Transformations for Volumetric Range Distribution Queries

Steven Martin*
The Ohio State University

Han-Wei Shen†
The Ohio State University

ABSTRACT

Volumetric datasets continue to grow in size, and there is continued demand for interactive analysis on these datasets. Because storage device throughputs are not increasing as quickly, interactive analysis workflows are becoming working set-constrained. In an ideal workflow, the working set complexity of the interactive analysis portion of the workflow should depend primarily on the size of the analysis result being produced, rather than on the size of the data being analyzed. Past works in online analytical processing and visualization have addressed this problem within application-specific contexts, but have not generalized their solutions to a wider variety of visualization applications. We propose a general framework for reducing the working set complexity of the interactive portion of visualization workflows that can be built on top of distribution range queries, as well as a technique within this framework able to support multiple visualization applications. Transformations are applied in the preprocessing phase of the workflow to enable fast, approximate volumetric distribution range queries with low working set complexity. Interactive application algorithms are then adapted to make use of these distribution range queries, enabling efficient interactive workflows on large-scale data. We show that the proposed technique enables these applications to be scaled primarily in terms of the application result dataset size, rather than the input data size, enabling increased interactivity and scalability.

Keywords: Histograms, Range queries, Metadata generation, Volume synopsis

Index Terms: Computer Graphics [I.3.6]: Graphics data structures and data types—Volume range queries, metadata generation, histogram range queries

1 INTRODUCTION

Volumetric datasets continue to grow in size, and there is continued demand for interactive analysis on these datasets. Storage capacities and compute capabilities have also increased in workstation environments, but the storage throughputs and core memory sizes available have not increased at a similar rate. This means that an increasing number of analysis applications are becoming limited by the size of the data required by the algorithm, rather than by the computation speed or out-of-core storage device capacities available.

Many analysis applications perform data reduction – reducing a subset of data from a large-scale dataset to a much smaller dataset. For example, in volume rendering, a 3D volume is reduced to a 2D image, where the size of the image is typically considerably smaller than the size of the volume. An ideally scalable algorithm, for large-scale data, would have an asymptotic working set complexity in terms of the image size, rather than in terms of the volume size.

The working set of an algorithm is the set of data elements required for its execution during a time interval [7]. Assuming that all of the data (with N elements) has a contribution to the solution of an analysis application, it is unrealistic to expect that, in general, we can change the asymptotic working set complexity, for the entire time span of workflow, to be less than $\mathcal{O}(N)$. However, it may be possible to change the working set complexity of the interactive analysis portion of the workflow by applying data transformations in the preprocessing phase. This can facilitate scalable interactivity by making the working set complexity of the interactive portion primarily depend on the result size, rather than the size of the input data.

One approach to tackling this challenge is to perform a preprocessing pass that reduces the complexity for traditional analysis methods applied in the interactive phase. This is the single-ended transformation approach. For example, a volume could be downsampled to contain only as many samples as contained by the images being rendered. It could then be directly rendered using traditional volume rendering algorithms. The advantage to this approach is that the interactive portion of the workflow will not require any changes to have reduced working set complexity. However, the major downside is the amount of sampling error that will be introduced for most volumes.

Another approach is to introduce specialized analysis algorithms for the interactive phase of the workflow, in addition to a preprocessing phase. This is the dual-ended transformation approach. In keeping with the volume rendering example, one example of this approach is Fourier Volume Rendering (FVR) [21]. Using FVR, the working set complexity of the direct volume rendering algorithm is reduced[1] from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$ during the interactive phase, with an $\mathcal{O}(N^3)$ preprocessing pass. In this example, several points can be illustrated.

Firstly, the overall computational complexity is greater than the working set complexity in this case ($\mathcal{O}(N^3 logN^3)$ vs. $\mathcal{O}(N^3)$). But, for the interactive portion of the workflow, the working set complexity and computational complexity both decrease from $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2)$ and $\mathcal{O}(N^3)$ to $\mathcal{O}(N^2 logN^2)$, respectively. Secondly, some flexibility in volume rendering has been sacrificed for the sake of interactivity, but this may be acceptable for many users. Thirdly, the overall working set complexity remains the same, as can be expected, but the working set complexity of the interactive portion of the workflow has decreased considerably. Finally, this data transformation facilitates only a fairly limited set of data analysis approaches. In summary, some flexibility and preprocessing time has been sacrificed for increased interactivity by adapting the application algorithm as well as introducing preprocessing. This exemplifies the approach we take.

An increasing number of analysis applications are considering histograms, or other summary statistics, as their input, rather than just the input volume. For example, instead of computing isosurfaces for every cell in the volume, a lower resolution volume characterizing the density of isosurfaces ("fuzzy isosurfaces") can be computed using histograms [29]. Another example is Histogram Spectra, where the differences between histograms are used to characterize error in level of detail selections [22].

These applications depend on distribution range queries. A distribution range query evaluates an estimate of the probability density function (PDF) of the values contained within a rectangular cuboid region of a volume. We propose an approach that enables ef-

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

[1] for an NxNxN volume to be rendered into an NxN image

ficient evaluation of distribution range queries on multivariate volume data. This is accomplished by generating metadata during the preprocessing phase, then loading it on-demand for queries in the interactive phase. For multiple applications this enables the working set complexity to be primarily a function of the analysis result size, rather than the size of the input data.

This core contribution has three parts. The first, discussed in §3.2, is a generalization of integral histograms to the continuous domain and to multivariate volumes, **integral distributions**. The second, discussed in §3.4, is a decomposition of these integral distributions into a hierarchical structure, **span distributions**, that facilitates effective storage as metadata. The third is a proposal, in §4, for how to apply the technique for improved working set complexity in a few different applicatios with accompanying analyses.

Algorithms are provided both for the construction of metadata in the preprocessing phase, and for the servicing of queries using this metadata in the interactive analysis phase. To show the generality of the benefits of the approach, a working set complexity analysis is provided for two applications using this metadata. We believe that this work provides a good foundation on which to build scalable analysis applications.

2 Related Work

Range queries have been widely explored in the field of Online Analytical Processing (OLAP) and are beginning to be explored in more detail in the field of Visualization. This section will overview some of the higher level techniques that have been applied in OLAP to answer range queries in general, as well as techniques that focus more on computing distributions of ranges. Additionally, techniques in visualization and graphics that facilitate of distribution range queries will also be discussed.

In OLAP, it is typical for range queries to be for simple results such as a summation or maximum. However, these can be viewed merely as specific types of aggregation operators. Similarly, distribution range queries are also a type of aggregation operator. Thus, while most of these techniques seek to solve range sum queries, some of them can be adapted to distribution range queries. One group of techniques [10] [33] applies wavelet decomposition to the space to approximate sums. Hou et al. [13] proposes cosine transforms as an alternative. These techniques work well for scalar data. The authors also introduce the concept of approximate reconstructions that sacrifice spatial accuracy.

Another group of techniques [8] [11] generate histograms to approximate scalar data for the purposes of range queries. These techniques greatly depend on the bounds they choose for the regions they approximate with histograms. Koudas et al. [18], Karras [15], and Poosala et al. [26] focus more on the aspect of the problem involving the choice of bounding volumes. While these techniques can be motivating applications for the use of histogram range queries, as discussed in §4.1, they would be difficult to directly apply to answering distribution queries and may consume a considerable amount of space. Hixels [29] are a simple case, where the volume is broken into blocks over which histograms are computed. This has a disadvantage in that it must have a high resolution (and consequently a large working set) to support range queries with varying spatial positions and scales.

Prefix sum-based techniques in OLAP [9] [2] motivate the approach we have taken to answering distribution range queries. Fundamentally, these techniques compute a prefix sum then perform a series of additions and subtractions between prefix sum values to compute a range sum. Summed-area tables, in computer graphics, apply the same basic concept to texture mapping [6]. Integral histograms [27] extend this summed area table approach to supporting histogram range queries in the context of images. Much of the work in OLAP in this area focuses on facilitating fast updating of the prefix sum data, rather than just fast queries, which introduces a design compromise that we do not necessarily need to make in the visualization context. However, those same works [4] [9] [19] do introduce concepts that support subdivision of a volume into subdomains for the purposes of improving space and time complexity, as well as increasing parallelism.

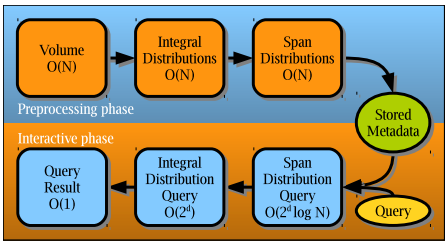


Figure 1: The preprocessing phase transforms the volume data into metadata using the transformation pipeline in equation (2). This requires $\mathcal{O}(N)$ working set complexity, for a volume with N elements. In the interactive phase, queries for distribution range queries are evaluated by reading parts of the metadata on demand into the transformation pipeline in equation (3). The working set complexity for this phase depends primarily on the result query size rather than the size of the input volume.

Integral distributions generalize existing techniques to multivariate volumes to support reductions in working set complexity for the interactive portion of multiple applications. Span distributions leverage spatial coherence in integral distributions to further reduce working set sizes, as well as supporting hierarchical, multiresolution approximate queries. The next section explains the details of both of these techniques.

3 Technique

The goal of the technique is to facilitate working set-efficient distribution range queries of volumetric data. $Q(\vec{s_0},\vec{s_1},\vec{t}) : \mathbb{R}^{2d+m} \rightarrow \mathbb{R}$ is the probability density for the vector value t within the region of the vector field $V : \mathbb{R}^d \rightarrow \mathbb{R}^m$, with m-component vectors, bounded by the points $\vec{s_0}$ and $\vec{s_1}$:

$$Q(\vec{s_0},\vec{s_1},\vec{t}) = \frac{\int_{\vec{s_0}}^{\vec{s_1}} h(V(\vec{s}),\vec{t})d\vec{s}}{\int_{\vec{s_0}}^{\vec{s_1}} 1 d\vec{s}}$$
$$h(\vec{u},\vec{t}) = \begin{cases} 0 & : \quad u \neq t \\ 1 & : \quad u = t \end{cases} \tag{1}$$

where the integrals are volume integrals and d is the dimensionality of the volume. In the context of scientific volume data, the vector field V typically contains the contents of the m dependent variables.

Direct evaluation of equation (1) for a discretized volume requires a working set of size $\mathcal{O}(N)$, where N is the number of sample points within the bounding box of interest. For interactive queries on large-scale data this is impractical. Our technique reduces this working set for each query by transforming V into **span distributions**, which are a hierarchical repesentation of **integral distributions**, in the preprocessing phase. This enables efficient evaluation of an **integral distribution field** $W(\vec{s}) : \mathbb{R}^{d+m} \rightarrow \mathbb{R}$. Then, W is used, instead of V, in an alternative formulation of equation (1), to evaluate Q. Because evaluating Q using W instead of V requires far fewer values, the working set complexity is reduced. The high level process is shown in figure 1.

The integral distribution field is a mapping from $\mathbb{R}^{d+1}$ to $\mathbb{R}$, rather than being defined in terms of a discrete domain. Additionally, the above equations are formulated in terms of general probability density functions rather than probability mass functions (which can be represented by histograms). Discretization and storage strategies must be considered for both.

3.1 High level overview

The two phases of the proposed framework, the preprocessing phase and the interactive phase, are shown in figure 1. Metadata is generated in the preprocessing phase using a series of transformations:

$$V(\vec{s}) \xrightarrow{I} W(\vec{s},\vec{t}) \xrightarrow{D} X_i(\vec{s}) \xrightarrow{S} Y_{k,i} \tag{2}$$

where D is a distribution value discretization function, introduced in §3.3.1. S is a spatial discretization function, such as span distributions, introduced in §3.3.2. X is a value-discrete, spatially-continuous representation of W and Y is a value-discrete, spatially-discrete representation of X. The integral distribution function, W, is introduced in section 3.2.

This metadata is loaded on-demand to evaluate queries. A sequence of transformations are applied for each query:

$$\dot{Y}_{k,i} \xrightarrow{S^{-1}} \dot{X}_i(\vec{s}) \xrightarrow{D^{-1}} \dot{W}(\vec{s},\vec{t}) \rightarrow \dot{Q}(\vec{s_0},\vec{s_1},\vec{t}) \tag{3}$$

where the dotted functions depend on only a subset of the metadata, rather than the original Q function in equation (1). The inverse discretization functions needed for evaluating queries are introduced in sections 3.3.1 and 3.3.2.

3.2 Integral Distribution Function

The integral distribution function maps a point in a multivariate volume to the distribution of the volume between that point and the origin of the vector field. This is an extension of integral histograms [27], which themselves are an extension of the use of 2D prefix sums in graphics (summed area tables [6]) and multidimensional prefix sums in OLAP [9] [2].

The integral distribution function I is defined as:

$$I(\vec{s},\vec{t}) = Q(0,\vec{s},\vec{t}) \tag{4}$$

where Q is from equation (1). This can be used to transform a vector field $V : \mathbb{R}^{d+m} \rightarrow \mathbb{R}$ into an integral distribution field, W:

$$\begin{gathered} W(\vec{s},\vec{t}) = I(\vec{s},\vec{t}) \\ (\forall t \in \mathbb{R}^m) \wedge (\forall s \in U) \end{gathered} \tag{5}$$

where U is the set of positions in the domain of V, the input volume.

This is the intermediate representation that our metadata seeks to represent, though without directly storing it. For the sake of clarity, let $\dot{W}$ be equivalent to W, but computed using the metadata, rather than by directly evaluating equation (5). In other words, any evaluation of W produces the same value as $\dot{W}$, but $\dot{W}$ depends only on the metadata, while W depends on evaluating Q.

With the integral distribution field, an alternative to equation (1) can be constructed to produce Q. For a one dimensional volume, where the domain of V is $\mathbb{R}$, this is simply:

$$\dot{Q}(s_0,s_1,\vec{t}) = \dot{W}(s_1,\vec{t}) - \dot{W}(s_0,\vec{t}) \tag{6}$$

Because $\dot{W}$ is known a priori from the metadata, Q can effectively be evaluated simply by using two lookups into $\dot{W}$, rather than by evaluating an integral over V. This can trivially be extended to the 2D case, where the domain of V is $\mathbb{R}^2$:

$$\dot{Q}(\vec{s_{00}},\vec{s_{11}},\vec{t}) = \dot{W}(\vec{s_{00}},\vec{t}) + \dot{W}(\vec{s_{11}},\vec{t}) - \dot{W}(\vec{s_{10}},\vec{t}) - \dot{W}(\vec{s_{01}},\vec{t}) \tag{7}$$

Generalizing to vector fields with $\mathbb{R}^d$ domains, this becomes:

$$\dot{Q}(\vec{s_{\{0\}^d}},\vec{s_{\{1\}^d}},\vec{t}) = \sum_{i \in \{0,1\}^d} (-1)^{d-||i||_1} \dot{W}(\vec{s_i},\vec{t}) \tag{8}$$

where $\{\mathscr{S}\}^d$ raises the set $\{\mathscr{S}\}$ to the d^{th} Cartesian power. For example, $\{0,1\}^2$ is $\{\{0,0\},\{0,1\},\{1,0\},\{1,1\}\}$. For an evaluation of $\dot{Q}$ in d spatial dimensions, 2^d lookups into the W field are required. This formulation of $\dot{Q}$ is similar to the formulation used for integral histograms [27], but generalized to a continuous domain.

In this work we do not seek to store the integral distribution field directly. Rather, we discretize it then decompose it into a more appropriate format for large-scale data. This involves two related parts: discretization of the field in terms of the spatial component($\vec{s}$) and a discretization of the field in terms of the value component($\vec{t}$). The next section addresses both aspects.

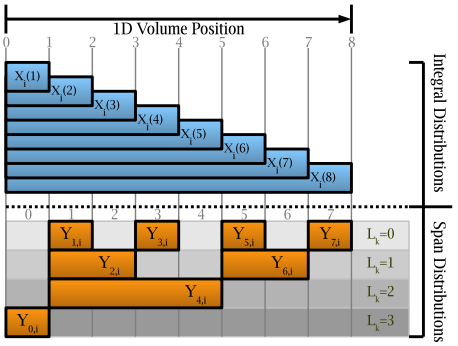


Figure 2: In this example, a 1D integral distribution volume ($X_i(s)$) is discretized into 8 span distributions ($Y_{k,i}$) as described in equation (9). The span distribution at index 6, for example, is computed by subtracting $X_i(5)$ from $X_i(7)$.

3.3 Discretization

The integral distribution field is an $\mathbb{R}^{d+m} \rightarrow \mathbb{R}$ mapping with two parameters: an $\mathbb{R}^d$ vector field spatial position parameter, and an $\mathbb{R}^m$ vector field value parameter. Different discretization strategies can be taken for these two different components. The former is discussed in section 3.3.2 and the latter is discussed in section 3.3.1.

3.3.1 Distribution Discretization

The distribution discretization problem seeks to provide two functions: a discretization function and a reconstruction function. The discretization function, D, maps an input m-dimensional probability density function (PDF) to a finite number of real numbered values. The reconstruction function, D^{-1}, enables the reconstruction of PDFs from the values produced by the discretization function.

For example, a simple discretization function could be a mean combined with a variance. The associated reconstruction function would then be a normal distribution. This is not appropriate for many real cases, but it serves as a simple example.

In effect, the goal is to provide compact models of probability density functions that are appropriate for the underlying distributions. Many approaches exist for solving this problem [3]. The following are a few approaches that are appropriate for use with span histograms, and are of relatively low implementation complexity.

For cases where m is small (where there are few variables in the multivariate volume), histograms can be an effective tool for discretization. When m is greater than 1, this discretization takes the form of joint histograms. Due to the curse of dimensionality, joint histograms may not be effective in handling cases where m is large [3]. Gaussian mixture models or polynomial fits may be acceptable alternatives to histograms in cases where m is large.

In our end goal of evaluating queries in the form of Q in equation (1), the $\vec{t}$ value may actually take multiple values for a single query position. This is because most applications will be interested in the PDF for more than one value. Thus, any discretization model applied should consider this. Unless only very few values of $\vec{t}$ are

needed, it will generally make more sense to provide the user application the entire distribution model as the result of Q, rather than a single value.

When histograms are used, choices must be made on what binning strategy may be used. When two histograms are to be added or subtracted, the operation reduces to a simple vector addition or subtraction, when the bin bounds are the same for the operands. If they do not line up, error will be introduced during the addition because in the cases of partial bin overlap, it is not clear how to distribute the values between overlapping bins. This implies that the same histogram binning should be used for all histograms stored in the metadata. However, it does not apply any restrictions on what the specific binning strategy used should be, other than that it should be suitable for the entire dataset. This is a well-explored problem [31] [17], but there is still substantial room for future work in exploring it in the context of large-scale data.

For the purposes of exploring the applications in this paper, a histogram discretization is used and globally-uniform equal-width binning is assumed. However, for the purposes of the definition of span distributions, no assumptions are made about the discretization other than that they can be added and subtracted.

3.3.2 Spatial Discretization

Similarly to the distribution discretization problem, the spatial discretization problem seeks to provide two functions: a discretization function, and a reconstruction function. The discretization function, S, maps the discretized distribution at every point in the input spatial domain, as produced by the distribution discretization function (D), to a finite number of real values. The reconstruction function, S^{-1}, maps the resulting values from the discretization function back to the input domain.

For example, a simple discretization function would be nearest neighbor sampling onto a uniform grid. The associated reconstruction function for a spatial position would return the value at the nearest sample. Combining this spatial discretization function with a histogram distribution discretization function is the approach taken by integral histograms [27].

Span distributions are an alternative spatial discretization function, designed to be more working set-efficient by supporting multiresolution approximate reconstruction of the W field and taking advantage of spatial coherence between nearby regions.

3.4 Span Distributions

The volumes affecting each integral distribution tend to have considerable overlap, as shown in figure 2. This implies that their respective distributions will be similar. For example, consider the case in a 3D volume where an integral distribution is defined for the point $<1,1,1>$. If a neighboring integral distribution is defined for the point $<1,1,1.01>$, 99% of the contributing volume will overlap, placing an upper bound on the possible difference between the distributions.

Two key observations follow from this. First, a hierarchical spatial discretization can be used effectively for facilitating approximate integral distribution reconstruction. Secondly, the information entropy of the difference between neighboring integral distributions will tend to be considerably smaller than the information entropy of the individual integral distributions. Span distributions are designed to take advantage of both of these observations.

Span distributions are a spatial discretization strategy, mapping $X_i(\vec{s})$ to $Y_{k,i}$, taking the place of S in equation (2). In the case of $d=1$, where V has only one spatial dimension, the span distribution discretization function is defined as:

$$\begin{aligned} Y_{G(s),i} &= X_i(s) - X_i(s - G^{-1}_{(B_{G(s)})}) \\ B_k &= 2^{L_k} \\ L_k &= (\text{least significant nonzero bit index in } k) \end{aligned} \tag{9}$$

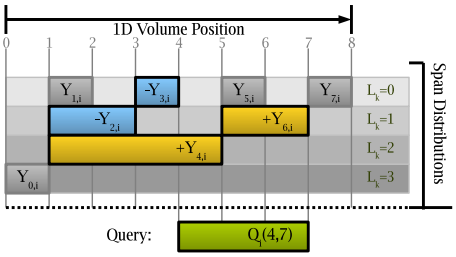


Figure 3: Distribution range queries are executed by evaluating the integral distirbution of each corner of the range using equation (10), then combining them using equation (8). In this example, the range query is evaluated using 4 span distributions, subtracting the span distributions ($Y_{2,i}$ and $Y_{3,i}$) that contribute to the $X_i(4)$ integral distribution, and adding the span distributions ($Y_{4,i}$ and $Y_{6,i}$) that contribute to the $X_i(7)$ integral distribution.

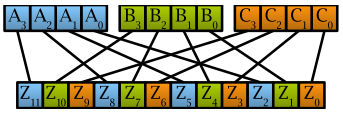


Figure 4: The Z-order space-filling curve maps a d-dimensional integer coordinate to a 1-dimensional integer coordinate. In this example, a 3D coordinate with 4 bits per component is mapped to a a single 1D coordinate with 12 bits.

where $G(s)$ and G_k^{-1} are nearest neighbor mappings from $\mathbb{R}$ to $\mathbb{Z}$ and $\mathbb{Z}$ to $\mathbb{R}$ respectively.

The inverse transform (S^{-1} in equation (3)) maps $Y_{k,i}$ to $X_i(\vec{s})$. In the case of $d=1$, where V has only one spatial dimension, the span distribution reconstruction function is defined as:

$$X_i(s) = \begin{cases} Y_{G(s),i} + X_i(G^{-1}_{B_{G(s)}}) & : \quad G(s) \neq 0 \\ 0 & : \quad G(s) = 0 \end{cases} \tag{10}$$

Intuitively, the forward transformation discretizes the spatial positions to a uniform grid, then stores a distribution for each nonzero bit in the discretized grid coordinate index. This decomposes the input integral distributions into one or more span distributions. The inverse transform performs the reverse of this, fetching one span distribution for each nonzero bit in the discretized grid coordinate index. Figure 3 shows an example of this being used for range queries.

Extending the above equations from one dimension to d dimensions only requires a modification to the $G(\vec{s})$ and G_k^{-1} nearest neighbor mappings. Specifically, the real-valued spatial positions are mapped to integer positions, component wise, on a uniform grid. Then, a single integer is produced from these coordinates' integer positions by using the Z-order space-filling curve [24]:

$$G(\vec{s}) = \left\lfloor Z(\text{diag}(\vec{N})\vec{s}) + \tfrac{1}{2} \right\rfloor \tag{11}$$

$$G_k^{-1} = \text{diag}\left(\tfrac{1}{\vec{N}}\right) Z^{-1}(k) \tag{12}$$

where Z is the Z-order space-filling curve and $\text{diag}(\vec{v})$ produces a diagonal matrix from vector $\vec{v}$. Figure 4 shows an example of a Z-order curve encoding of a 3 dimensional integer vector. Use of Z-order space filling curves for hierarchical representations has been applied before [25], due to their favorable storage locality properties and simplicity. However, they have not been used in the context of representing data structures similar in purpose or structure to integral distributions.

This section has provided a definition of span distributions, in terms of the transformations (discretization and reconstruction) be-

tween $Y_{k,i}$ and $X_i(\vec{s})$. The next section discusses how $Y_{k,i}$, the span distributions, are stored.

3.5 Storage of Span Distributions

The $Y_{k,i}$ field, produced by the transformation discussed in the previous section and shown in equation (2), is the metadata that is to be stored on disk, but it must be encoded first. Multiple considerations must be made to facilitate efficient storage and use.

Because the goal is to leverage the hierarchical representation of span distributions to facilitate approximate queries, it makes sense to store the elements for each level contiguously, rather than interleaving the elements from different levels. This is taken advantage of in section 4.1. Additionally, because user applications are likely to be interested in the PDF evaluated at ranges of values, rather than a single value, values that contribute to the same PDF discretization should be stored contiguously. Finally, working sets can be further reduced by applying entropy coding.

To construct the hierarchical storage model, $Y_{k,i}$ is separated into levels. The level to which a span distribution is assigned is L_k, from equation (9), which is simply the index of the rightmost nonzero bit in k. In the case of $k = 0$, the level index is $\log_2 N$ where N is the number of span distributions. For multidimensional indices, k is the Z-order index of the grid coordinate, for a uniform grid superimposed on the volume. The result of this is that the number of span distributions in each level decreases as the level index increases. Additionally, because the width of each span distribution is a function of the level number (as can be seen in equation (9)), the volume of the space contributing to a span distribution will tend to increase as the level index increases.

Each level is stored as a set of chunks, with each chunk storing a sequence of entropy coded span distributions. To further improve entropy coding performance, each span distribution within a chunk (other than the first span distribution) is stored differentially with respect to the previous span distribution in the chunk. Each chunk stores an entropy coding model and a set of entropy codes. Because the information entropy of each span distribution can vary per-level, the number of span distributions that are stored per chunk should also vary per-level. This is necessary to maintain a favorable ratio between entropy coding model sizes and entropy code array sizes.

In our experiments, we found that the size of levels, in terms of total information entropy, varies exponentially with respect to the level number. This can be seen in figure 5. Similarly, the number of span distributions per level also varies exponentially. Thus, because the total information entropy of a level is equal to the number of span distributions times the information entropy of each span distribution, the information entropy of the span distributions can also be modeled as an exponential.

If the level index is ℓ, then the total size for levels can be modeled as:

$$\mathscr{H}_\ell = \alpha_1 e^\ell + \alpha_0 \tag{13}$$

Additionally, the number of span distributions in a level can be modeled as:

$$\mathscr{N}_\ell = 2^{\log_2 N^d - \ell} \tag{14}$$

With this, the entropy per span distribution can then be modeled as:

$$\mathscr{M}_\ell = \frac{\mathscr{H}_\ell}{\mathscr{N}_\ell} = \beta_1 e^\ell + \beta_0 \tag{15}$$

Assuming that the size of the entropy model for a given chunk is constant, and is $\mathscr{F}$, and the ideal ratio between the entropy model between the size of the model and the size of the code sequence is γ, the ideal number of span distributions, $\mathscr{L}$, for a given level ℓ is:

$$\frac{\mathscr{F}\gamma}{\mathscr{M}_\ell} = \mathscr{L} \tag{16}$$

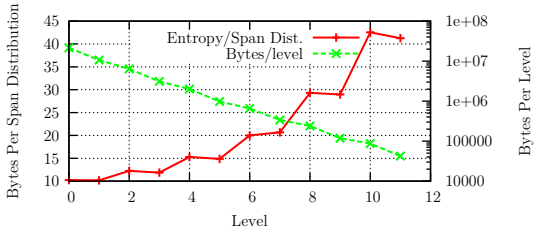


Figure 5: Both the size of the levels, and the number of span distributions in the levels, exponentially decreases as the level number increases. The ratio between the size of the span distributions and the number of span distributions enables modeling of the entropy per span distribution.

The optimal value for γ depends on the latency to throughput ratio of the storage devices being used. If latency is high, then γ should be high. Similarly, if latency is low, then γ should be low. At the extreme, if γ is too large, then the cost to perform a fetch of a span distribution can be excessively large. In practice, for solid-state drives with static Huffman coding, a γ value of around 30 was found to be reasonable.

Because L_k determines the level number each successively lower level number has higher spatial precision. If spatial precision can be sacrificed, then span distributions do not necessarily need to be loaded for all levels, nor do they need to be stored for all levels. In other words, span distributions can be selectively loaded to facilitate approximate queries.

3.6 Approximate Queries with Span Distributions

Figure 6: The relationship between the error bound and the stored size for varying numbers of levels skipped

Approximate queries can be performed simply by not storing (or not using) some of the lower numbered levels. This can be accomplished by modifying equation (10) to not include levels whose index is less than a threshold. For example, in figure 2, dropping the highest detail level would be equivalent to not storing the row of span distributions for $L_k = 0$.

Each span distribution has a corresponding region of space from which its distribution is drawn. This region is implied by equation (9). The corresponding volume of a span distribution is the intersection of the two contributing integral distributions in the equation (9) subtracted from the union of the same two contributing integral distributions.

For one dimensional volumes of size N, the mean volume of a corresponding region for a span distribution within a level ℓ is $N^{-1}2^\ell$. For d-dimensional N^d volumes, this generalizes to:

$$2^{\left(1-d+\frac{\ell}{d}\right)}(N+1)^{(d-1)}N^{-d} \tag{17}$$

Intuitively, this means that as the level number is increased, the mean volume increases exponentially. This implies that the the potential error that can be introduced by dropping a low numbered level will tend to be considerably lower than the potential error that can be introduced by dropping a high numbered level.

In addition to this, as discussed in the previous section, the total size of span distributions on disk increases exponentially, as a function of level number. Combining these two observations, we can see that low levels of the span distribution data contribute the least amount of potential error, yet cost the most amount of space on disk. This enhances the effectiveness of a technique to reduce query cost and metadata size by dropping low numbered levels.

3.7 Comparing Span Distributions

Span distributions provide a spatial discretization that enables approximate queries, and generalize support for distribution range queries to arbitrary distributions, rather than just histograms.

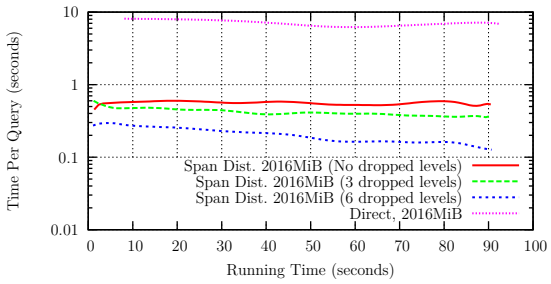


Figure 7: Out-of-core data, query time, randomly positioned and sized queries, 2016MiB source data

In supporting approximate queries, span distributions introduce a hierarchical component to the metadata structure, as discussed in the previous section. If the working set is considered for the time interval of a single query (rather than the time interval for an entire workflow), this will introduce an $\mathcal{O}(logN)$ factor for spatial discretizations with N samples. However, if the number of levels stored is held fixed, regardless of the data size, this reduces the working set to $\mathcal{O}(1)$. In either case, the method is considerably faster than $\mathcal{O}(N)$ methods. Figure 7 shows a typical result on a dataset several times larger than the size of the core memory available.

The following table summarizes different aspects of some alternative methods for evaluating distribution range queries:

	Span distrib.	Integ. hist.[27]	Hixels [29]
Working set	$\mathcal{O}(logN)$, $\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(N)$
Spatial discretization	hierarchical	uniform	uniform
Distribution discretization	general	histograms	histograms
Compression	entropy coding	raw	raw

The working set in this table refers to the working set required for a single query of random location and size in a volume, where N is the number of discrete elements stored and the number of dimensions is assumed to be constant. While the asymptotic complexity of the working set for Integral Histograms [27] is less than that of Span Distributions for a single query, the size of the stored metadata is considerably larger. Figure 8 exhibits this difference in metadata sizes. Larger metadata sizes will affect cache performance, which can be observed in figure 9.

The above considers the working set only in terms of the time intervals associated with individual queries. The next section discusses working sets in the context of the time intervals covering entire application workflows.

4 Working Sets in Applications

In the context of visualization workflows, working sets are having an increasing impact. Working set complexity for a time interval, by definition, also places a lower bound on the compute time during

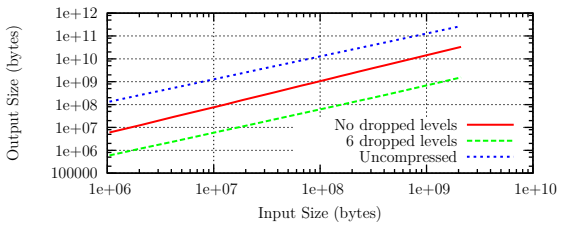


Figure 8: Comparison of metadata sizes from applying lossless compression with span distributions versus lossy compression with span distributions versus no compression with integral histograms

that workflow. In fact, in many cases, when the working set is out-of-core, the time due to storage operations will be much larger than the time spent on computation.

We concentrate on the case where the size of the entire dataset is considerably larger than the in-core memory limit of the system, but the working set fits in-core. However, we assume that the working set cannot be adequately predicted a priori. Thus, cache warming cannot be used to pre-load the data outside of the interactive portion of the workflow. In this situation, the application will tend to be throughput-bound in the interactive portion of the workflow. Because of this, the working set of the over the entire time interval of the interactive portion of the workflow can be used to identify the performance characteristics of the workflow.

We take three steps in looking at performing working set analysis for a given application. First, the application algorithm is characterized. Next, the application query patterns are identified. Then, knowing the query patterns, the working set is analyzed in the context of the workflow.

The following sections apply these steps for a couple different applications. The first application (§4.1) exemplifies a class of applications where distribution range queries can be used for error-bounded data reduction. The second application (§4.2) exemplifies a class of applications where distribution range queries can be used for interactive data summarization. The different classes have working set characteristics.

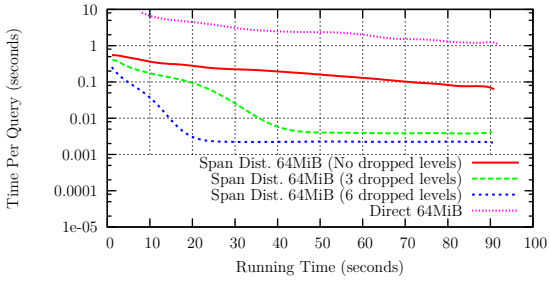


Figure 9: Out-of-core data, query time transient response, randomly positioned and sized queries, 64MiB source data

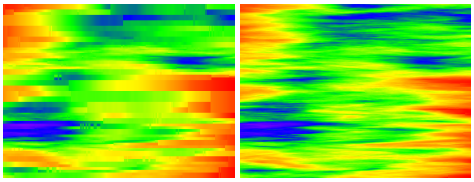
4.1 Application: Hövmoller diagrams

Hövmoller diagrams are used in meteorology to highlight wave phenomena [14]. They are 2D plots where one axis typically shows longitude or latitude and the other axis shows time. Each point in the plot shows the aggregation, or sum, of the remaining axes. Effectively, these diagrams are sum aggregation queries, aggregating sequences of samples along one axis into sums on a per element basis in the other axes. Similar aggregation queries have also been performed in higher dimensional visualization contexts [30]. We propose a method using histograms to estimate these diagrams subject to interactively-chosen error constraints.

Sum aggregation queries of this form can be evaluated using histograms. Suppose we pick a bounding box within the volume, across which we want to evaluate sums down one axis. In 3D this will produce a 2D image. To estimate the range of each of these sums we can take the histogram of the bounding box region. If the axis along which we want to sum values has n entries within the bounding box, then the sum of the top n events in the histogram is the upper bound of the sum for each entry on the other two axes. Similarly, the sum of the bottom n events in the histogram is the lower bound of the sum for each entry on the other axes. To compute the complete image, quadtree subdivision can be performed subject to a constraint on error.

The algorithm applied to produce this is effectively performing a breadth-first search, within the quadtree of the 2D projection for squares of the image space that can have their sums approximated with a single distribution range query. Because the octree subdivision is in image space, the maximum number of distributions that can be executed is $\mathcal{O}(M \log M)$ where M is the number of image space pixels. Note that the number of samples in the volume, N, is not present.

Span distributions and integral histograms can both yield $\mathcal{O}(M)$ performance in this case. This is substantially better than the $\mathcal{O}(NM)$ performance that will result from the use of uniform hixels or directly computing the histograms off the volume data. However, there is a substantial difference between the actual access patterns of span distributions and integral histograms. Because span distributions are hierarchical, and this algorithm is a breadth-first search, the accesses to span distributions will be concentrated into more localized regions of the metadata, potentially yielding improved utilization of read-ahead. Figure 9 shows this behavior.



(a) Tolerance of ± 10 (b) Tolerance of ± 5

Figure 10: Approximate sum aggregation of 3D volumes for Hövmoller diagrams as discussed in §4.1. The horizontal axis is longitude and the vertical axis is time. The tolerance provides a bound on how far the approximate sums may be from the true sums. The dataset is from a simulation produced by the Pacific Northwest National Laboratory to examine the Madden-Julian Oscillation [12].

4.2 Application: Transfer function design

Performing transfer function design on time-varying data has long been a challenge, with the dynamic range of values of interest being unclear for the entire time series if they are not known a priori [32]. For static datasets, histograms have been used in the context of interactive workflows to generate transfer functions [23] [28] [20] [5]. Using span distributions enables the use of interactively computed transfer functions using regions covering the entire time domain.

Figure 11 exhibits an application where this is applied on the 4D NCAR dataset [2], the result of a fluid dynamics simulation. In the left pane of the application window an aggregation of the 4D volume onto a 2D plane, computed with the same method discussed in §4.1, is applied. The user can drag and resize the query region box in the left pane to select a 4D range of interest for the transfer function. The cumulative distribution function of the histogram is used as a lookup table to warp the color portion of the transfer function such that contrast is maximized for values that have a high frequency of occurrence in the region of interest selected. The opacity of the transfer function is determined by the values of the histogram bins. Figure 11 shows different regions of interest selected for the transfer function with the same timestep.

Queries in the left pane are performed, in real-time, on the entire time series. The volume rendered in the right pane is for a single time step. This enables the user to interactively select a transfer function that is generated in a way consistent with the entire time series, without needing to have the entire dataset resident in memory.

Additionally, with fast interactive queries, techniques that depend on cursors for histogram determination can be supported on large data. For example, Martin et al. [23] uses the interactive manipulation of cursors on slice planes to incrementally construct transfer functions, which requires fast distribution range queries. For multidimensional transfer function construction, the technique introduced by Kniss et al. [16] could be extended to use distribution range queries of subvolumes, in addition to the entire volume.

In this application, the algorithm itself performs single range queries for histograms in a 4D volume, then uses the histograms to compute the transfer function. Every time the user moves the cursor for a region of interest, a new query is performed. If L unique queries are perormed, then using integral histograms will require a working set of $\mathcal{O}(L)$. Span distributions, however, can take advantage of other properties of this workflow.

Generally, the queries will have considerable overlap and be performed over relatively large subvolumes of the input 4D volume. Because of this, approximate span distributions can be used effectively and the number of levels used can be chosen to be appropriate for the granularity of the queries that the user wishes to perform. This results in the span distribution method also requiring $\mathcal{O}(L)$, but with a considerably smaller actual working set size due to the approximate queries.

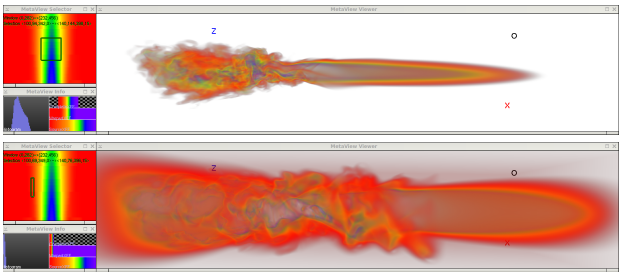


Figure 11: Interactive transfer function design for large-scale time-varying volume data, using interactive 4D distribution range queries, as discussed in §4.2. The user moves a region of interest in the left pane on a projection of the volume. The distribution of the region of interest is then used to generate transfer functions in the right pane, using the same technique as Martin et al.[23]

5 Future Work and Conclusion

Working set reduction, in the context of interactive workflows, will continue to be of great interest for many applications in visual analysis. In this work we proposed a general framework within which this problem can be approached. A transformation is performed in the preprocessing phase to facilitate distribution range queries, then the application is adapted to utilize approximation algorithms that can make use of these range queries. The general pipeline is broken into two transformations and their inverses: a distribution discretization, and a spatial discretization. Both of these transformations are used to facilitate an effective representation of integral

[2] this is the same dataset applied by Akiba et al. [1]

distributions, a continuous multivariate generalization of integral histograms.

Building on this framework, we focus on a spatial discretization strategy: span distributions. Span distributions facilitate efficient, potentially-approximate, distribution range queries through a hierarchical decomposition of integral distributions. We show that this approach can be used to construct scalable algorithms for analysis applications – algorithms whose time complexity varies in terms of the analysis result size, not the input data size.

Future work could include extending this approach to more applications. For example, the same range queries applied for transfer function design and Hövmoller diagrams could also be used to enable new volume rendering algorithms whose working sets depend primarily on the target image resolution rather than the volume size. Other applications could include fuzzy isosurfaces, classification, and feature detection. With the proposed framework, a single pass of preprocessing can produce metadata that enables algorithms with scalable working set characteristics. For a range of applications, the working set complexity can be changed to depend primarily on the result size, rather than the input data size.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant IIS-1017635, US Department of Energy DOE-SC0005036, Battelle Contract No. 137365, and Department of Energy SciDAC grant DE-FC02-06ER25779, program manager Lucy Nowell.

REFERENCES

[1] H. Akiba, K.-L. Ma, and J. Clyne. End-to-end data reduction and hardware accelerated rendering techniques for visualizing time-varying non-uniform grid volume data. *International Workshop on Volume Graphics*, 0:31–225, 2005.

[2] F. Bengtsson and J. Chen. Space-efficient range-sum queries in olap. In *In Yahiko Kambayashi, Mukesh Mohania, and Wolfram W, editors, Data Warehousing and Knowledge Discovery: 6th International Conference DaWaK, volume 3181 of Lecture Notes in Computer Science*, pages 87–96. Springer, 2004.

[3] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] S.-J. Chun, C.-W. Chung, and S.-L. Lee. Space-efficient cubes for olap range-sum queries. *Decis. Support Syst.*, 37(1):83–102, Apr. 2004.

[5] C. D. Correa and K.-L. Ma. Visibility-driven transfer functions. In *Proceedings of the 2009 IEEE Pacific Visualization Symposium*, PACIFICVIS '09, pages 177–184, Washington, DC, USA, 2009. IEEE Computer Society.

[6] F. C. Crow. Summed-area tables for texture mapping. *SIGGRAPH Comput. Graph.*, 18(3):207–212, Jan. 1984.

[7] P. J. Denning and S. C. Schwartz. Properties of the working-set model. *Commun. ACM*, 15(3):191–198, Mar. 1972.

[8] F. Furfaro, G. M. Mazzeo, D. Saccà, and C. Sirangelo. Compressed hierarchical binary histograms for summarizing multidimensional data. *Knowl. Inf. Syst.*, 15(3):335–380, May 2008.

[9] S. Geffner, D. Agrawal, A. El Abbadi, and T. Smith. Relative prefix sums: An efficient approach for querying dynamic olap data cubes. Technical report, Santa Barbara, CA, USA, 1999.

[10] S. Guha. Space efficiency in synopsis construction algorithms. In *Proceedings of the 31st international conference on Very large data bases*, VLDB '05, pages 409–420. VLDB Endowment, 2005.

[11] S. Guha, N. Koudas, and D. Srivastava. Fast algorithms for hierarchical range histogram construction. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '02, pages 180–187, New York, NY, USA, 2002. ACM.

[12] S. Hagos and L. R. Leung. Moist Thermodynamics of the Madden-Julian Oscillation in a Cloud-Resolving Simulation. *Journal of Climate*, 24:5571–5583, Nov. 2011.

[13] W.-C. Hou, C. Luo, Z. Jiang, F. Yan, and Q. Zhu. Approximate range-sum queries over data cubes using cosine transform. In *Proceedings of the 19th international conference on Database and Expert Systems Applications*, DEXA '08, pages 376–389, Berlin, Heidelberg, 2008. Springer-Verlag.

[14] E. Hovmöller. The trough-and-ridge diagram. *Tellus*, 1(2):62–66, 1949.

[15] P. Karras. Optimality and scalability in lattice histogram construction. *Proc. VLDB Endow.*, 2(1):670–681, Aug. 2009.

[16] J. Kniss, G. Kindlmann, and C. Hansen. Multidimensional transfer functions for interactive volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 8(3):270 – 285, jul-sep 2002.

[17] K. H. Knuth. Optimal Data-Based Binning for Histograms. *ArXiv Physics e-prints*, May 2006.

[18] N. Koudas, S. Muthukrishnan, and D. Srivastava. Optimal histograms for hierarchical range queries (extended abstract). In *Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '00, pages 196–204, New York, NY, USA, 2000. ACM.

[19] W. Liang, H. Wang, and M. E. Orlowska. Range queries in dynamic olap data cubes. *Data Knowl. Eng.*, 34(1):21–38, July 2000.

[20] C. Lundstrom, P. Ljung, and A. Ynnerman. Local histograms for design of transfer functions in direct volume rendering. *Visualization and Computer Graphics, IEEE Transactions on*, 12(6):1570 –1579, nov.-dec. 2006.

[21] T. Malzbender. Fourier volume rendering. *ACM Trans. Graph.*, 12(3):233–250, July 1993.

[22] S. Martin and H.-W. Shen. Histogram spectra for multivariate time-varying volume LOD selection. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 39 –46, Oct. 2011.

[23] S. Martin and H.-W. Shen. Interactive transfer function design on large multiresolution volumes. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, Oct. 2012.

[24] Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical Report Ottawa, Ontario, Canada, 1966.

[25] V. Pascucci and R. Frank. Global static indexing for real-time exploration of very large regular grids. In *Supercomputing, ACM/IEEE 2001 Conference*, page 45, nov. 2001.

[26] V. Poosala and V. Ganti. Fast approximate answers to aggregate queries on a data cube. In *Proceedings of the 11th International Conference on Scientific and Statistical Database Management*, SSDBM '99, pages 24–33, Washington, DC, USA, 1999. IEEE Computer Society.

[27] F. Porikli. Integral histogram: a fast way to extract histograms in cartesian spaces. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 829 – 836 vol. 1, june 2005.

[28] T. Ropinski, J.-S. Praßni, F. Steinicke, and K. H. Hinrichs. Stroke-based transfer function design. In *IEEE/EG International Symposium on Volume and Point-Based Graphics*, pages 41–48. IEEE, 2008.

[29] D. Thompson, J. Levine, J. Bennett, P.-T. Bremer, A. Gyulassy, V. Pascucci, and P. Pebay. Analysis of large-scale scalar data using hixels. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 23 –30, oct. 2011.

[30] J. J. van Wijk and R. van Liere. Hyperslice: visualization of scalar functions of many variables. In *Proceedings of the 4th conference on Visualization '93*, VIS '93, pages 119–125, Washington, DC, USA, 1993. IEEE Computer Society.

[31] M. P. Wand. Data-based choice of histogram bin width. *The American Statistician*, 51:59–64, 1996.

[32] J. Woodring and H.-W. Shen. Semi-automatic time-series transfer functions via temporal clustering and sequencing. *Computer Graphics Forum*, 28(3):791–198, June 2009.

[33] Y.-L. Wu, D. Agrawal, and A. El Abbadi. Using wavelet decomposition to support progressive and approximate range-sum queries over data cubes. In *Proceedings of the ninth international conference on Information and knowledge management*, CIKM '00, pages 414–421, New York, NY, USA, 2000. ACM.