Eurographics/ IEEE-VGTC Symposium on Visualization (2006)
Thomas Ertl, Ken Joy, and Beatriz Santos (Editors)

Improving the Quality of Multi-resolution Volume Rendering

H. Younesy [1] and T. Möller [1] and H. Carr [2]

[1]Graphics, Usability and Visualization Lab (GrUVi), Simon Fraser University, Vancouver, Canada
[2]University College, Dublin, Ireland.

Abstract

We propose a novel method to improve the quality of multi-resolution visualizations. We reduce aliasing artifacts by approximating the data distribution with a Gaussian basis function at each level of detail for more accurate rendering at coarser levels of detail. We then show an efficient implementation of our novel Gaussian based approximation scheme and show its superiority using numerical tests and compelling renderings.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computing Methodologies]: Computer Graphics - Methodology and Techniques E.1 [Data]: Data Structures

1. Introduction

Visualizing scientific and medical volumetric data continues to require processing ever larger data sets. Many algorithms have been proposed for optimizing performance, especially for datasets too large to fit in main memory. Such datasets are commonly visualized with hierarchical rendering algorithms based on multi-resolution representation of the data, trading off image quality against rendering speed.

Although image quality is implicitly balanced against rendering speed, few authors have addressed the question of multi-resolution image quality, i.e. the quality of the images rendered at coarser levels of resolution. In this paper, we address this question by using statistical information to decrease aliasing artifacts in direct volume rendering.

Conventional multiresolution algorithms downsample voxel opacities with the mean value of all high resolution voxels that contribute to each low resolution voxel. Especially for transfer functions with relatively narrow domains, this causes features visible at high resolution to disappear, blur or deform in low resolution images. Our first contribution is therefore to demonstrate how these defects can be reduced by storing statistical information such as histograms in the multi-resolution data structure.

However, histograms are sufficiently expensive that they negate most of the advantage of the multi-resolution representation, so our second contribution is to demonstrate that a simple Gaussian distribution, stored as mean and standard deviation, can significantly improve image quality at little or no additional cost in memory or rendering time. We do so by describing a novel approach that efficiently combines Gaussian distributions with transfer functions to generate output colors and opacities.

The remainder of this paper is organized as follows: After a review of related work in Section 2 we present the details of our method and the Gaussian basis transfer function in Section 3. Experimental results are discussed in Section 4, followed by conclusions and future work in Section 5.

2. Related Work

Since it is not always possible to interactively process data in the original sampling resolution, level of detail and multi-resolution techniques have been proposed by many researchers to balance between architectural constraints (e.g. performance and memory) and fidelity. Multi-resolution techniques have been employed in a variety of visual data representation approaches including geometric rendering [WG92, CDL*96, LWC*02] and volumetric data visualization [LHJ99] especially for out-of-core applications [SCM99].

Multi-resolution volume rendering techniques generally use spatial hierarchies to represent different levels of detail. Often the goal is to maintain a specific frame-rate [LS02] at the expense of image quality. The quality of the image can then be updated when user interaction stops using techniques such as progressive refinement [LH91, PPL*99, RYL*96].

Most hierarchical multi-resolution schemes use the mean

values of the underlying function values in order to construct the representations in different levels of detail. Notable exceptions are wavelet-based multi-resolution schemes [GLDH97, HVU98], which use better filtering methods to improve the quality of lower levels of the multi-resolution hierarchy. In particular, Guthe et al. [GWGS02, GW04] use a block based wavelet compression and employ graphics hardware to handle large datasets at interactive speed. Kraus et al. [KE01] use a topology guided downsampling to preserve topology of a scalar volume field in coarse levels of detail.

Some of the proposed multi-resolution hierarchies for volume rendering also address the underlying transfer function being used during the rendering. Ljung et al. [LLYM04] use a level-of-detail scheme, selecting the appropriate level of detail during data decompression using information from the transfer function being applied. Wittenbrink et al. [WMG98] have shown that interpolation of the underlying function *before* the application of a transfer function is of utmost importance in order to assure the best quality. Röttger et al. [RKE00, RE02] have since shown that the application of a transfer function to the underlying scalar data can create arbitrarily high frequencies, making it difficult to render the data properly. To address this, they proposed pre-integrated volume rendering to minimize the impact of the modulation of the underlying signal by the transfer function.

Other related work deals with the quality and proper sampling in the volume rendering pipeline. Mueller et al. [MMI*98] combine elliptical Gaussian reconstruction kernels with a Gaussian low-pass filter to reduce the aliasing artifacts in volume splatting. Zwicker et al. [ZPvBG02] extends this idea to point-based data. For a high-quality texture-based volume rendering, Engel et al. [EKE01] avoid additional slices by integrating non-linear transfer functions in a pre-processing step known as pre-integrated volume rendering.

However, none of these approaches has dealt explicitly with the proper use of transfer functions nor exploited the underlying distribution of the function values to improve image quality. This paper therefore proposes a novel method that improves the quality of multi-resolution rendering by approximating the distribution of function values at coarse levels of detail.

3. Rendering

Given a desired level of performance, hierarchical rendering uses criteria such as viewing distance [WWHW97], projection area [LS02] and gaze distance [LW90] to identify the level of detail at which the data must be rendered. At a given level of detail, one should ideally produce the highest possible image quality for a given performance cost. To understand how to do so, we consider the theoretical implications of multi-resolution downsampling, then develop a method that significantly reduces image artifacts.

3.1. Accurate Multi-resolution Transfer Functions

In general, hierarchical rendering decreases rendering time by substituting a small data set downsampled from the full data set. This tends to result in aliasing problems such as the "jaggies" and discontinuities visible in Figure 2, caused by the loss of detail information during construction of the multi-resolution representation. In particular, information is lost because the data in a subvolume is represented by the mean of the samples in the subvolume. This mean value represents the data values well if they form smooth regions with little variance, but data sets often display their most interesting behaviour near sharp gradients for which the mean alone is not a good summary of a subvolume. This causes drastic problems in the resulting images because the transfer function often does not have a smooth transition between colors and opacities for different values. More sophisticated summaries of the subvolume can, however, be substituted for the mean, and this idea forms the basis of our approach.

As an example, consider a subvolume with values $\{1,1,1,1,5,5,5,5\}$ and mean $\mu = 3$ and suppose $V(x)$ is a transfer function which maps real numbers to color values. Unless $V(3)$, the color specified in the transfer function for 3, is near to the color produced by blending $V(1)$ and $V(5)$, the summary information will misrepresent the data in the final visual image.

For simplicity, we develop notation in one dimension: higher dimensions are essentially identical, but with denser notation (i.e. integration along three dimensions). Assume that the scalar field to be visualized is $f(x)$, and that the sample points x_i are evenly spaced: i.e. that $f(x_i) = f(iT)$ for some sampling distance T. For convenience, we use f_i to refer to $f(x_i)$.

Typically, the transfer function V is a vector-valued function mapping input data values to output colors and opacity. We write this transfer function as $V : \mathbb{R} \rightarrow (R,G,B,\alpha)$, where R,G,B,α are respectively red, green, blue and opacity, and state the rendering task in terms of visualizing $V(f(x))$, of which we only know the values $V(f_i)$. A proper coarse visualization of this discrete dataset would involve smoothing $V(f(x))$ with a low-pass filter in order to avoid aliasing, then sub-sampling. For level n in the multi-resolution hierarchy, this can be written as:

$$V_{n+1}(f(y)) = \int_{-\infty}^{\infty} V_n(f(y-x))w(x)dx \qquad (1)$$

where

$$f(x) = \sum_{k=-\infty}^{\infty} f(k)h(x-k) \qquad (2)$$

where w is the (continuous) low-pass filter used for computing the $(n+1)$st level of the multi-resolution pyramid and h is a (continuous) interpolation filter used to reconstruct the underlying function f from its discrete representation. The importance of Equation 2 in volume rendering has been well documented by Wittenbrink et al. [WMG98].

The formulation in Equation 1 is not particularly useful, as it requires recomputing the multi-resolution pyramid after each change of the transfer function V. A more useful formulation of Equation 1 derives from the sifting property of the Dirac delta function δ [Bra99]:

$$V_{n+1}(f(y)) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} V_n(s)\delta(s-f(y-x))w(x)dsdx \quad (3)$$

$$= \int_{-\infty}^{\infty} V_n(s)\left(\int_{-\infty}^{\infty}\delta(s-f(x))w(y-x)dx\right)ds$$

$$= \int_{-\infty}^{\infty} V_n(s)\left(\int_{f^{-1}(s)} w(y-x)dx\right)ds \quad (4)$$

, where $\int_{f^{-1}(s)}$ is the integral over the inverse image $f^{-1}(s)$ of the isovalue s.

If w is a box filter with support of size N, then $\int_{f^{-1}(s)} w(y-x)dx$ is equivalent to $\frac{1}{2N}\int_{f^{-1}(s)|[y-N,y+N]} dx$, i.e. the restriction of the integral to the support of the filter.

In the discrete case with a box filter, this distribution is merely the histogram of a local neighborhood of $f(y)$ weighted by $1/2N$: the histogram therefore also has support of size $2N$.

Regardless of the low-pass filter chosen, $\int_{f^{-1}(s)} w(y-x)dx$ is the weighted distribution of f in the support of the filter w, and the discrete version is the weighted histogram of f in the support of w. Thus, for any low-pass filter w, we can guarantee accuracy of low-resolution images within the limits of reconstruction by storing $H^w(s,y) = \int_{f^{-1}(s)} w(y-x)dx$ for each location y, where H^w is the *weighted* histogram centered at y. Note that all our histograms are local, i.e. they collect statistics about the underlying function f in the neighborhood of y. If we were to store these weighted histograms at each multi-resolution node, we would then compute the value at this node $V_{n+1}(f(y))$ in the following way:

$$V_{n+1}(f(y)) = \int_{-\infty}^{\infty} V_n(s)H^w(s,y)ds \quad (5)$$

This allows us to represent our function in a multi-resolution pyramid with proper anti-aliasing and without *a priori* knowledge of the transfer function. For multi-dimensional transfer functions we would use a multi-dimensional histogram, which can be derived using a multi-dimensional delta function in Equation 1.

3.2. Discretizing the Local Histograms

While the above equations were derived on the assumption of continuity, we normally work with discrete representations and discrete histograms. We therefore need to approximate $f(s)$ at a resolution dependent on the transfer function to assure that $V(f(s))$ is sampled above the Nyquist frequency. This is still an open problem: no algorithm is known to determine such a sampling frequency accuractly. Only an approxiamte upper bound has been determined [Kra03]. We leave this problem open, and work with the same sampling

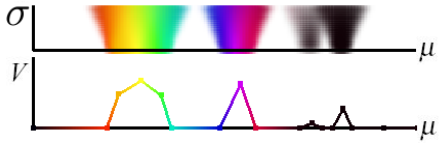


Figure 1: *Bottom: a transfer function with the opacities as the vertical axis (V). Top: the computed Gaussian basis transfer function; The horizontal and vertical axes are respectively mean (μ) and standard deviation (σ) and each point has been drawn with the color and opacity computed for the related μ and σ.*

as for the underlying scalar field f_i. Given this, and assuming that we have sampled 8-bit values ranging from 0 to 255, we can rewrite Equation 5 as follows:

$$V_{n+1}(f_i) = \sum_{j=0}^{255} V_n(j)H^w(j,i) \quad (6)$$

with

$$H^w(j,i) = \sum_{k=-N}^{N} \delta[j-f_{i-k}]w_k \quad (7)$$

where N is the filter support of w and $\delta[\,]$ is the discrete form of the delta impulse function. As noted above, the second term is simply the histogram of f weighted by the filter w.

It is important to point out that we need to low-pass filter the signal f_i for supersampling the function $V(f)$ [WMG98]. However, for sub-sampling $V(f)$, as is common in multi-resolution analysis, we need to low-pass filter V.

3.3. Gaussian Histogram Approximation

In practice, storing histograms for each block in the hierarchy requires a lot of storage and computation, thus slowing down rendering and negating much of the benefit from more accurate representation. To balance these competing demands, we summarize the subvolume, not with a weighted histogram, but with a compact approximation of the weighted histogram function. In our particular implementation we use a Gaussian distribution as a first approximation.

We therefore store the mean (μ) and standard deviation (σ) for each voxel at coarse resolutions. To find the color and opacity of a voxel with values (μ,σ), we then perform the following integration:

$$V_{n+1}(f_i) = \int_{-\infty}^{+\infty}\left(\frac{1}{\sigma_i\sqrt{2\pi}}e^{-(s-\mu_i)^2/(2\sigma_i^2)}\right)V_n(s)ds \quad (8)$$

Computing this integral explicitly for each voxel is expensive. But since V generally remains constant during rendering, we can treat the integral as a function of μ and σ and store it in a two dimensional transfer function lookup table

whose indices are the mean isovalue and the standard deviation. Each time the user changes the transfer function, we update the lookup table, which takes a few milliseconds. Figure 1 shows a sample transfer function (bottom) and the computed transfer function based on the Gaussian basis function (top). For the top transfer function, points have been drawn with the color and opacity computed for the related μ and σ.

We note that the Gaussian method may fail to approximate histogram distributions properly if they have more than one major maximum (peak). This typically happens at the border between a sampled phenomenon and empty space in the volume. One solution for this case would be to use more than one Gaussian function for approximating the distributions when the error of a single Gaussian approximation is not negligible, but we restrict ourselves to a single distribution for efficiency. Although this paper does not address this approach, we use a straight forward method to solve the problem of the empty regions, by not considering the voxels in empty spaces (voxels that are always set to have zero opacity) in calculating μ and σ.

4. Implementation and Results

Our goal in this work was to provide higher-quality volume renderings at coarse resolutions of a multi-resolution dataset. We identified five principal factors affecting the image quality: the type of data visualized, the simplification level of the data, the downsampling filter chosen during simplification, the statistical information retained at various levels of downsampling, and the type of transfer function chosen.

The data sets we chose as representative of different domains were a synthetic dataset (sphere), a medical dataset (Visible Human Male), and a fluid dynamics simulation dataset (Richtmyer-Meshkov). The synthetic dataset is a spherical distance function sampled at a resolution of 256^3, while the medical dataset is the visible human male head CT dataset with a resolution of $512 \times 512 \times 512$, and the simulation dataset is one frame (frame 250) of the Richtmyer-Meshkov instability dataset from Lawrence Livermore National Laboratory with a resolution of $2048 \times 2048 \times 1920$.

Although we wanted to use the full resolution of these datasets, the memory requirements of storing a histogram per each voxel in the coarse resolutions, precluded full comparisons for data sets larger than 64^3. Moreover, by choosing the coarsest levels of simplification, the characteristic visual artifacts were more visible. We therefore chose the four levels of simplification between 64^3 and 8^3.

For statistical information, we rendered our images with full histogram in each voxel (the best statistical information available), with the Gaussian distribution proposed in this paper, and with the conventional mean value. We chose three different filters (box, tent and cubic) to compare their effects on the output. Finally, we chose two transfer functions that oscillated between red, green and blue, one at high frequency, the other at low frequency, to emphasize the effects of sample distribution in the subsampled data.

Figures 2 - 4 show some of the images that we generated, chosen to illustrate the effects of the various choices. Figure 2 shows the effects of the Gaussian and mean distributions on the visible male head data set at full resolution (i.e. 256^3). Images using the histogram were not computed due to memory limitations, and a typical transfer function was chosen for displaying bone and other features.

As we can see from Figure 2, the Gaussian distribution does a better job of preserving features such as the diagonal line across the skull and the locations and shape of the eye sockets and mouth. And, at the coarsest level of resolution, the mean distribution generates curious colour artifacts due to ramps in the transfer function, while the Gaussian distribution does not. This disparity between Gaussian distribution and mean value was also visible across all resolutions, filters, transfer functions and datasets. We have omitted displaying all of our images, for clarity and to conserve space.

Figure 3 shows the effects of using different downsampling filters to reduce the Richtmyer-Meshkov data set from 64^3 down to 16^3 using a high frequency transfer function which oscillates from red to green to blue. We note that the histogram distribution gives a result that is quite close to the 64^3 resolution, no matter what filter is used, although blurring is apparent. Using mean distribution results in the colours that are sometimes unrelated to any of the samples in the block, leading to the artifacts shown, while the Gaussian distribution does a better job of approximating the correct integral values.

Similarly, Figure 4 shows the effects of different transfer functions at the two lowest resolution levels in the spherical data set. For a high-frequency transfer function, the correct image is of thin concentric shells of red, green and blue. A correct image will involve a fairly uniform mixture of red, green and blue throughout the volume: i.e. the grey tones visible in the histogram distribution images. However, for the mean distribution, the effect of the multi-resolution representation is to sample a single opacity for each sub-block, resulting in the color artifacts visible in this figure. Again, the Gaussian distribution, while not eliminating these artifacts entirely, is much more satisfactory.

In comparison, the right two columns of Figure 4 depict the low-frequency transfer function and the obtained images show a set of thick concentric spheres of slowly varying color. Even at coarse resolutions, we expect that these spheres will be distinguishable, and from the figure we see that this is indeed the case. In this case, the artifacts generated by the mean distribution are less prominent, but still visible, while the Gaussian distribution is nearly as good as the histogram distribution.

Resolution	Histogram	Gaussian/ Mean (Unpacked)	Mean (Packed)
256^3	-	77.2*MB*	58.5*MB*
128^3	3.6*GB*	9.7*MB*	7.3*MB*
64^3	460*MB*	1.3*MB*	920*KB*

Table 1: *Storage requirement for each distribution.*

In addition to visual comparison, we evaluated our results analytically by computing the root mean squared error between the approximated distribution of f for each voxel and the histogram distribution for the same voxel. This measures the error induced in f by our approximation before applying the transfer function. Although we plotted these errors for all of the images we generated, the plots consistently showed that Gaussian distributions had consistently smaller errors, and Figure 5(a) may be taken as representative.

As Figure 5(a) shows, the error of the Gaussian distribution was consistently less than the error of the mean approximation at each level of resolution, with the error diminishing under the Gaussian distribution as the resolution was coarsened further. This occurs because the downsampled voxels represent progressively more of the original data, with a Gaussian distribution becoming a better fit to the data as more samples accumulate. Although different low-pass filters affect the visual quality at coarser levels of detail (cubic is better than linear, and linear better than box), it does not have a major effect on the approximation errors. This is because the filters do not necessarily cause the histograms to become closer to Gaussian distributions.

We also computed root mean square error between the images produced with histogram distribution and images produced with Gaussian and mean distributions. This gives a measure of image error after applying transfer functions and filters. We computed the RGBA color of each voxel using the Gaussian approximation and mean approximation and compared it to the color when using the full histogram distribution for that voxel, averaging the root mean squared errors of red, green, blue and alpha components. A perceptual color space such as *CIE Lab* might be a better basis for an insightful comparison. Figure 5(b) and Figure 5(c) respectively show the error in rendered images for different filters and for different transfer functions.

Again, in these plots similar to the rendered figures, the average error of rendering is consistently less for the Gaussian distribution than the mean distribution. It is interesting to note that the actual downsampling filter is less significant than the statistical information chosen to represent the function.

4.1. Memory and storage overhead

We expected that encoding the standard deviation values for each voxel would add around 33% (1-byte σ added to 1-byte μ + 2-bytes gradient) overhead in terms of memory and processing power. Surprisingly there was no significant memory or processing overhead, presumably due to compiler optimization of data structures. Since voxel values (1 byte mean) and normals (2 bytes) are word aligned to 4 bytes by the compiler, adding the standard deviation (1 byte) adds no memory overhead in practice. During volume rendering, voxel color was determined by a look up to a table indexed by discretized μ and σ. The look up table is computed once at a negligible cost each time user changes the transfer function. The result of this was that we did not notice any slowdown due to our algorithm. We did consider not packing our data structures to achieve a storage reduction of 20%, but found that the additional overhead for runtime word alignment offsetted this, and reverted to an unpacked format. Table 1 shows the storage requirement to keep the multi-resolution information for each distribution.

5. Conclusion and future work

In this paper we have developed a mathematical framework for improving multi-resolution image quality based on transfer functions applied to scalar data. Using a Gaussian basis to approximate the data distribution at each level of detail, rather than the commonly used mean approximation, we are able to composite color more accurately and reduce aliasing artifacts. Our methods allow any low-pass filter to be applied to build the multi-resolution pyramid without a-priori knowledge of the transfer function.

In future we would like to extend our framework to visualize multi-modal and time-varying datasets, and consider the effect of an adaptive approximation scheme with different function bases for better approximation at different levels of detail of different datasets. We would expect to apply the work of Drew et al. [DWL98] with the Singular Value Decomposition of a wavelet-compressed histogram. We are also investigating possible ways to include gradients and other derived information in our current transfer function scheme.

References

[Bra99] BRACEWELL R.: *The Fourier Transform and Its Applications*, 3rd ed. McGraw-Hill, 1999. 3

[CDL*96] CHAMBERLAIN B., DEROSE T., LISCHINSKI D., SALESIN D., SNYDER J.: Fast rendering of complex environments using a spatial hierarchy. In *GI '96: Proceedings of the Conference on Graphics Interface '96* (1996), pp. 132–141. 1

[DWL98] DREW M. S., WEI J., LI Z.-N.: Illumination–invariant color object recognition via compressed chro-

maticity histograms of color–channel–normalized images. In *ICCV98* (1998), IEEE, pp. 533–540. 5

[EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), pp. 9–16. 2

[GLDH97] GROSS M. H., LIPPERT L., DITTRICH R., HÄRING S.: Two methods for wavelet-based volume rendering. *Computers & Graphics 21*, 2 (1997), 237–252. 2

[GW04] GUTHE S., WAND M.: Advanced techniques for high-quality multi-resolution volume rendering. *Computers and Graphics 28*, 1 (2004). 2

[GWGS02] GUTHE S., WAND M., GONSER J., STRASSER W.: Interactive rendering of large volume data sets. In *In Proceedings of IEEE Visualization 2002* (2002), pp. 53–59. 2

[HVU98] HORBELT S., VETTERLI M., UNSER M.: High-quality wavelet splatting for volume rendering. In *Wavelet Applications Workshop* (Monte Verità TI, Switzerland, September 28-October 2, 1998). 2

[KE01] KRAUS M., ERTL T.: Topology-guided downsampling. In *Proceedings of International Workshop on Volume Graphics '01* (2001), pp. 139–147. 2

[Kra03] KRAUS M.: *Direct Volume Visualization of Geometrically Unpleasant Meshes*. PhD thesis, University of Stuttgart, 2003. 3

[LH91] LAUR D., HANRAHAN P.: Hierarchical splatting: a progressive refinement algorithm for volume rendering. In *SIGGRAPH '91: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques* (1991), pp. 285–288. 1

[LHJ99] LAMAR E., HAMANN B., JOY K. I.: Multiresolution techniques for interactive texture-based volume visualization. In *Proceedings of the Conference on Visualization '99* (1999), pp. 355–361. 1

[LLYM04] LJUNG P., LUNDSTRÖM C., YNNERMAN A., MUSETH K.: Transfer function based adaptive decompression for volume rendering of large medical data sets. In *Proceedings of IEEE Symposium on Visualization and Graphics 2004* (2004), pp. 25–32. 2

[LS02] LI X., SHEN H.-W.: Time-critical multiresolution volume rendering using 3D texture mapping hardware. In *Proceedings of the 2002 IEEE Symposium on Volume Visualization and Graphics* (Piscataway, NJ, USA, 2002), IEEE Press, pp. 29–36. 1, 2

[LW90] LEVOY M., WHITAKER R.: Gaze-directed volume rendering. In *SI3D '90: Proceedings of the 1990 Symposium on Interactive 3D Graphics* (1990), pp. 217–223. 2

[LWC*02] LUEBKE D., WATSON B., COHEN J. D., REDDY M., VARSHNEY A.: *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002. 1

[MMI*98] MUELLER K., MÖLLER T., II J. E. S., CRAWFIS R., SHAREEF N., YAGEL R.: Splatting errors and antialiasing. *IEEE Transactions on Visualization and Computer Graphics 4*, 2 (1998), 178–191. 2

[PPL*99] PARKER S., PARKER M., LIVNAT Y., SLOAN P.-P., HANSEN C., SHIRLEY P.: Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics 5*, 3 (1999), 238–250. 1

[RE02] RÖTTGER S., ERTL T.: A two-step approach for interactive pre-integrated volume rendering of unstructured grids. In *Proceedings of the 2002 IEEE Symposium on Volume Visualization and Graphics* (2002), pp. 23–28. 2

[RKE00] RÖTTGER S., KRAUS M., ERTL T.: Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Proceedings of IEEE Visualization 2000* (2000), pp. 109–116. 2

[RYL*96] REED D. M., YAGEL R., LAW A., SHIN P.-W., SHAREEF N.: Hardware assisted volume rendering of unstructured grids by incremental slicing. In *Proceedings of the 1996 IEEE Symposium on Volume Visualization* (Piscataway, NJ, USA, 1996), IEEE Press, pp. 55–ff. 1

[SCM99] SHEN H.-W., CHIANG L.-J., MA K.-L.: A fast volume rendering algorithm for time-varying fields using a time-space partitioning (TSP) tree. In *Proceedings of IEEE Visualization '99* (1999), pp. 371–377. 1

[WG92] WILHELMS J., GELDER A. V.: Octrees for faster isosurface generation. *ACM Transactions on Graphics 11*, 3 (July 1992), 201–227. 1

[WMG98] WITTENBRINK C. M., MALZBENDER T., GOSS M. E.: Opacity-weighted color interpolation for volume sampling. In *Proceedings of the 1998 IEEE Symposium on Volume Visualization* (New York, NY, USA, 1998), ACM Press, pp. 135–142. 2, 3

[WWHW97] WATSON B., WALKER N., HODGES L. F., WORDEN A.: Managing level of detail through peripheral degradation: effects on search performance with a head-mounted display. *ACM Transactions on Computer-Human Interaction 4*, 4 (1997), 323–346. 2

[ZPvBG02] ZWICKER M., PFISTER H., VAN BAAR J., GROSS M.: Ewa splatting. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 223–238. 2

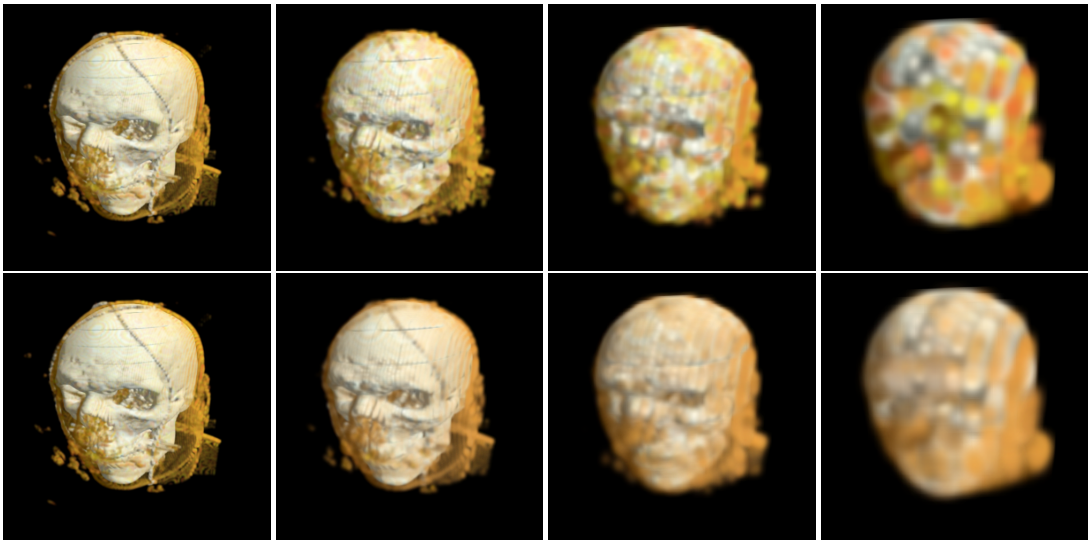


Figure 2: *Renderings of the Visible Human Male Head (256^3) at different levels of details. Top row: using mean value for approximation. Bottom row: using Gaussian function for approximation. From left to right, each image represents an additional level of coarsening by a factor of two in each dimension.*

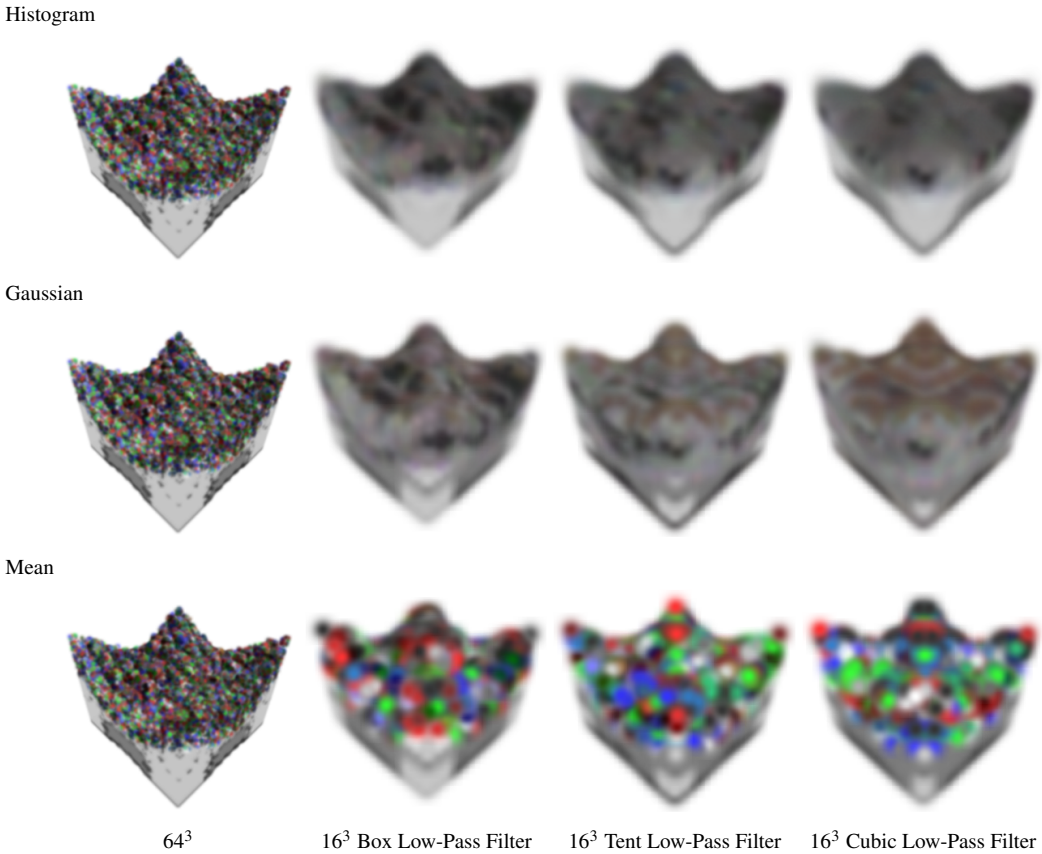


Figure 3: *Effects of downsampling distributions with different filters on the Richtmyer-Meshkov Instability dataset.*

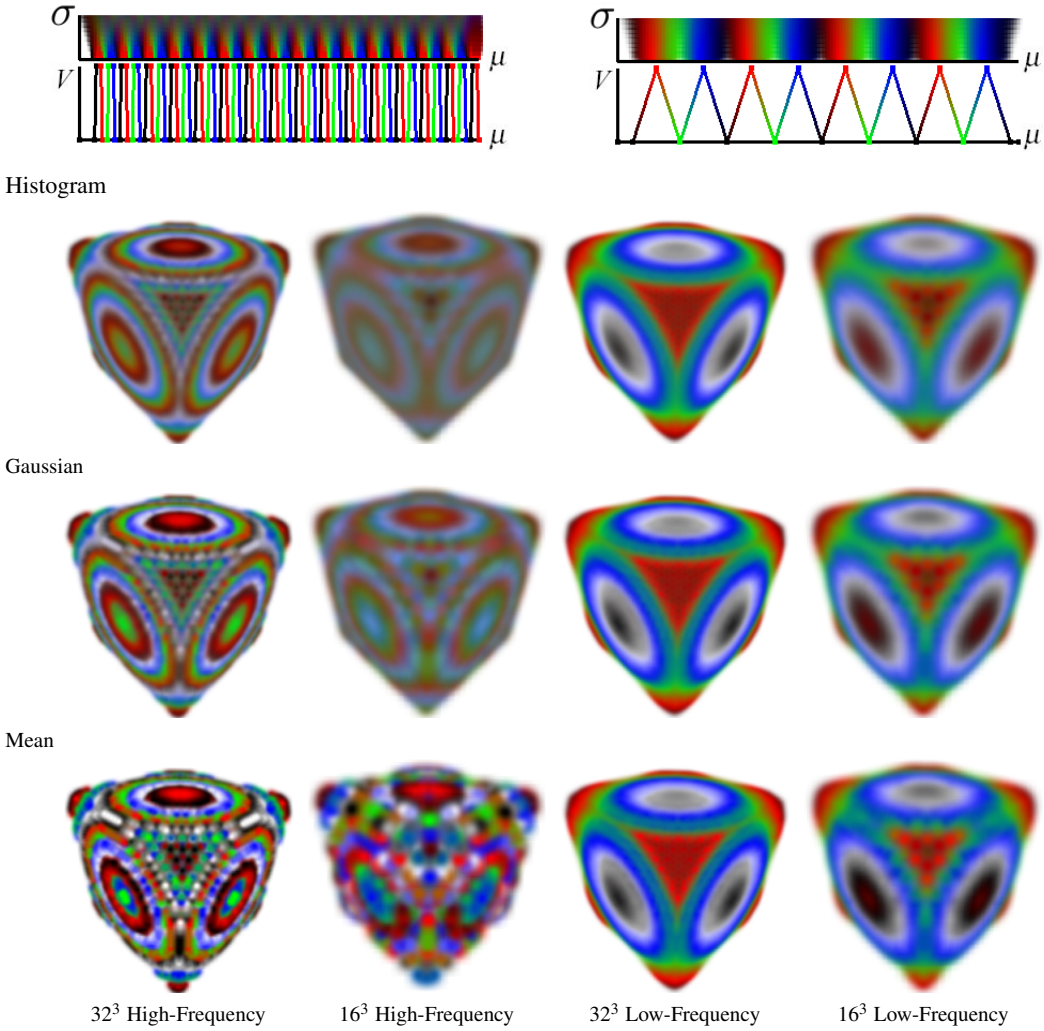
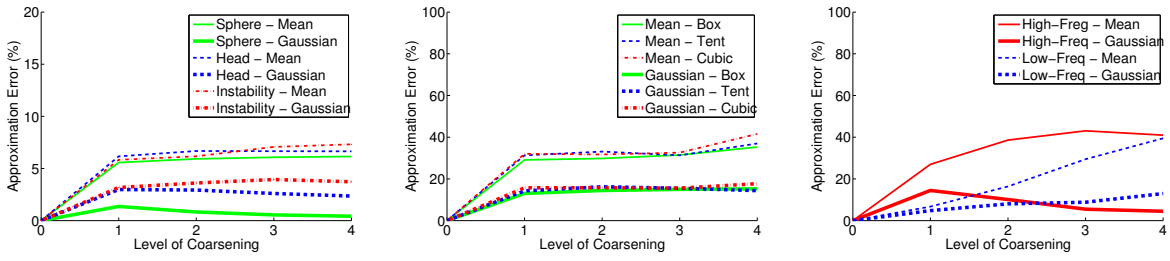


Figure 4: *Rendering with different transfer functions on the spherical distance dataset.*



(a) Histogram Approximations Errors (b) Effect of Filters (c) Effect of Transfer Functions

Figure 5: *(a) Errors of the mean and Gaussian approximations of the histogram for the three test datasets. At a certain level of detail, for each method, the reported error is the root mean squared error of the voxels' approximated histogram in comparison with their actual histogram, for all voxels in that level of coarsening. Rendering errors of the mean and Gaussian approximations (f) of the histogram on (b) the Richtmyer-Meshkov Instability dataset using different filters and (c) the spherical distance dataset using a high-frequency and low-frequency transfer function (V).*