Histogram Spectra for Multivariate Time-Varying Volume LOD Selection

Steven Martin*
The Ohio State University

Han-Wei Shen†
The Ohio State University

ABSTRACT

Level of detail techniques are widely applied to minimize sampling error subject to working set size constraints. Typical large data sets being produced today have many variables sampled across time-varying volumes. Visualization of these multivariate volumes is commonly phrased in terms of conditional expressions such as "show variable A where variable B is between B_1 and B_2." The bounds, B_1 and B_2, tend to be specified during the interactive portion of the workflow. Thus, to maximize quality over the salient interval, level of detail selection should also be interactive. We introduce the concept of histogram spectra to quickly and compactly quantify the statistical sensitivity of volumes to sampling. Salient interval volumes of one or more variables are used to select which parts of the histogram spectra are important. The level of detail selection problem, over a time-varying, multivariate, multiresolution volume, is then posed as an integer programming problem using the histogram spectra. We propose an efficient solution enabling interactive LOD selection on large, out-of-core volumes and show its efficacy on two real data sets from different problem domains.

Keywords: Level of detail selection, Multivariate volume visualization, Time-varying volume visualization.

Index Terms: Computer Graphics [I.3.6]: Graphics data structures and data types—Level of detail selection, Multivariate volume visualization, Time-varying volume visualization

1 INTRODUCTION

Large, time-varying, multivariate volumes are commonly encountered in scientific visualization. As the available compute power for simulation has increased, the quantity of data produced has increased commensurately. However, storage system throughput and latency have not improved at the same rate. Analysis tools such as volume renderers that seek to enable interactive visual analysis must scale to support interactivity on these larger data sets.

The ability to interactively rotate, translate, and focus in on time-varying volume simulation data can increase the understanding of the data. If the data is too large to fit into memory at its highest resolution, techniques must be applied to choose the subset, or level of detail, of the data that maximizes quality subject to the working set size constraints as determined by hardware resource availability.

All subsets of the value domain of the data are not necessarily of the same importance, and it is often possible for users to make informed guesses at what subsets are important. However, these informed guesses often need to be part of the interactive workflow. This means that level of detail selection must be done interactively.

For example, in the context of a weather simulation, the scientist may be interested in the vertical velocity of clouds. It is clear that, if we have a wind field defined over the entire volume and the clouds do not cover the entire volume, only a portion of the wind field is important. Similarly, a large portion of the volume variable that determines cloud density will also be unimportant where it is below the threshold for clouds. Thus, we can refine the level of detail selection based on intervals of interest, for both the cloud variable and the velocity variable, to maximize information density within the data loaded. By focusing only on the quality of a limited interval volume of the volume, we can attain higher quality than if levels of detail were selected in an interval-agnostic manner.

The best level of detail for a multiresolution data set is the one that minimizes the error over the intervals of interest subject to a size constraint. This introduces two challenges:

- Selection of the level of detail: General binary integer programming, which can be used for LOD selection, is NP-hard. We need to offer a more efficient alternative.
- Error estimation for intervals of interest: For level of detail selection, given intervals of interest, it is necessary to estimate the error introduced by downsampling. If no metadata is stored, the entire data set must be loaded every time the error is to be estimated for a new interval of interest. We need to generate metadata to facilitate faster error estimation.

This work provides two core contributions, one addressing each of these challenges. First, we introduce the novel concept of histogram spectra, which are used to estimate the statistical sensitivity of time-varying volumes to sampling. Histogram spectra are stored as metadata, enabling the estimation of error without having to access the volume data directly. Secondly, we introduce an efficient level of detail selection algorithm utilizing the linear relationship between histogram spectra predicted error and RMS error. Our technique enables fast, interactive LOD selection with reasonable preprocessing times and low implementation complexity.

This paper is organized as follows. §2 discusses previous work as related to our work. §3.1 through §3.4 introduce the concept of histogram spectra. Our solutions to the level of detail selection problem are discussed in §3.5 and §3.6. Considerations for applying the algorithm to multivariate data are discussed in §3.7. Finally, the results are examined in §4.

2 RELATED WORK

The challenge of interactively visualizing large data, both in scientific visualization and in general graphics, has inspired much previous work. Common to many of the methods is the concept of multiresolution data availability. Two critical aspects in dealing with multiresolution data are designing a multiresolution representation, and deciding which portions of the multiresolution data to load in an application.

Many approaches to multiresolution volume representation have been explored. Wavelets are a widely used method, offering multiple levels of detail with little or no space overhead. Westermann, et al. [18] developed a method for directly rendering wavelet transformed volume data. Wang, et al. [17] propose a method for rendering very large wavelet-transformed volume data and subsequently extended [16] the method to time varying volume data using a wavelet time-space partitioning tree. While wavelets can provide for efficient storage of large multiresolution volumes, they do have a disadvantage in that to access a given level of detail of a volume multiple levels of the wavelet hierarchy must be accessed.

An alternative to directly storing a multiresolution volume is to generate metadata that can be used to skip large portions of the high

*e-mail:martinst@cse.ohio-state.edu
†e-mail:hwshen@cse.ohio-state.edu

resolution volume that are not needed. For example, Gregorski, et al. [7] developed a method for preprocessing tetrahedral volumes such that diamonds of min-max values are identified. This enables fast reconstruction of isosurfaces subject to a user-specified error tolerance without having to visit the entire volume. However, this approach does not directly apply to our problem because we are performing general level of detail selection rather than computing isosurfaces. Instead, we are using interval volumes to weight the importance of different portions of a volume for the purposes of error estimation in level of detail selection.

More similar to our technique are techniques that downsample the volume into a multiresolution hierarchy, resulting in some data duplication, but at the same time enabling flexible reconstruction with minimal computational overhead and fewer reads. Gao, et al. [6] developed a distributed architecture for volume rendering of distributed data using multiresolution hierarchies while considering visibility to reduce data movement and applying prefetching in a load on need context. LaMar, et al. [9] use a multiresolution texture hierarchy to accelerate volume rendering on graphics hardware.

Rendering a multiresolution hierarchy requires the selection of a subset of blocks (level of detail) from the multiresolution hierarchy that need to be rendered and/or loaded. Conceptually, this can be thought of as the construction of a cut through the multiresolution hierarchy. In our technique we explicitly generate multiresolution cuts (or level of detail selections) through a 4D multiresolution volume. Boada, et al. [1] also directly generate cuts through a multiresolution hierarchy in the context of volume visualization. Gyulassy, et al. [8] also directly generate cuts through a multiresolution hierarchy but combine it with view-dependent error calculations. However, there are substantial differences between these methods and ours. First, their multiresolution volumes are 3D rather than 4D, which has implications for the complexity of the algorithms and the severity of errors introduced by the use of interpolation with lower levels of detail. Secondly, their cut construction algorithms construct the cuts in a top-down manner from the lowest level of detail. Construction of the cut from the lowest level of detail may sometimes be less computationally expensive than our optimization-based approach. However, especially in data sets with wide ranges of levels of detail like those we tested, it poses the potential of missing features at the higher levels of detail if the lowest level of detail is sufficiently undersampled and other measures are not taken.

Constructing the level of detail selection, regardless of whether it is constructed in an optimization-oriented or bottom-up manner, requires deciding when to decimate or refine the level of detail selection. Two general optimization approaches that can be taken are error-constrained and size-constrained. In the case of error-constrained approaches the objective is to minimize the size of the working set subject to the error constraint. In the case of size-constrained approaches the optimization function may seek to maximize importance or minimize error subject to a working set size constraint. Size-constrained approaches are more appropriate when hardware limits on the maximum interactive working set size are important in interaction.

Error-constrained approaches have been widely applied in visualization. Wang, et al. [16] consider both spatial and temporal error constraints in the rendering of wavelet data. Danskin, et al. [4] consider image-space error constraints on rendering error in volume ray tracing. Gregorski, et al. [7] consider error tolerances in the extraction of time-varying isosurfaces. These techniques contrast with ours in that ours is size-constrained rather than error-constrained. However, it is important to consider that even though our technique is not error-constrained the error can still be quantified in the results.

Size-constrained (and by extension, load time-constrained) approaches have also been widely applied in visualization, as well as general graphics applications. Saito [12] developed a time-constrained point rendering approach for previewing volumes and argues that constant frame rates are beneficial for interaction. Shin, et al. [14] developed a quadtree-based approach for fixed frame rate continuous LOD terrain visualization. Lindstrom, et al. [10] applies a height field simplification algorithm to keep a constant frame rate. Funkhouser, et al. [5] proposed an adaptive rendering algorithm that seeks to maintain a constant frame rate for virtual environment visualization. Certain, et al. [3] applies wavelets within a time-constrained context to maintain constant framerates for multiresolution surface viewing. While all of these works are from different contexts, they all consider consistency of frame rate and working set size to be important enough for interaction to make it a constraint on their level of detail choices.

The concept of importance (expressed via interval selection using the weighting function, in our technique) can help facilitate higher quality by allowing some subsets of the data to have higher priority over other subsets of the data. The concept of importance sampling has been widely used in visualization and rendering. Danksin, et al. [4] and Viola, et al. [15] both apply importance sampling in the context of volume rendering. Similarly to our technique, Martin, et al. [11] considers user-specified isovalues in computing a fixed distribution of work in the context of distributed-data isosurface computation, though that work does not consider multivariate data and does not pose the problem directly as an optimization problem.

While level of detail selection has been widely used and explored, we are not aware of a proposal for error prediction similar to the concept of histogram spectra, nor have we found a similar greedy solution to the level of detail selection problem that can efficiently utilize the histogram spectra for 4D, multivariate, multiresolution volumes.

3 Level of Detail Selection

Assume that a time-varying volume is divided into a set of 4D subvolumes (or "bricks".) Each subvolume is sampled into a set of levels of detail, each with a different sampling frequency. The goal of the level of detail selection algorithm is to select the level of detail for each subvolume that maximizes quality subject to a working set size constraint. The data flow of the level of detail selection process in our technique is exhibited in fig. 1.

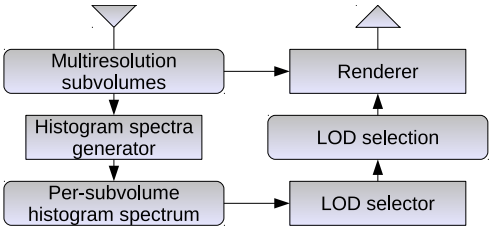


Figure 1: The histogram spectra generator takes a multiresolution bricked volume and generates a histogram spectrum for each subvolume ("brick") of the volume. This will be done as a precomputation step in the data preparation phase. The LOD selector then uses that, with a set of user-defined parameters such as intervals of interest, to produce a LOD selection set. The LOD selection can be performed interactively.

3.1 Histogram Spectra

Let $f_a(x)$ be the probability density function (PDF) of a subvolume sampled at sampling frequency a. The histogram spectrum of the subvolume is then a mapping $\mathbb{R}^2 \rightarrow \mathbb{R}$

$$h(x,a) = |f_b(x) - f_a(x)| \quad (1)$$

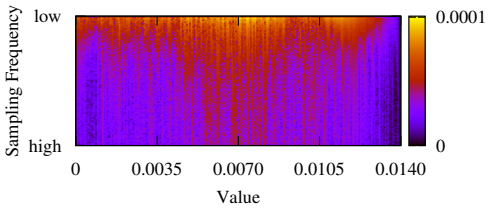


Figure 2: This histogram spectrum of a single plane of a single timestep of the QVAPOR variable of the climate test data set (defined in §4) is typical of histogram spectra. Moving up on the vertical axis corresponds to downsampling, and each column corresponds to the potential change in the area of an isosurface as a function of sampling frequency. Columns with brighter colors in this plot correspond to values that are more sensitive to sampling. Rows with brighter colors correspond to sampling frequencies with greater overall, unweighted, error.

where b is the sampling frequency of the highest level of detail of the subvolume and x is the value parameter to the PDF.

For a volume comprised of multiple subvolumes, the set of histogram spectra is comprised of the histogram spectrum of each subvolume. Each subvolume is processed independently of every other subvolume.

Evaluating $h(x,a)$ for a given x and a yields a value proportional to the absolute difference between the surface area of an isosurface with value x in the subvolume sampled with sampling frequency a and the surface area of an isosurface with value x in the subvolume sampled with frequency b. The relationship between isosurface area and histograms has been examined in depth by Scheidegger, et al. [13] and Carr, et al. [2]. If no information has been lost in value x by sampling with a frequency a versus sampling with a frequency b then $h(x,a) = 0$.

3.2 Weighted Histogram Spectra

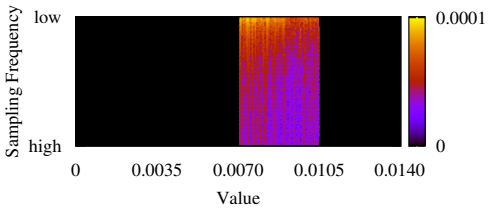


Figure 3: The weighting function is used to control the width of the interval volumes of interest in the context of the level of detail selection. In this example a weighting function was chosen to place importance on the interval of values from 0.0070 to 0.0105. The weighting function is applied over the columns of the histogram spectrum, facilitating the computation of histogram spectrum predicted error as in equation (4).

A weighting function, $w(x)$, is defined as an $\mathbb{R} \to \mathbb{R}$ mapping from the volume value domain to weights. Conceptually $w(x)$ should reflect the important interval volumes (intervals of interest) for the current visualization task, having a higher value within the interval volumes than outside the interval volumes.

The weighted histogram spectrum, defined for a subvolume as in equation (1), is then:

$$h_w(x,a) = w(x)h(x,a) \tag{2}$$

Evaluating $h_w(x,a)$ for a given x and a yields a value proportional to the weighting function $w(x)$ and the difference in isosurface surface areas as in equation (1). This is significant because it enables the estimation of the error in intervals of interest from the histogram spectrum using equation 4.

Typically, when $w(x)$ is defined directly by the user, it will be in the form of:

$$w(x) = \begin{cases} 1 & x \in Y \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where Y is a user-defined set of important values. For example, choosing $Y = 0.3$ would mean that error is only considered to be important if it affects an isosurface with isovalue 0.3. However, it is not required that $w(x)$ be in this form and indeed for multivariate data it can be useful for it to be in a different form, as can be seen in §3.7.

3.3 Predicted Error Using Histogram Spectra

The error of a scalar subvolume at a given sampling frequency a can be estimated using the histogram spectra via

$$E(a) = \int_{-\infty}^{+\infty} h_w(x,a)dx \tag{4}$$

Effectively this sums the difference in surface area for every isosurface in the subvolume, weighted by the user-specified weighting function $w(x)$.

The RMS error of a subvolume is proportional to $E(a)$. A linear relationship was observed, as in fig. 4, on our test cases for $0.1b < a \leq b$ – when the sampling frequency is greater than about one tenth the ground truth sampling frequency. In a 4D volume, one tenth the ground truth sampling frequency would equate to roughly a 10,000x reduction in size. This is data-dependent, but does demonstrate that histogram spectra can be used to predict RMS error resulting from a range of downsampling operations on real-world data sets. Most importantly for the purposes of the greedy algorithm discussed in §3.6, it means that a ratio between two RMS errors is the same as the ratio between the corresponding two histogram spectrum predicted errors.

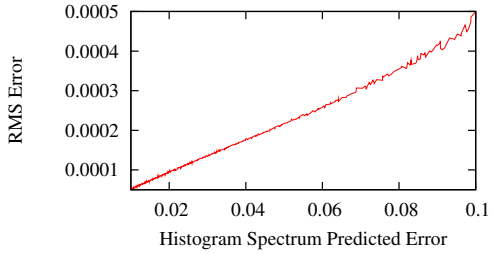


Figure 4: The RMS error is proportional to the histogram spectrum predicted error. This fig. exhibits a test case on the QVAPOR variable of the climate data set (defined in §4), and is typical of what we have observed on other data sets. The exact scaling factor to determine the RMS error depends on the units of the data in the field and the norm of the weighting function. However, this does not need to be computed because only the relative differences between errors need to be used in the algorithm discussed in §3.6. Because the RMS error is linearly proportional to the histogram spectrum predicted error, the ratio between two RMS errors is the same as the ratio between their corresponding histogram spectrum predicted errors.

3.4 Discretization of Histogram Spectra

In practice, for multiresolution data, only a finite number of levels of detail can be considered. Similarly, the resolution required for discrete forms of the probability density functions used in the histogram spectra is also limited.

In our implementation we store a uniformly sampled histogram spectrum for each subvolume as a 2D array of floats. The histogram resolution (the number of columns) determines the narrowest interval volume that can be considered for level of detail selection. Too few columns will reduce the effectiveness of the algorithm, while too many will waste space. It should be chosen to reflect the minimum width of intervals that the user is likely to be interested in for level of detail selection. In future work we may consider alternative sampling strategies for the histograms, if we can find an application that requires them.

When there are M levels of detail, M frequencies (rows) are stored for the histogram spectra. However, if the highest sampling frequency of a level of detail is the same as the ground truth sampling frequency, it is not necessary to store the rows of the histogram spectra corresponding to that level because they can be assumed to be zero.

The resulting floating point data can be compressed losslessly with floating point image compression techniques, but in our test cases the space consumed by the histogram spectra was not found to be large enough to warrant this.

3.5 Integer Programming Formulations for LOD Selection

The goal of the level of detail selection problem is to compute the level of detail index L_i for every subvolume i of N subvolumes such that the error is minimized and the size of the subvolumes to be loaded for the level of detail is below a threshold, $S_{\max}$. This can be structured as a nonlinear integer programming problem

$$\text{argmin}_L \sum_{i=1}^{N} E_i(a_{L_i}) \tag{5}$$

with the constraints

$$\sum_{i=1}^{N} S_{i,L_i} \leq S_{\max}; 1 \leq L_i \leq M; L_i \in \mathbb{Z} \tag{6}$$

where a_k is the sampling frequency for level of detail index k, $S_{i,j}$ is the load size for LOD j of block i, and M is the number of levels of detail. This optimization problem is nonlinear because the optimization arguments are used as arguments to the nonlinear $E_i(a)$ function within the objective function.

An alternate, equivalent, binary, linear integer programming formulation can be constructed by recognizing that there are a finite number of levels of detail:

$$\text{argmin}_H \sum_{i=1}^{N} \sum_{j=1}^{M} E_i(a_j) H_{i,j} \tag{7}$$

with the constraints

$$\sum_{i=1}^{N} \sum_{j=1}^{M} H_{i,j} S_{i,j} \leq S_{\max} \tag{8}$$

$$\sum_{j=1}^{M} H_{i,j} = 1; 1 \leq i \leq N; i \in \mathbb{Z} \tag{9}$$

It follows from equation (9) that the solution to the binary, linear integer programming problem is related to the nonlinear integer programming problem by:

$$H_{i,j} = \begin{cases} 1 & L_i = j \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

This is linear because the only potentially nonlinear part of the objective function, $E_i(a)$, is now dependent on a set of constants, the possible level of detail sampling frequencies, rather than the argument being optimized as in equation (5).

With equation (10) it can be seen that the constraint equations (8) and (6) as well as the objective functions (5) and (7) are respectively equivalent.

General linear programming packages such as the GNU Linear Programming Kit (GLPK) can be applied to solve the binary integer programming problem described in equation (7). However, general binary integer programming is NP-hard. Binary integer programming packages, such as that offered by GLPK, often integrate acceleration strategies to more efficiently solve special cases of general binary integer programming problems. However, we found (as in the results in §4.2) that these strategies were generally insufficient for attaining reasonable running times for interactive level of detail selection.

Instead, we propose a greedy algorithm for solving the nonlinear integer programming problem described in equation (5) that yields approximate solutions very close to the optimal solution, with considerably lower computational complexity. We still present the binary integer programming form because it provides a way to easily apply existing integer programming optimization packages such as GLPK to the problem for performance testing, and it provides for extensibility in future work.

3.6 Greedy Algorithm for Nonlinear Integer Programming Formulation

Because the number of subvolumes multiplied by the number of levels of detail, N, in a data set may be very large, a greedy approximation to the integer programming problem is more practical. For example, with 1 MiB subvolumes (which equates to roughly 23^4 univariate 4D blocks with 4 bytes per variable) and 8 levels of detail, a 1 TiB volume would have approximately 8 million unknowns in equation (7) and 1 million in equation (5). Even if a nonlinear but polynomial time direct solution was possible for the integer programming problem, the performance would still be insufficient for performing LOD selection during the interactive portion of the workflow.

Our approach is to consider the set of potential levels of detail for all subvolumes, then apply them to the subvolumes in order of increasing error density until the size constraint is satisfied. We propose a three step greedy algorithm for accomplishing this:

1. **Estimate the result error** for every LOD for every subvolume, as described in §3.6.1. This requires $\mathcal{O}(N)$ time using histogram spectra.
2. **Compute the error density values and sort** the potential LOD assignments by them, as described in §3.6.2. This requires $\mathcal{O}(NlgN)$ time.
3. **Assign the best LOD** to every block using the sorted list, as described in §3.6.3. This requires $\mathcal{O}(N)$ time.

3.6.1 Error estimation

Every possible level of detail selection for a subvolume has an associated estimate of the sampling error that would be present due to the choice of the level of detail. Whenever the intervals are changed via the weighting function, equation (3), the error estimate will need to be recomputed. If RMS error is computed directly, instead of using the precomputed histogram spectra with equation (4), the entire volume will need to be revisited to compute the error, which is impractical within an interactive workflow. However, if the histogram spectra are used to estimate error using equation (4) then only the histogram spectra need to be visited. The histogram spectra are much smaller than the entire volume and have already been computed during either the data preparation or data generation phases.

3.6.2 Sorting and the heuristic

A list is constructed containing an entry for every potential LOD assignment, for every subvolume. Each entry has a heuristic value (error density), an LOD index, and a subvolume index. The heuristic value used for subvolume i with LOD j, $A_{i,j}$, is defined as follows:

$$A_{i,j} = \frac{E(a_j)}{S_{i,j}} \tag{11}$$

where a_j is the sampling frequency of LOD j, and $S_{i,j}$ is the size in bytes loaded for LOD j of subvolume i. $E_i(a_j)$ is equation (4) evaluated for subvolume i or, alternatively, the directly computed RMS error which was used for the performance tests in §4.2.

This list is then sorted in ascending order of the heuristic, $A_{i,j}$. This results in a list of potential LOD assignments sorted by ascending error density.

3.6.3 LOD assignment

Conceptually the goal is to choose levels of detail that minimize error density, subject to a size constraint, $S_{\max}$. The following algorithm is applied to assign levels of detail using the sorted list:

Listing 1: LOD assignment algorithm

```
L         :=(list produced by sorting)
B         :=(LOD assignment for each subvolume)
N_subvols:=(number of subvolumes)
S_max     :=(maximum working set size)
S_total   :=N_subvols * getSubVolSizeForLOD(1)
for (i in 1..L.length) AND S_total>S_max
  if B[L[i].block]<L[i].lod
    S_total-=getSubVolSizeForLOD(B[L[i].block])
    B[L[i].block]:=L[i].lod
    S_total+=getSubVolSizeForLOD(B[L[i].block])
```

When the solution is feasible, we have found this algorithm produces results close to the optimal solution produced by directly applying general binary integer programming algorithms as can be seen in fig. 5. When the $S_{\max}$ constraint is too low for a feasible solution, it gracefully results in the lowest detail level being specified for all blocks.

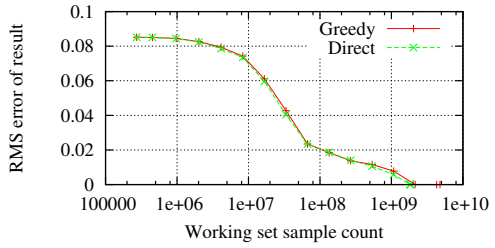


Figure 5: Directly solving the integer programming problem with a general integer programming package is impractical due to the high computational complexity involved in solving the NP-hard problem. Our greedy algorithm as described in §3.6 yields nearly identical results with $\mathcal{O}(NlgN)$ where N is linearly proportional to the number of subvolumes.

3.7 Multivariate Considerations

Multiple variables with different units are commonly used simultaneously in visualizations. For example, in a weather simulation we may be interested in volume rendering clouds in the context of a water vapor field. In this case, this implies that we want high levels of detail where lower levels of detail would introduce too much error in either the water vapor field, or the cloud field. These variables that are used to guide the selection of levels of detail are called *guiding variables*.

3.7.1 Optimization

Histogram spectra are generated for each *guiding variable*. Separate weighting functions are applied for each variable to produce weighted histogram spectra for each, enabling the estimation of error for each guiding variable independently using equation (4). Effectively the nonlinear integer programming problem in equations (5) and (6) can be extended to include C variables:

$$\text{argmin}_L \sum_{k=1}^{C} \sum_{i=1}^{N_k} E_{k,i}(a_{k,L_{k,i}}) \tag{12}$$

with the constraints

$$\sum_{k=1}^{C} \sum_{i=1}^{N_k} S_{k,i,L_{k,i}} \leq S_{\max} \tag{13}$$
$$1 \leq L_{k,i} \leq M; L_{k,i} \in \mathbb{Z}$$

where $a_{k,j}$ is the sampling frequency for level of detail index j of variable k, $S_{k,i,j}$ is the load size of LOD j of block i of variable k, N_k is the number of subvolumes in variable k of the volume, M is the number of levels of detail, and C is the number of variables. The binary linear integer programming formulation in equations (7), (8), and (9) can be similarly extended:

$$\text{argmin}_H \sum_{k=1}^{C} \sum_{i=1}^{N_k} \sum_{j=1}^{M} E_{k,i}(a_{k,j}) H_{k,i,j} \tag{14}$$

with the constraints

$$\sum_{k=1}^{C} \sum_{i=1}^{N_k} \sum_{j=1}^{M} H_{k,i,j} S_{k,i,j} \leq S_{\max} \tag{15}$$

$$\sum_{j=1}^{M} H_{k,i,j} = 1; 1 \leq i \leq N; i \in \mathbb{Z}; 1 \leq k \leq C; k \in \mathbb{Z}$$

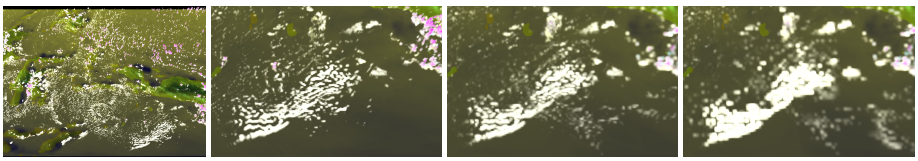
The optimization solutions presented in §3.6 apply identically to this multivariate case as they do to the univariate case. The multivariate forms' objective functions are defined as the sum of multiple univariate objective functions. Similarly, the multivariate forms' constraints are the logical conjunction of multiple univariate constraint sets. When C is 1 the multivariate optimization equations reduce to the univariate optimization equations.

3.7.2 Conditional importance

Sometimes there are variables that are only important where the *guiding variables* are within a particular interval. We refer to these variables as *following variables*. For example, consider the case where a user wants to see the vertical velocity of clouds in the context of a volume rendering where the cloud density defines the opacity and the cloud color is defined by the vertical velocity. In this case the *guiding variable* is the cloud density and the vertical velocity is the *following variable*.

From the standpoint of optimization the guiding variables and following variables are treated identically. However, the weighting functions for following variables should take into account conditioning by the guiding variables. In many circumstances, such as in the above cloud velocity example, the probability distribution of the following variable within the interval of interest of the guiding variable is different from the probability distribution of the following variable over the entire volume. This has been observed in our test data sets, as can be seen in fig. 7.

While it is not absolutely required, for following variables, the weighting function should be chosen to assign increased weight where the conditional probability density of the following variable is high. This will reduce the relative importance, with respect to



(a) Ground truth (b) Ground truth, zoom (c) Narrow intervals, zoom (d) Wide intervals, zoom

Figure 6: Several variables from the climate data set are rendered for a single timestep. The white, opaque parts are clouds defined by the QCLOUD variable. The magenta regions are clouds with high vertical velocities, as determined by the W variable. The yellow exhibits water vapor density as determined by the QVAPOR variable. The volume is a curvilinear volume, with the Z variable of its mesh determined by the MESHZ variable. All of these variables have their levels of detail determined by the level of detail algorithm. Figures 6a and 6b are generated from the ground truth resolution, while figures 6c and 6d have levels of detail selected for a 4GiB working set size constraint. Figure 6c was generated with narrow intervals of interest, while fig. 6d was generated with wide intervals of interest. Like in fig. 9, selecting narrow intervals of interest yields results closer to the ground truth than selecting wide intervals of interest.

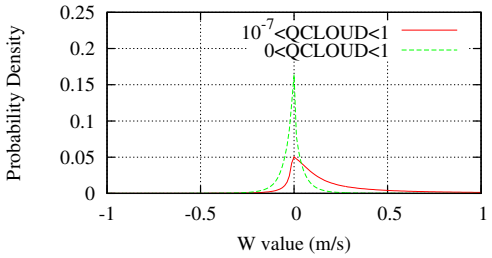


Figure 7: In some cases, with multivariate fields, a user is interested in seeing a variable A where variable B is between B_0 and B_1. This interval $[B_0 : B_1]$ is expressed as a weighting function for the histogram spectra of B. The choice of the best weighting function for A depends on the statistical dependence between A and B. If A is not independent of B then we can use the conditional probability density function of A given the case that B lies within $[B_0 : B_1]$ as a starting point for constructing a weighting function for A. In this example it can be seen that the PDF of the vertical velocity(W) in the climate data set is different for different intervals of the cloud density(QCLOUD.)

level of detail selection, assigned to values that fall outside of guiding variable interval volumes of interest. We have found that this works effectively as can be seen in the climate data test case in the results.

4 Results

Fundamentally, the goal of our technique is to permit interactive selection of levels of detail on data sets much larger than can be fit in-core or loaded interactively. This enables users to interactively select levels of detail that focus on the intervals of interest within time-varying volumes, which offer increased quality for a given sample size constraint. We performed experiments to look at three aspects of this: running times, visual quality, and statistical quality.

4.1 Test data sets

Two data sets were used for experiments, one being from a climate simulation at Pacific Northwest National Laboratory and the other being from a turbulent combustion simulation at Sandia National Laboratory. Both were time-varying volume data sets.

4.1.1 Climate

The climate data set is a set of multivariate volume timestep snapshots from a long-term weather simulation of the region around Indonesia in the context of climate change research. The 4D multiresolution data set used for the purposes of the experiments was 117GiB with 8 levels of detail, sampled on a time-varying geopotentially-defined curvilinear mesh with 41 timesteps broken into 18,944 4D subvolumes. The data set contained variables for geopotentially-defined elevation(MESHZ), cloud density(QCLOUD), water vapor density(QVAPOR), and vertical velocity(W).

The MESHZ variable was comprised of 1,184 4D subvolumes with 8 levels of detail ranging from 5.8MiB per subvolume to 64 bytes per subvolume. The QCLOUD, QVAPOR, and W variables were each comprised of 5,920 4D subvolumes with 8 levels of detail ranging from 1.3MiB per subvolume to 64 bytes per subvolume. The total size of the histogram spectra, discretized into 128 histogram bins and 7 frequencies per subvolume, for the entire data set, was 67MiB, 0.056% of the size of the multiresolution volume. Additional static 2D variables included for the purposes of producing renderings were the land elevation, vegetation fraction, and surface normals.

4.1.2 Combustion

The combustion data set is a set of volume timestep snapshots from a simulation of the injection of fuel into two countercurrent air streams in which combustion occurs. A single variable, the mixing fraction (referred to as MIXFRAC), was used for the purposes of testing. The 4D multiresolution data set, defined on a regular grid, was 69GiB with 8 levels of detail and 121 timesteps broken into 17,010 subvolumes. Levels of detail ranged from 1.1MiB per block to 64 bytes per block. The total size of the histogram spectra, discretized into 128 histogram bins and 7 frequencies per subvolume, for the entire data set was 60MiB, 0.085% of the size of the multiresolution volume.

4.2 Running time comparisons

The two major bottlenecks to interactive LOD selection for varying intervals of interest are the load time for error estimation, and the computation time for solving the optimization problem to minimize error for a given size constraint.

The test platform was a Linux PC with an Intel Core 2 6600 dual core CPU, 4GiB of main memory, and a hard drive with the IBM JFS filesystem capable of approximately 30MiB/s with 5-10ms latency for data storage.

4.2.1 Error estimation

For the error estimation aspect, we compare using the histogram spectra versus directly estimating the RMS error from the data. Estimating the error with the histogram spectra predicted error (HSPE) only requires loading the substantially smaller discretized

histogram spectrum for each subvolume. Estimating the error directly with RMS error (RMSE) requires loading the entire data set every time the LOD changes.

In the following tests LOD selection was performed for a size constraint and a target interval on the climate QVAPOR and combustion MIXFRAC variables, in conjunction with our greedy algorithm for the LOD selection. The size constraint choice and target interval choice do not affect the timing results.

Heuristic	Data set	LOD Selection Time
RMSE/size	QVAPOR	1782.0s
HSPE/size	QVAPOR	0.1s
RMSE/size	MIXFRAC	3900.3s
HSPE/size	MIXFRAC	0.2s

Using HSPE to compute the error in the heuristic, $A_{i,j}$, clearly outperforms using RMSE on both data sets. This is because using HSPE only depends on the histogram spectra, which have already been precomputed in the non-interactive portion of the workflow. In contrast, RMSE requires reading the entire volume data set every time the list described in §3.6.2 is constructed. Figure 8 shows the typical relationship between the error performance of the HSPE and RMSE error estimators.

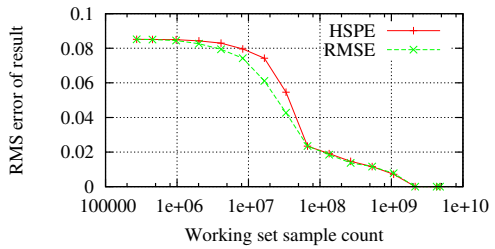


Figure 8: This fig. shows the error for different working set size constraints, using different error estimators in the LOD selection algorithm. The E_j function in the optimization problem as referenced by equation (5) can be approximated using equation (4) instead of directly computing the RMS error (RMSE). The prediction of error using the histogram spectra predicted error (HSPE) yields results close to the direct RMS error. By using equation (4) with histogram spectra it is possible to avoid loading samples from the source volume when performing LOD selection, substantially improving performance.

4.2.2 Optimization

For the optimization aspect, we compare our greedy approximation to a direct integer programming approach. While general binary integer programming is an NP-hard problem, packages like GLPK apply some techniques to improve performance. Further information about the techniques GLPK applies can be found in the GLPK source code.

For GLPK with the QVAPOR data set, binary linear integer programming was performed with 47,360 variables. The greedy algorithm required only 5,920 entries to be sorted because only one is needed per block, rather than one per block per LOD. For the MIXFRAC data set, binary linear integer programming was performed with 136,080 variables, and the greedy algorithm required only 17,010 entries to be sorted.

The running time for GLPK is sensitive to the target interval of interest, while the running time for the greedy algorithm is unaffected by the target interval of interest. This is because the techniques GLPK can apply depend on the coefficients in the linear programming problem, while the greedy approximation simply sorts by the heuristic then assigns the levels of detail. In all cases the GLPK performance was slower.

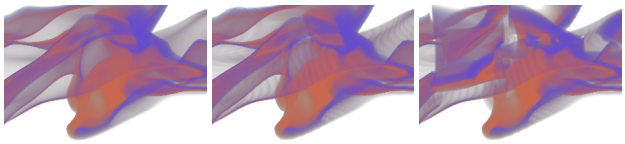
Solver	Data set	Solving time
GLPK	QVAPOR	7625.6s
Greedy	QVAPOR	0.1s
GLPK	MIXFRAC	142.1s
Greedy	MIXFRAC	0.2s

The greedy algorithm substantially outperforms GLPK. The reason why the greedy algorithm is much faster is because its computational complexity is $\mathcal{O}(NlgN)$ as discussed in §3.6. This stands in contrast to the general integer programming methods applied by GLPK which, for nontrivial inputs, are much worse than polynomial time and exponential in the worst case. The result of the greedy algorithm was also found to consistently be very close to the to that of GLPK, as can be seen in fig. 5.

4.2.3 Histogram Spectra Computation

Computation of the discrete histogram spectrum for a single subvolume requires the computation of a histogram for different sampling frequencies for the subvolume. Each subvolume can be processed independently, yielding an embarrassingly parallel streaming algorithm that can be easily implemented on GPUs, multi-core, and/or multi-node platforms. Because of this, the computation of histogram spectra is likely to be read-bound, rather than compute-bound, on most system configurations. However, placement within the visualization workflow will determine the true cost of this operation.

If the histogram spectra computation is done in-situ, during the data generation phase, no additional reads are required because the histogram spectra can be computed as the data is written out to disk. If the histogram spectra computation is done as a separate pass during the data preparation phase then the volume needs to be streamed in from disk storage once, in its entirety. It is likely that software engineering considerations specific to each application will dictate which approach is appropriate. In either case, the computation process scales linearly with the number of data samples.



(a) Ground truth (b) Narrow intervals (c) Wide intervals

Figure 9: Values of MIXFRAC from the combustion data set within the range $[0.45 : 0.55]$ are rendered for a single timestep, where values less than 0.5 are blue and those greater than or equal to 0.5 are orange. Figure 9a is a crop of an image generated using the ground truth resolution, while figures 9b and 9c have levels of detail selected for a 250MiB working set size constraint. Figure 9b has a weighting function that is 1 for values in the range $[0.45 : 0.55]$ and 0 elsewhere. Figure 9c has a weighting function that is uniformly 1. The narrower interval of interest used for fig. 9b clearly yields a result closer to the ground truth than the wide interval of interest that was used for fig. 9c.

4.3 Visual and statistical comparisons

Both from a statistical and visual standpoint, choosing narrow, salient intervals to focus on for error reduction yields improved quality. The univariate case was tested using the combustion data set, and the multivariate case was tested using the climate data set. Narrow interval widths are the minimal interval widths needed to cover the non-transparent portions of the color-opacity transfer functions used for producing the figures, while wide interval widths cover the entire value domain.

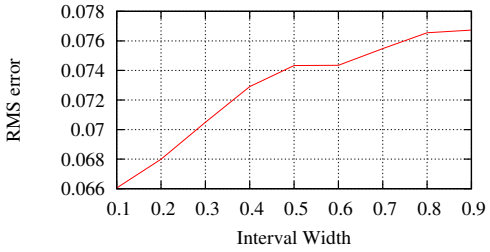


Figure 10: For a fixed working set size constraint, increasing the width of the range of values defining the interval volumes of interest results in increased error. This figure, which was generated using the QVAPOR variable of the climate test data set defined in §4, is typical of what we have observed. This is to be expected because a larger interval volume will encompass more samples yet the information density is likely to remain similar. Thus, the narrower the interval volume of interest, the fewer samples are needed to reconstruct the volume with a given level of error.

4.3.1 Combustion

Level of detail selection operations were performed for different interval widths and centers on the MIXFRAC variable of the combustion data set. Figure 10 exhibits the typical dependence observed of the RMS error on the width of the interval volume of interest as defined by a user via the weighting function. For a fixed working set size constraint, increasing the width tends to result in increased error. This is reasonable, because a larger interval volume will encompass more samples yet the information density is likely to remain similar.

The implications of this increased error can be observed in fig. 9. Artifacts are typical of block-wise downsampling, with a smoothing effect on the data and discontinuities at block (or subvolume) boundaries. Animations of the time series exhibit the improvement of the narrow interval over the wide interval more dramatically than the images. Choosing narrower intervals clearly yields images closer to the ground truth.

4.3.2 Climate

Similarly to the combustion data set, level of detail selection operations were performed for different interval widths. However, multiple variables were considered simultaneously. The QCLOUD, QVAPOR, and MESHZ were *guiding variables* while the W was a *following variable*, as described in §3.7. Using the guidelines in 3.7.2, the weighting function for the *following variable*, W, was conditioned by QCLOUD using the conditional PDF in fig. 7.

Figure 6 exhibits the results. Like the combustion data set in fig. 9, the quality was higher for narrower intervals. The figures producing using narrower intervals of interest are more similar to the ground truth than those produced using wider intervals of interest.

5 Conclusion

We have introduced the concept of histogram spectra as a new approach for efficiently estimating error due to downsampling for interval volumes of time-varying, multivariate, multiresolution volumes. A new optimization approach for level of detail selection was then introduced taking advantage of the linear relationship between the histogram spectra predicted error and RMS error. Both the optimization approach and the histogram spectra are easy to implement in software, increasing the practical applicability of our approach.

These contributions enable interactive level of detail selection on large, multivariate, multiresolution volumes for user-specified intervals of interest. By enabling the interactive selection of intervals of interest for the purposes of level of detail selection, increased visual and statistical quality can be obtained.

References

[1] I. Boada, I. Navazo, and R. Scopigno. Multiresolution volume visualization with a texture-based octree. *The Visual Computer*, 17(3):185–197, 2001.

[2] H. Carr, D. Brian, and D. Brian. On histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics*, 12:1259–1266, September 2006.

[3] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 91–98, New York, NY, USA, 1996. ACM.

[4] J. Danskin and P. Hanrahan. Fast algorithms for volume ray tracing. In *VVS '92: Proceedings of the 1992 workshop on Volume visualization*, pages 91–98, New York, NY, USA, 1992. ACM.

[5] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 247–254, New York, NY, USA, 1993. ACM.

[6] J. Gao. Distributed data management for large volume visualization. In *in Proc. IEEE Visualization*, pages 183–189. IEEE Computer Society Press, 2005.

[7] B. Gregorski, J. Senecal, M. A. Duchaineau, and K. I. Joy. Adaptive extraction of time-varying isosurfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(6):683–694, 2004.

[8] A. Gyulassy, L. Linsen, and B. Hamann. Time- and space-efficient error calculation for multiresolution direct volume rendering, 2005.

[9] E. LaMar, B. Hamann, and K. I. Joy. Multiresolution techniques for interactive texture-based volume visualization. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 355–361, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[10] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. A. Turner. Real-time, continuous level of detail rendering of height fields. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 109–118, New York, NY, USA, 1996. ACM.

[11] S. Martin, H.-W. Shen, and P. McCormick. Load-balanced isosurfacing on multi-gpu clusters. In *EGPGV '10: Proceedings of Eurographics Symposium on Parallel Graphics and Visualization 2010*, pages 91–100, May 2010.

[12] T. Saito. Real-time previewing for volume visualization. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 99–106, New York, NY, USA, 1994. ACM.

[13] C. E. Scheidegger, J. M. Schreiner, B. Duffy, H. Carr, and C. T. Silva. Revisiting histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics*, 14:1659–1666, November 2008.

[14] B.-S. Shin and E.-K. Choi. An efficient clod method for large-scale terrain visualization. In *ICEC*, pages 592–597, 2004.

[15] I. Viola, A. Kanitsar, and M. E. Groller. Importance-driven volume rendering. In *VIS '04: Proceedings of the conference on Visualization '04*, pages 139–146, Washington, DC, USA, 2004. IEEE Computer Society.

[16] C. Wang, J. Gao, L. Li, and H.-W. Shen. A multiresolution volume rendering framework for large-scale time-varying data visualization. In *Volume Graphics*, pages 11–19, 2005.

[17] C. Wang, J. Gao, and H.-W. Shen. Parallel multiresolution volume rendering of large data sets with error-guided load balancing. In *EGPGV*, pages 23–30, 2004.

[18] R. Westermann. A multiresolution framework for volume rendering. In *VVS '94: Proceedings of the 1994 symposium on Volume visualization*, pages 51–58, New York, NY, USA, 1994. ACM.