 33$. Then we find that

$$\begin{aligned} A_0 &= (A_L)_0 + (A_R)_0 \\ A_2 &= (A_L)_2 + (A_R)_2 \\ A_1 &= (A_L)_1 + (A_R)_1 - 1.5(A_L)_0 + 4(A_R)_0 \\ A_3 &= (A_L)_3 + (A_R)_3 - 1.5(A_L)_2 + 4(A_R)_2 \quad \square \end{aligned}$$

Now the error in the approximation, $e^2 \stackrel{\text{def}}{=} \|g - f\|^2$, over the volume $V$, is computable as follows:

$$\begin{aligned} \langle g, g\rangle &= \langle g|_L, g|_L\rangle + \langle g|_R, g|_R\rangle \\ \langle f, f\rangle &= \textstyle\sum_i A_i^2 / \langle b_i, b_i\rangle \\ e^2 &= \langle g - f, g - f\rangle = \langle g, g\rangle - \langle f, f\rangle \end{aligned}$$

To summarize, in each subproblem, $V_L$ and $V_R$, we compute $\{A_i\}$ and $\langle g, g\rangle$, then use those results to compute these quantities for $V$. To decompose in several $j$ directions, we decompose in one after the other, recursively. At the bottom level, these quantities are computed directly from the data.

# 4 Implementation of the Hierarchy

Our flexible $k$-dimensional tree encodes structural information about the tree, model information describing the data within the region, and evaluation information to control compression and image quality.

## 4.1 Structural Information

The hierarchy design is an extension to higher dimensions of the BON octree strategy [18]. As shown there, this strategy can achieve significant savings when the grid resolutions are unequal, or are not powers of 2; higher dimensions tend to create greater savings. Figure 3 illustrates the main idea in 2D.

Two types of structural information are stored within the tree. The first is a pointer to the first child of this node, which must be 4 bytes. Sibling tree nodes are contiguous, so one pointer suffices. Nodes on height 1 point to the first child in the data. The second type is the branching pattern, stored as a bit vector (in our case, one byte, allowing up to 8 dimensions).

## 4.2 Model Information

The model information within each node represents the data beneath that node, either exactly or approximately. In general, the approximation is closest near the bottom of the tree, and gets worse higher up. We have experimented with three data models described below.

In choosing a model, one can give priority either to continuity between regions or to a best fit of the data over the defined region. We chose the latter, as being a more appropriate representation for scientific data. However, for some purposes, one might prefer a model that minimizes discontinuity along boundaries. Even if continuity between regions on a particular hierarchy level is maintained, when imaging is done using different tree levels, discontinuities will result.

The main discussion assumes the data type is float (4 bytes). In general, the model information can use the same type as the data, when this type is shorter, such as *short* (2 bytes) or *byte* (1 byte).

### 4.2.1 The Voxel Mean Model

In this simple voxel model, each node stores one value representing the average of the data values of all points beneath it. (We store this as a four byte floating point value, though for greater compression, it could be the type of the original data if that is smaller than four bytes. While succinct, the model may show discontinuities in imaging where these regions meet, even when drawn at the deepest voxel level. The spatial cost of storing the mean is one floating point value. For 3D data, there are about $n/7$ nodes for $n$ data points; for 4D, the figure is about $n/15$. The model is easily compressed by truncating the tree and discarding data.

We initially implemented the mean model using Haar wavelets [2]. However, the detail functions (7 coefficients in 3D and 15 in 4D) need to be combined to recover the means during tree traversal. For little extra cost, we simply store the mean and retain the data.

### 4.2.2 The Voxel Trilinear Model

The voxel trilinear model stores a trilinear function at each node that best fits the data values represented by the node. The trilinear nodes on height one of the tree fits the data points exactly, so the data can be discarded and regenerated as needed from the trilinear coefficients.

Our implementation uses eight floating point coefficients (32 bytes) per node, no matter what the type of the data. Assuming the coefficients require as many bytes as a data item, this model uses the same space as the mean model for data and model combined ($8n/7$ in 3D, $16n/15$ in 4D). It sometimes provides a better approximation of the data at higher levels of the tree.

#### 4.2.3 The Cell Trilinear Model

In this standard cell model, data points lie at the corners of cells and are shared between neighboring cells. At height one, a node covers (up to) a $2^k$ array of cells. Assuming a $k$-linear function over the cell regions, the data field is continuous, so rendering at the deepest level can provide a more continuous image than the previous methods. Without compression, the spatial cost of this model, assuming floating point 3D data, is the size of the original data $n$ plus approximately $n/7$ nodes each of which contains 8 trilinear coefficients, or $15n/7$.

Data compression is more complex for a cell model, because of data shared between nodal regions. One solution is to separate the data into smaller grids with redundant data points along the boundaries. Clusters of 125 yield $125n/64$ data samples and $n/7$ internal tree nodes. To achieve compression, a substantial fraction of clusters need to be discardable.

### 4.3 Evaluation Information

The two types of evaluation information stored in the hierarchy are nodal error and data importance.

The *nodal error* is an average deviation of the model ($f(\mathbf{p})$) from the data ($g(\mathbf{p})$) within the region $V$ (with $|V|$ data points) covered by the node. In the $L_q$ norm, the equation is:

$$e = \left( \frac{1}{|V|} \sum_V |g(\mathbf{p}) - f(\mathbf{p})|^q \right)^{\frac{1}{q}}$$

For $q = 2$ (see Section 3.3), $e$ can be computed from $\langle g, g \rangle$ and the coefficients of $f$; also, either a sum or an integral can be used. For $q \neq 2$, each node's $e$ must be computed from scratch. We use a floating point value (4 bytes) for this. However, as with model information, the type of the data may be used if it is shorter.

The second evaluation metric is *data importance*. In many data sets, different data values and/or different regions have different interest levels, e.g. air surrounding a CT scan. Using an interactive transfer function editor, the user can design an importance function giving each data value an importance between 0 and 1 (the default is 1). At each tree node the maximum importance of any data point in its region is stored (in one byte).

## 5 Selective Traversal

A number of evaluation parameters are used to control the traversal, most of which are calculated on the fly. Left in their default state, the parameters have no effect and traversal continues until a node with no error is found. User-defined parameters are:

1. *Model Error Threshold*: An allowed error which is multiplied by the data set's standard deviation. Nodes with *nodal error* (possibly modified below) at or below this threshold are rendered as single objects.
2. *Data Importance*: If data importance is activated, the node's *nodal error* is multiplied by its *importance* before being compared to the threshold.
3. *Pixel Coverage Weighting*: A pixel coverage value gives nodes projecting to small regions reduced importance.
4. *Region Restrict*: Restricts traversal to a rectangular 3D region.
5. *Dimension Restrict*: For data with greater than three dimensions, defines which three should be used for imaging, as well as the constant values for other dimensions.
6. *Tree Depth*: Depth in the tree which traversal never exceeds.
7. *Allowed Time*: An option somewhat orthogonal to the above, the system may calculate the deepest level that, using a rendering cost estimate, can be rendered in the allowed time.
8. *Clipping*: If the nodal region is not visible, the traversal returns immediately.

The evaluation metrics defined above are checked at each node of the tree during selective traversal to determine whether traversal should descend further or return. If traversing for imaging (not compression), the region is drawn when traversal stops. If traversal is for compression, this node becomes a leaf.

## 6 Rendering Methods

Our hierarchical approach is not restricted to any particular rendering method; we use direct volume rendering using coherent projection [17]. This method calculates information concerning the projection of a rectilinear cell and uses hardware Gouraud-shading for rapid rendering. It produces quite good images rapidly. We recently added a new rendering method using 3D texture maps; this is described elsewhere [22].

Consider renderings with no compression on a 3D volume of resolutions $(r_x, r_y, r_z)$. with $n = r_x r_y r_z$. For the voxel mean model, one constant value region is drawn for each data point. The voxel and cell trilinear models treat the projected region as a trilinear function, which is evaluated at region corners. "Voxel trilinear" draws about $n/8$ cells, most covering 8 voxels. "Cell trilinear" draws $(r_x-1)*(r_y-1)*(r_z-1)$ cells. Constant value coherent projection is approximately twice as fast as when corner values vary.

It should be pointed out that the evaluation parameters, except for coverage, do not take into account imaging issues, such as transfer function mappings, or discontinuity between neighbor regions. The hierarchical model could accommodate a more image-based metric (each a model giving better continuity between subregions) , should this be considered more important.

## 7 Error Analysis

Any visualization method must consider both the validity of the representation compared to the data and the quality of the image (more difficult to measure). Our basic metric for data validity is the nodal error, possibly weighted during traversal by parameters described previously. Our metric for image quality is the closeness of the resultant image with some weighted error to a *standard image* produced by the same visualization method allowing no error.

In judging image quality we attempt to quantify what is partly a subjective evaluation. To do so, we have examined five *error levels* determined by the variations from the standard image. The five error levels are quantified by their root mean squared image errors (RME2), and their maximum absolute image error (MAE) compared to the standard image. By image error, we mean pixel-by-pixel color difference between the standard image and the image with error, scaled to the range 0-255, counting only non-black pixels.

The five levels are shown in Table 3. Subjectively, levels 1 and 2 are nearly identical to a no error image, and levels 3 and 4 somewhat clearer differences. (We were only interested in evaluations that provided images with good information content.)

## 8 Experimental Results

In our experiments, we explore the space and time costs of using a hierarchy, how much this can be reduced by compression, and what evaluation parameters provide a good balance between imaging time and quality. We especially wished to explore ways to quantify our results. The user interface is described elsewhere [20]. We used a selection of rectilinear 3D and 4D data sets.[3] Table 1 shows characteristics of the data sets. Statistics were run on a Silicon Graphics Reality Engine II, with a 100 Mhz processor and 64 megabytes of memory.

| Data [Type] | Resolution | Samples | StdDev |
|---|---|---|---|
| Hipiph [f] | 64x64x64 | 262,144 | .01845 |
| Sod [b] | 97x97x116 | 1,091,444 | 15.62 |
| P6985 [f] | 244x91x64 | 1,421,056 | .004208 |
| Dolphin [s] | 320x320x40 | 4,096,000 | 612.3 |
| CTHalf [s] | 251x512x113 | 14,521,856 | 612.6 |
| Mandelbrot [f] | 256x256x256 | 16,777,216 | 443.98 |
| CTHead [s] | 512x512x113 | 29,622,272 | 564.1 |
| Radm [f] | 29x69x63x9 | 1,134,567 | .001014 |
| Heart [s] | 256x256x8x16 | 8,388,608 | 57.38 |

Table 1: Data Set Characteristics. Note Radm and Heart are 4D. ([f] = float, [s] = short, [b] = byte.)

### 8.1 Space Usage

To explore lossless and lossy representations, we first assume, for ease of comparison, that all our data was floating point (4 bytes). Later we summarize differences for shorter data types. All nodes have one pointer (4 bytes), one error (4 bytes), one importance value (1 byte), one branch pattern (1 byte). In our implementation, mean nodes add one float (4 bytes) and trilinear nodes eight floats (32 bytes). As our machine forces array structures to be multiples of 4 bytes, mean nodes take 16 bytes and trilinear nodes 44 bytes. Thus, the voxel models without compression take about 1.6 times the data size and the cell model takes about 2.6 times the data size on 3D data sets. For 4D, uncompressed voxel models take about 1.3 times the data size and cell models take about 2.3 times. The hierarchy on 4D data sets is about 20% more space efficient than using multiple 3D trees.

Table 2 shows spatial usage, comparing the size taken by the tree and data after compression to the size of the original data alone. For cell trilinear, the compressed version either retains the entire data, or retains clusters of 27 data points (in 3D), whichever is smaller. In general, lossless compression doesn't allow much space savings, though on the CT data the large homogeneous regions did allow significant reduction. This was even more true when we used obvious restriction and reduced importance, as in the dolphin data set, when we centered only on the dolphin. Allowing more error, of course, gave better compression. The one exception was the Sod data set, which is highly varying throughout.

Compared to *float* data, relative hierarchy space can be up to 45% higher for 3D *single byte* data, up to 20% higher for 3D *short integer* data, and up to 24% higher for 4D *byte* or *short* data. Tree space usage can be reduced by using pointerless trees, if compressed trees are undesirable.

[3] Hipip (High Potential Iron Protein) is from L. Noodleman and D. Case, Scripps Clinic, La Jolla, Ca. Sod is an electron density map of superoxide dismutase from D. McRee, Scripps Clinic, La Jolla, Ca. P6985 is CFD data from U. Rist at the University of Stuttgart. Dolphin is a CT-scan of a dolphin head from T. Cranford, UCSC. The CTHead and CTHalf (the same data set) was from UNC. RADM (regional acid deposition model) is from C. Landreth of NCSC and R. Dennis of US EPA. The beating heart was CT data from J. M. Pfaff of Tower Imaging and C. A. Morioka of Cedars-Sinai Medical Center.

| Data Set | Data Model | No Error | 1% Error | 5% Error |
|---|---|---|---|---|
| Hipip | Voxel Mean | 1.57 | 1.03 | 0.55 |
| | Voxel Trilinear | 1.57 | 0.59 | 0.27 |
| | Cell Trilinear | 2.57 | 1.21 | 0.42 |
| Sod | Voxel Mean | 1.12 | 1.11 | 1.01 |
| | Voxel Trilinear | 1.46 | 1.44 | 1.31 |
| | Cell Trilinear | 2.49 | 2.45 | 2.23 |
| P6985 | Voxel Mean | 1.58 | 1.44 | 0.48 |
| | Voxel Trilinear | 1.58 | 0.46 | 0.14 |
| | Cell Trilinear | 2.56 | 0.76 | 0.19 |
| Dolphin | Voxel Mean | 1.26 | 0.36 | 0.17 |
| | Voxel Trilinear | 1.27 | 0.42 | 0.24 |
| | Cell Trilinear | 2.28 | 0.76 | 0.50 |
| CTHalf | Voxel Mean | 0.87 | 0.79 | 0.48 |
| | Voxel Trilinear | 0.95 | 0.92 | 0.58 |
| | Cell Trilinear | 2.55 | 1.78 | 0.93 |
| CTHead | Voxel Mean | 0.73 | 0.68 | 0.43 |
| Radm (4D) | Voxel Mean | 1.14 | 0.71 | 0.44 |
| Heart (4D) | Voxel Mean | 1.27 | 1.26 | 1.26 |
| + Importance/Restrict | | | | |
| Dolphin | Voxel Mean | 0.21 | 0.17 | 0.08 |
| | Voxel Trilinear | 0.12 | 0.11 | 0.06 |
| | Cell Trilinear | 0.65 | 0.37 | 0.19 |

Table 2: Space Usage by Test Data Sets. Numbers represent the ratio of the necessary space for the data representation divided by the size of the original data set (assumes data values take 4 bytes, mean nodes 16 bytes, trilinear nodes 44 bytes). "+ Importance/Restrict" means restriction and/or importance was used.

### 8.2 Image Evaluation and Selective Traversal

We compared images generated with varying amounts of error to those made with no error for their particular data model (see Table 3). This table also gives an indication of the cost of building the tree, which need only be done once and saved. The times are given in c.p.u. seconds using one processor. Significant speedups are often seen with very small degradation in image quality. Figure 4 (in color images) shows images differences for cell trilinear models on the hipip data set. With level 2 image error, the image was created 10 times faster than using no error, though differences were very small. With level 4 image error, the image was created 65 faster than using no error, but discontinuities are apparent. Pixel values in the difference image are multiplied by 5 to show them more clearly.

We found the described levels of image difference helpful, but not entirely satisfactory. Other characteristics, which we are not using to control rendering, clearly play a major role in image quality. Also, while there is always a monotonic relationship between image errors and data model errors, it is not very consistent between volumes.

For our final examination, we explored raising the nodal error to higher exponents (Section 4.3), giving greater relative weight to large errors. With larger exponents, images often gave a better representation of small, highly variant regions, in equivalent time. Figure 5 shows a region of the hipip data set using a voxel mean model with difference nodal error exponents for no error and level 4 image errors. Notice that the small red and blue spheres that appear in the no error image disappear in the image when the nodal error is exponent 2 (square error), start to reappear at exponent 6, and largely reappear at exponent 12. The level 4 image error pictures all take approximately the same time to draw; the allowed error is distributed differently.

## 9 Conclusions

We found that the hierarchical strategy was extremely successful in providing a flexible imaging approach. It provides a number of easy-to-use parameters that control the image quality and speed in an intuitive manner. Because the parameters are related to the error compared to the original volume, they allow users to control the accuracy of the image. While hierarchies do not compress as well as other methods, the compressed version can be used nearly as quickly as the original data. Further, in many cases, users may feel more confident than we did in using restriction and importance to reduce the necessary data size.

In general, we found the voxel mean model more successful than we expected, usually giving quite good images. The voxel trilinear model could sometimes produce equivalent images in much less time, but was inconsistent and sometimes produced irritating discontinuities. The cell trilinear produced the most consistently satisfying images.

Some problems of discontinuities may be due to the limited expressibility of the trilinear data model. We have begun to experiment with quadratic data models for better continuity between regions [19]. Also, rendering methods that attempt to use a consistent model at shared faces could reduce rendering artifacts.

We believe the hierarchical approach could be extremely helpful for irregularly sampled data sets. providing advantage of a regular geometry as well as a weighting for the concentration of samples in different regions. We are presently exploring this research issue in the context of curvilinear data sets.

## Acknowledgements

Funds for the support of this study have been allocated by a cooperative agreement with NASA-Ames Research Center, Moffett Field, California, under Interchange No. NCA2-430, and by the National Science Foundation, Grant Number ASC-9102497, and Grant Number CDA-9115268.

## References

[1] Judy Challinger. Scalable parallel volume raycasting for nonrectilinear computational grids. In *IEEE Parallel Visualization Workshop*, October 1993.

[2] C. K. Chui. *An Introduction to Wavelets.* Academic Press, Inc., 1992.

| Data Set | Create Tree (sec) | No Tree Regions/ second | No Tree (sec ) | Go Deep RME2=0 MAE=0 | Level 0 RME2=0 MAE=0 | Level 1 RME2≤3 MAE≤10 | Level 2 RME2≤6 MAE≤20 | Level 3 RME2≤9 MAE≤40 | Level 4 RME2≤12 MAE≤80 |
|---|---|---|---|---|---|---|---|---|---|
| **Voxel Mean** | | | | | | | | | |
| Hipip | 0.6 | 35K | 7.45 | 8.62 | 8.81 | 4.35 | 1.11 | 0.61 | 0.17 |
| P6985 | 3.05 | 33K | 43.52 | 47.74 | 47.63 | 29.46 | 14.86 | 10.87 | 4.87 |
| CTHalf | 56.8 | 53K | 274.8 | 308.5 | 187.0 | 121.5 | 108.1 | 91.8 | 53.8 |
| CTHead | 167.2 | 55K | 533.07 | 653.0 | 340.0 | 173.2 | 100.1 | 69.7 | 37.5 |
| **Voxel Trilinear** | | | | | | | | | |
| Hipip | 1.2 | 34K | 7.80 | 2.12 | 2.20 | 0.72 | 0.44 | 0.17 | 0.13 |
| P6985 | 6.79 | 31K | 45.46 | 12.32 | 13.46 | 7.00 | 5.72 | 3.61 | 2.12 |
| **Cell Trilinear** | | | | | | | | | |
| Hipip | 7.76 | 18K | 14.54 | 18.21 | 18.30 | 3.17 | 1.78 | 0.67 | 0.28 |
| P6985 | 42.49 | 18K | 80.33 | 99.46 | 101.10 | 47.59 | 32.59 | 11.72 | 5.61 |
| CTHalf | 491.1 | 19K | 728.01 | 911.00 | 578.46 | 400.36 | 350.82 | 215.92 | 82.23 |

Table 3: Time versus Error Level by Test Data Sets, discussed in Section 8.2. Times are c.p.u. seconds on Reality Engine II. "No Tree" means coherent projection. "Go Deep" means go to tree leaves.


[3] Robert A. Drebin, Elliot K. Fishman, and Donna Magid. Volumetric three-dimensional image rendering: Thresholding vs. non-thresholding techniques. *Radiology*, 165:131, 1987.

[4] Thomas Funkhouser and Carlo Sequin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Computer Graphics (ACM Siggraph Proceedings)*, 27(4):247–254, August 1993.

[5] Steven J. Gortler, Peter Schroeder, Michael F. Cohen, and Pat Hanrahan. Wavelet radiosity. *Computer Graphics (ACM Siggraph Proceedings)*, 27(4):221–230, August 1993.

[6] David Laur and Pat Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics (ACM Siggraph Proceedings)*, 25(4):285–288, July 1991.

[7] Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.

[8] Marc Levoy. Volume rendering using the fourier projection-slice theorem. In *Proceedings of Graphics Interface '92*, Vancouver, B.C., 1992. Also Stanford University Technical Report CSL-TR-92-521.

[9] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

[10] Tom Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, July 1993.

[11] Donald J. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19:129–147, 1982.

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

[13] Paul Ning and Lambertus Hesselink. Vector quantization for volume rendering. In *Visualization '93*, San Jose, Ca, October 1993. IEEE.

[14] Peter Shirley and Allan Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics*, 24(5):63–70, December 1990.

[15] Allen Van Gelder and Jane Wilhelms. Rapid exploration of curvilinear grids using direct volume rendering. In *Visualization 93 Conference*, San Jose, CA, October 1993. IEEE. (extended abstract) Also, University of California technical report UCSC-CRL-93-02.

[16] Lee Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–76, August 1990.

[17] Jane Wilhelms and Allen Van Gelder. A coherent projection approach for direct volume rendering. *Computer Graphics (Proceedings ACM Siggraph)*, 25(4):275–284, 1991.

[18] Jane Wilhelms and Allen Van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227, July 1992. Extended abstract in ACM Computer Graphics 24(5) 57–62; also UCSC technical report UCSC-CRL-90-28.

[19] Jane Wilhelms and Allen Van Gelder. Multidimensional trees for controlled volume rendering and compression. Technical Report UCSC-CRL-94-02, CIS Board, University of California, Santa Cruz, January 1994.

[20] Jane Wilhelms, Allen Van Gelder, Tom Goodman, Orion Wilson, Ron MacCracken, and Andy John. Design decisions for a direct volume renderer user interface – some whys and hows. In *SPIE Conference on Electronic Imaging*, pages 102–113, San Jose, California, February 1994.

[21] Peter Williams. Interactive splatting of nonrectilinear volumes. In *Visualization '92*, pages 37–44. IEEE, October 1992.

[22] Orion Wilson, Allen Van Gelder, and Jane Wilhelms. Direct volume rendering via 3d textures. Technical Report UCSC-CRL-94-19, CIS Board, University of California, Santa Cruz, 1994. (submitted for publication).

Error Level 0

Error Level 4

Error Level 2

Diff(*5) 0 vs. 4

Figure 4: The Hipip data set using the cell trilinear model showing image error levels 0, 2, and 4. The differences between image error levels 0 and 4 are shown scaled by 5.

No Error

Error Exponent 2

Error Exponent 6

Error Exponent 12

Figure 5: The Hipip data set using the voxel mean model showing a no error image and three images with level 4 image errors using nodal error exponents 2, 6, and 12. Features reappear with higher exponents.