SCI INSTITUTE
TECHNICAL REPORT

Image-Space Acceleration for Direct Volume Rendering of Unstructured Grids using Joint Bilateral Upsampling

Steven P. Callahan and Cláudio T. Silva

UUSCI-2008-002

Scientific Computing and Imaging Institute
University of Utah
Salt Lake City, UT 84112 USA

June 10, 2008

Abstract:

We describe a new image-space acceleration technique that allows real-time direct volume rendering of large unstructured volumes. Our algorithm operates as a simple post-process and can be used to improve the performance of any existing volume renderer that is sensitive to image size. A joint bilateral upsampling filter allows images to be rendered efficiently at a fraction of their original size, then upsampled at a high quality using properties that can be quickly computed from the volume. We show how our acceleration technique can be efficiently implemented with current GPUs and used as a post-process for a wide range of volume rendering algorithms and volumetric datasets.

THE UNIVERSITY OF UTAH

Image-Space Acceleration for Direct Volume Rendering of Unstructured Grids using Joint Bilateral Upsampling

Steven P. Callahan and Cláudio T. Silva

Abstract—We describe a new image-space acceleration technique that allows real-time direct volume rendering of large unstructured volumes. Our algorithm operates as a simple post-process and can be used to improve the performance of any existing volume renderer that is sensitive to image size. A joint bilateral upsampling filter allows images to be rendered efficiently at a fraction of their original size, then upsampled at a high quality using properties that can be quickly computed from the volume. We show how our acceleration technique can be efficiently implemented with current GPUs and used as a post-process for a wide range of volume rendering algorithms and volumetric datasets.

Index Terms—Acceleration Techniques, Image Processing, Direct Volume Rendering, Interactive Techniques

1 Introduction

The amount of data available from simulation and measurement is growing at an incredible rate. A major challenge for the visualization community is to develop methods that allow users to explore this data interactively [23]. For 3D scalar fields, direct volume rendering has become an important technique in research and commercial settings. Interactive volume rendering requires the efficient use of available computational resources to keep pace with the disparity, resolution, and complexity of the volumes that are commonly produced from simulations (e.g., computational fluid dynamics or structural mechanics) and measurements (e.g., environmental observation and forecasting systems).

Acceleration techniques that approximate full quality images are common to provide interactivity with volumes too large or complex to handle otherwise. The general idea is to switch to a reduced representation of the rendering during interaction, but still allow a full quality representation to be rendered if desired. Approximation strategies for both structured and unstructured volumes fall into two categories: those that operate in object-space and those that operate in image-space. Whereas object-space methods involve simplifying or downsampling the volume to reduce the amount of data rendered, image-space methods usually involve reducing the number of pixels that are rendered. The result is a fast approximation to the full-quality image that contains either low frequency error, such as blurring, or high frequency error, such as "jaggies" caused by aliasing, from object-space and image-space methods, respectively.

We propose an image-space approach that downsamples for efficient rendering then upsamples for display using a joint bilateral filter to remove aliasing artifacts while still preserving sharp features. Our upsampling algorithm is a post-process that can be used independently or in combination with existing object-space and image-space acceleration approaches with very little computation or implementation overhead.

The bilateral filter [31] was first introduced as a method for denoising images and works by combining a linear kernel, such as a Gaussian, with a non-linear, feature preserving term that weights the pixels based on intensities. The introduction of a separate reference image for performing the feature preservation is useful in some cases and is termed joint (or cross) bilateral filtering [12, 27]. This has recently been shown to be useful for enhancing images with solutions computed over downsampled images [17]. We build on this latter approach to improve volume rendering performance by rendering normally into a downsampled image and combining that with a low-cost reference image computed at full size. Instead of pixel intensities, our reference image contains depth information that can be used to encode the shape of the volume in a full size image. The result is an image that preserves the color of the downsampled image with the sharp features of the reference image. For opaque renderings, this has the appearance of smoothing the geometry in object-space, though it happens entirely in image-space. Figure 1 shows an example of the effect of our algorithm applied to an opaque rendering of a triangle mesh.

Our main contributions are as follows:

- We introduce a simple and fast image-space acceleration algorithm based on joint bilateral upsampling that results in substantial improvements for virtually any fragment or pixel bound volume renderer for unstructured grids;
- We provide a method for quickly capturing reference images of the volume that are used to improve the quality of our technique over traditional approaches;
- We describe how our algorithm can performed as a post-process on the latest graphics hardware with very little overhead;
- We provide quality and timing results for our image-space acceleration algorithm on a variety of datasets and using several volume rendering algorithms.

The rest of the paper is organized as follows. We summarize related work in Section 2. In Section 3 we describe our acceleration based on joint bilateral upsampling, how it can be applied to volume rendering, and the implementation details. In Section 4, we provide results of our algorithm and in Section 5 we discuss the trade-offs of its use. Finally, in Section 6 we conclude and provide avenues for future work.

2 Related Work

2.1 Acceleration for Direct Volume Rendering

Acceleration techniques for direct volume rendering have been the subject of much research in the visualization community. For a more complete summary of volume rendering algorithms for structured and unstructured grids, we refer the reader to recent surveys [16, 28, 30].

For structured grids, some of the original acceleration techniques are performed in image-space, and are still in use today to make ray casting more efficient. Levoy introduced the idea of casting one ray for multiple pixels [22], casting more rays in areas that vary across neighboring rays [21], or by not casting any rays in regions that do not contribute to the final image [20]. Extending these ideas, Danskin and Hanrahan [9] introduced adaptive ray sampling to sparsely sample along viewing rays in homogeneous regions. A similar object-space approach was introduced by Parker et al. [26], where the volume is partitioned into bricks that can be skipped during ray traversal. One class of acceleration techniques are multi-resolution or level-of-detail

• *Steven P. Callahan and Cláudio T. Silva are with the Scientific Computing and Imaging Institute at the University of Utah, E-mail: {stevec,csilva}@sci.utah.edu.*

(a) Original (b) Bilateral (c) Linear

Fig. 1. The dragon dataset rendered (a) normally at full opacity at 512^2, (b) upsampled from a 128^2 rendering using our joint bilateral filter that combines a low resolution color buffer with a high resolution depth buffer, and (c) upsampled linearly from a 128^2 rendering. Our bilateral upsampling is an acceleration method that can be applied to volumes to remove unwanted aliasing while still preserving sharp features.

(LOD) methods, that trade off quality of results for speed in rendering. With the advent of hardware-accelerated texture-based volume rendering [5], LaMar et al. [18] introduced a multi-resolution approach for slicing regions of the volume at different resolutions that are stored in an octree. This allowed regions of less interest, such as those farthest from the view point, to be drawn at a coarser resolution. Weiler et al. [34] improved upon this idea by making it more efficient and guaranteeing consistent interpolation between different resolution levels.

For unstructured grids, the image-space approaches for ray casting introduced by Levoy for structured grids are still applicable. However, object-space approaches are not so easily adapted. To mitigate this problem, Leven et al. [19] sample the unstructured grid regularly into an octree hierarchy that can be rendered using LOD techniques for structured grids. Volume simplification techniques, such as edge collapsing via the quadric error metric [15], provide the means for reducing the geometry representation to improve rendering performance. Adapting the simplification paradigm for LOD, Cignoni et al. [8] proposed a technique for creating a progressive hierarchy of tetrahedra that are stored in a multi-triangulation data structure that is updated dynamically for interactivity. More recently, Callahan et al. [6] introduced a simpler approach that samples the geometry during rendering based on a pre-computed importance. This approach avoids hierarchies and maintains interactivity by dynamically adjusting the amount of geometry rendered at each frame.

Our approach is similar in spirit to the original image-space approaches that reduce the number of pixels that are processed by rendering a downsampled image for speed. Thus performance gains using our method can be substantial for algorithms that are pixel bound, such as a ray caster or a raster algorithm with costly per fragment processing.

2.2 The Bilateral Filter for Upsampling

Since first introduced for image denoising [31], the bilateral filter has been used for many image processing applications, such as tone mapping for high dynamic range imaging [10]. The filter has also been used for problems other than image processing, such as mesh denoising [14]. For upsampling, the use of the bilateral filter is relatively new and has seen less use. Durand et al. [11] applied it to compute advanced shading effects with fewer samples. Sawhney et al. [29] adapted the filter for stereoscopic images at different resolutions. More recently, Kopf et al. [17] showed how joint bilateral upsampling could be used for computing downsampled solutions over an image and combining them with the original image. They demonstrated tone-mapping, colorization, stereo depth, and graph cuts as applications for the approach. Our method is similar to this latter technique, but instead of enhancing an image with a downsampled solution, we render our image, upsample it, then enhance it with a computed solution.

3 The Algorithm

Our acceleration algorithm is briefly summarized by the following steps:

1. Render the volume into a small offscreen image I using an existing volume rendering algorithm.
2. Render the boundary geometry of the volume at full size and capture the depths of the fragments in a reference image R.
3. Upsample the offscreen image I to full size using texturing hardware and combine it with the reference image R using the joint bilateral filter.

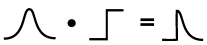
Figure 2 shows the visual effect of our algorithm on a volume. The details of each step in our method are described in the remainder of this section.

3.1 The Joint Bilateral Upsampling Filter

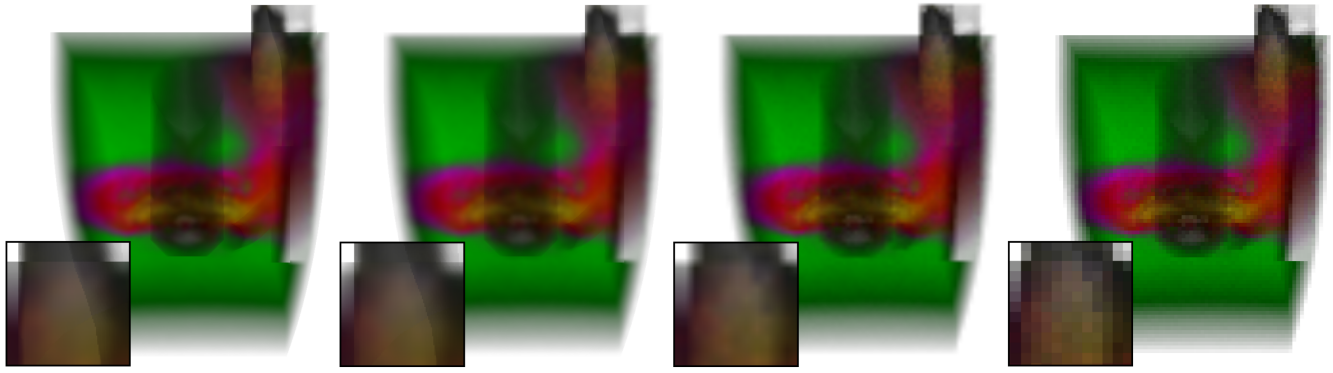
The original bilateral filter uses both a domain (spatial) and a range filter kernel on the input image to produce a denoised output image. For some position p, the filtered result is:

$$J_p = \frac{1}{k_p} \sum_{q \in \Omega} I_q f(\| p - q \|) g(\| I_p - I_q \|), \quad (1)$$

where f is the spatial filter, such as a low pass filter that operates on pixel colors centered over p, and g is the range filter kernel, such as a low pass filter that operates on pixel intensities centered over p. Ω is the spatial support of the kernels f and g, and k_p is the normalization computed as the sum of the f and g filter weights. Intuitively, $f \cdot g$ is just a new filter kernel that changes per pixel to respect intensity boundaries.



The joint bilateral upsampling filter uses separate images at different resolutions for the domain and range to compute an upsampled solution S from a given high resolution image I and a low resolution solution R that is used as a reference image:



(a) Original (b) Bilateral (c) Linear (d) Nearest

Fig. 2. The SPX dataset rendered using a software raycaster (a) normally at a 1024^2 resolution at one frame per second, and upsampled from a 128^2 image at ten frames per second using (b) our feature preserving joint bilateral upsampling, (c) linear interpolation, and (d) nearest neighbor interpolation (similar to a method that casts one ray per 8^2 pixel grid). Only the original and our bilateral method preserve the diagonal edge that appears in the center of the inset images.

$$S_p = \frac{1}{k_p} \sum_{q_\downarrow \in \Omega} R_{q_\downarrow} f(\| p_\downarrow - q_\downarrow \|) g(\| I_p - I_q \|), \tag{2}$$

where p and q denote coordinates in I, and $p_\downarrow$ and $q_\downarrow$ denote the corresponding coordinates in the low resolution reference image R. This formulation is used to compute costly solutions for high resolution images at lower resolutions. Our algorithm is different, as an inexpensive solution R at high resolution is used to upsample a low resolution image I. In the same notation, our filter could be expressed as:

$$S_p = \frac{1}{k_p} \sum_{q_\downarrow \in \Omega} I_{q_\downarrow} f(\| p_\downarrow - q_\downarrow \|) g(\| R_p - R_q \|), \tag{3}$$

where p and q denote coordinates in the high resolution reference image R, and $p_\downarrow$ and $q_\downarrow$ denote the corresponding coordinates in the low resolution image I.

3.2 Computing the Reference Image

To preserve features in our upsampled version, a full resolution reference image is needed for the range component of the bilateral filter. This reference image needs to be fast to compute and general enough to apply to a variety of volume renderers. For unstructured grids, it is common to represent the domain (or boundaries) of the volume to facilitate the understanding of the features that are contained therein. Fortunately, the boundaries are easy to capture—they are often already used by volume rendering algorithms as starting points for ray traversal [4, 32] or as the base case for object-space LOD [6]. Other sharp boundaries within the volume could also be used as well, such as those provided by isosurfaces, if they are readily available.

For our bilateral upsampling to faithfully preserve the features of the volume's domain, more than just the front-most boundary needs to be captured. Multiple depth layers are already used by many volume renderers to handle non-convex meshes either in software [4] by creating a sorted depth list for each pixel, or in hardware [3, 33] using depth peeling [13]. Depth peeling is a multi-pass algorithm that captures one layer of depth on each pass, starting with the nearest fragment per pixel, then the second nearest, third nearest, and so on. This can be performed efficiently in hardware by rendering the first pass normally, resulting in a depth buffer of the nearest surface. In subsequent passes, the depth buffer computed in the previous pass is used to *peel* away depths less than or equal to those already captured in previous passes. These depth peeling passes can even be reduced to a single pass using stencil routing [2].

To compute the reference image in an existing volume rendering algorithm, we leverage the framework already in place for capturing depths whenever possible. If depths are not already captured, we simply add a depth peeling pass to the renderer. The number of depth passes that are used is dependent on the volume being rendered and the opacity at which it is being rendered. In our experience, two or three layers are sufficient for most datasets to capture the visible boundary features.

3.3 Implementation

Our joint bilateral upsampling is implemented with minimal changes to an existing algorithm. The low resolution image I and the reference image R are rendered offscreen. Then in a final pass, a full resolution, screen-aligned quadrilateral is drawn that binds both images as textures and uses a fragment shader to perform the joint bilateral filter. If the texturing hardware is set to linearly interpolate I, the small resolution image will be upsampled to full resolution linearly in the shader, improving the quality of the upsampling by adding an inexpensive low pass filter. In the shader, the I and R textures are accessed using the same coordinates to retrieve color and depth information used in the joint bilateral filter.

For each pixel p in the final image, the joint bilateral filter is a low pass filter that blurs a fixed neighborhood around p to remove noise. Choosing the spatial support Ω for the filter should be based on the amount of upsampling that is being performed on I, i.e., more blurring is required for higher upsampling factors. We match the spatial support for the joint bilateral filter with the spatial support for the linear interpolation performed by texturing hardware: if upsampling to 1024^2, a 512^2 image will use $\Omega = 4$, a 256^2 image will use a $\Omega = 8$, etc.

For domain and range filters, f and g, we use Gaussian low pass filters:

$$f(x,y) = g(x,y) = e^{-D(x,y)^2/2\sigma^2}. \tag{4}$$

For the domain filter f operating on the low resolution image I, $D(x,y)$ is the distance between (x,y) and the origin of the filter p, and σ is the spread of the Gaussian, or $\Omega/2$. For the range filter g operating on the reference image R, $D(x,y)$ is the difference between the depth value at (x,y) and the depth value at p. For multiple depth layers, this simply becomes the distance between the vectors defined at (x,y) and p. The range σ is the value that expresses the resolution of the depth features that should be preserved. Thus, the range σ is dependent on the resolution of the depth buffer and should be as low as possible to capture depth changes, without causing artifacts due to depth precision. We have found a $\sigma = 0.01$ for the range to be adequate.

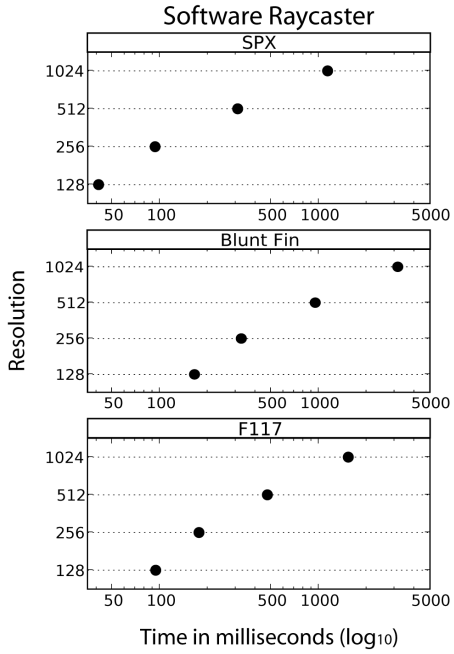


Fig. 3. Timing statistics for a software raycaster for various resolutions upsampling to 1024^2 using our technique. The 1024 resolution represents the time for a full quality image without upsampling.

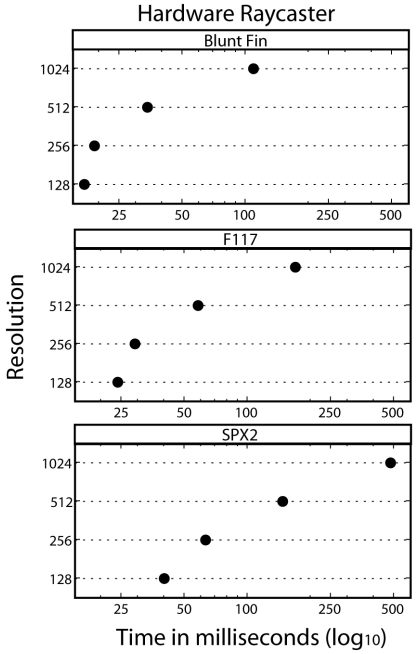


Fig. 4. Timing statistics for a hardware raycaster for various resolutions upsampling to 1024^2 using our technique. The 1024 resolution represents the time for a full quality image without upsampling.

4 Results

To demonstrate the flexibility of our joint bilateral upsampling algorithm for unstructured grids, we added it to existing source code for three popular algorithms and report numbers for the speed and quality of our technique.

4.1 Timing Results

The fragment shader for bilateral sampling itself is relatively inexpensive, for a 1024^2 image, kernel sizes of 4, 8, and 16 achieve framerates of 100 fps, 50 fps, and 15 fps, respectively, on a Quadro FX 5600 graphics card. For our timing experiments, we used the datasets shown in Table 1 to gather statistics for upsampling from 128^2, 256^2, and 512^2 to 1024^2 and compare them with the original rendering times for a 1024^2 image. At each resolution, we rendered the dataset from 14 viewpoints, defined by the corners and faces of a cube around the dataset, and averaged the times. To make comparisons between upsampling factors easier, we used a uniform kernel size of 12 (at 25 fps) for all resolutions. All of our results were rendered on a machine with 2 Dual Opteron 2.25 GHz processors, 4 GB RAM, and an NVidia Quadro FX 5600 graphics card with 1.5 GB RAM.

Dataset	Vertices	Tetrahedra
SPX	3 K	13 K
Blunt Fin	41 K	187 K
F117	49 K	240 K
SPX2	166 K	828 K

Table 1. Experimental datasets, with vertex and tetrahedron count, used for measuring rendering performance.

The first algorithm that we modified is a software raycaster from Bunyk et al. [4] that has freely available source code and runs completely on the CPU. The algorithm first rasterizes boundary triangles to capture starting and ending points for the rays at each pixel. It then marches rays through the volume cell to cell by exploiting connectivity of cell faces. To make the modification, we perform the ray casting as normal, except into a small image. We then use the existing depth capturing code to find the boundary depths in a large image. These two images are then bound as textures and rendered to the screen using our fragment shader written in OpenGL. Figure 3 shows a series of plots for several datasets comparing times (logarithmically scaled) for the varying resolutions. In our experiments, the acceleration for these datasets ranges from about 16 times to 28 times for 128^2 resolution images upsampled to 1024^2.

The second algorithm that we modified is a hardware-assisted raycasting algorithm from Bernardon et al. [3] that is freely available and is written in DirectX9. As with the software raycaster, we were able to adapt the algorithm to render into a small offscreen buffer and use the existing depth peeling routines to capture the depth in a full size offscreen buffer. An HLSL program was then used to perform the joint bilateral upsampling and display the final image. Figure 4 shows a series of plots for comparing times (also logarithmically scaled) for varying resolutions. With the hardware raycaster, acceleration gains range from about seven times for the smallest dataset to about twelve times for the largest for 128^2 resolution images upsampled to 1024^2.

4.2 Quality Results

By using a full size reference image, our joint bilateral upsampling is able to achieve better imagery than upsampling alone. We show this both quantitatively and visually. Figure 5 shows rate distortion curves for the quality of upsampling using our joint bilateral filter and linear interpolation (as provide by texturing hardware). The measurements were computed using root mean squared error (RMSE) comparisons between full quality images at 1024^2 and images upsampled at various resolutions. In all cases, our upsampling exhibits less error than with linear interpolation alone. Figure 6 shows rendered solutions at various resolutions for a visual comparison of the quality change.

One interesting application of our filter is in improving the appearance of existing acceleration techniques by denoising results while still preserving edges. We added our upsampling filter to the HAVS volume rendering algorithm [7] to improve the appearance of a dynamic LOD algorithm that operates by sampling the geometry of the volume [6]. HAVS sorts the triangles that compose the mesh first in object-space using a simple sorting routine, then in image-space by storing a fixed number of fragments. The LOD algorithm samples the triangles before the sorting, based on pre-computed importances, to make the rendering more efficient. Because the LOD already uses boundary geometry

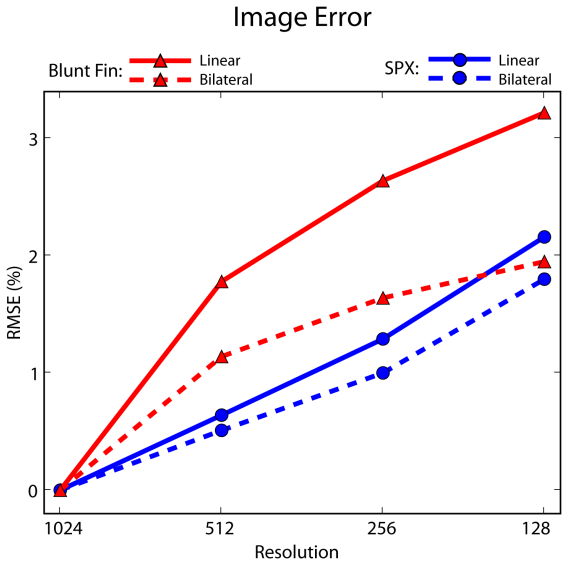


Fig. 5. Rate distortion curves for various resolutions comparing RMS error of full quality images versus our method and linear upsampling.

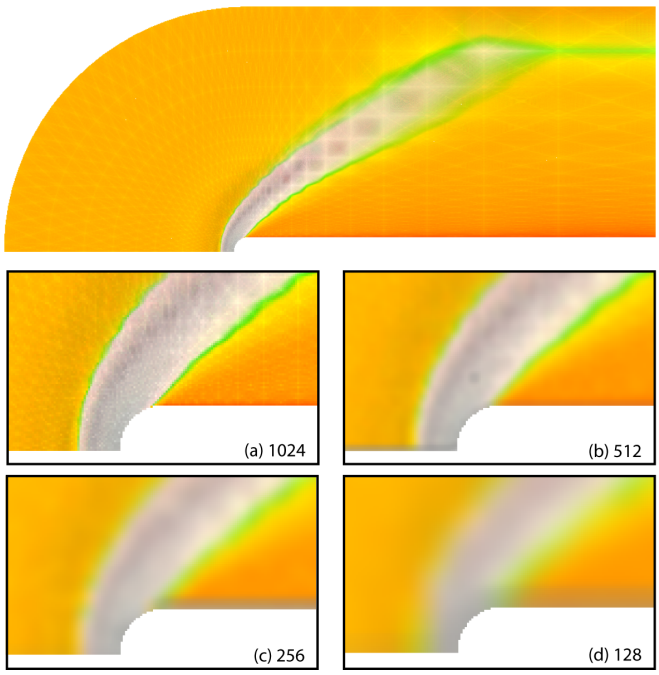


Fig. 6. The Blunt Fin dataset rendered into a 1024^2 image at (a) full quality and upsampled from (b) 512^2, (c) 256^2, and (d) 128^2 using our technique.

as the base sampling case, we were able to add an additional pass to render these boundaries into our reference image before combining it with the original rendering pass using joint bilateral upsampling.

Due to the nature of the algorithm, HAVS is more vertex bound than pixel bound. Thus, the acceleration when using our upsampling approach is negligible on the most recent graphics cards. However, by sparsely sampling the geometry in the mesh but leaving boundary geometry, the number of primitives rendered is reduced and the speed of the algorithm is improved. This sample-based simplification has the side effect of producing high frequency error in the reduced representation, unlike domain-based simplification techniques (i.e., simplification via edge collapses [15]). Using our joint bilateral filter on the resulting imagery, we are able to reduce the noise and improve the overall appearance of the LOD strategy with little affect on the performance. Figure 7 shows an example of this LOD before and after our joint bilateral filter is applied. Because the acceleration due to the LOD algorithm is dominant, the rendered image I does not need to be computed at a reduced representation and upsampling is not necessary.

5 Discussion

Because our method is simple, it can easily be utilized as a technique to accelerate interaction, while still allowing full quality images to be rendered when the user stops interacting with the viewing parameters. Tools such as ParaView [25] use a similar strategy during rendering, by either changing the number of slices for texture based methods, or number of rays for raycasters. We envision our algorithm as a replacement or enhancement for these existing techniques because it produces better approximations of the full quality image with less visual artifacts. Because it is easy to change the speed/quality trade-off by adjusting the upsampling factor, our algorithm could also be used for dynamic level-of-detail.

Many existing acceleration techniques that trade-off speed for image quality create high frequency error in the resulting image in the form of stair-casing or aliasing artifacts. In contrast, our method produces low frequency error, which results in more visually pleasing images that retain edges that are supposed to be in the image, while removing those that are not. Although we use the joint bilateral upsampling, other upsampling strategies have been introduced [35] and could be used instead. However, without the additional shape information that is provided by the reference image, other upsampling strategies are not likely to perform as well as they can result in halos and other undesired artifacts, as shown by Kopf et al. [17].

Our solution for performing joint bilateral upsampling for volume rendering was implemented in OpenGL and DirectX using fragment programs. We also implemented our algorithm with NVidia's CUDA library, which is efficient for offscreen processing, but not as fast as fragment programs for interactive graphics. Even for large images and large filter domains, the computational cost of our algorithm is not high relative to the volume rendering cost. As with most acceleration techniques, the trade off for image quality is performance (i.e., more downsamping results in higher speed).

Our experiments included several datasets at various sizes to demonstrate the acceleration that our technique can provide. The size of datasets used in these experiments was limited by the volume rendering methods we used, not by any limitations of our acceleration technique. Volume rendering algorithms that handle larger datasets, such as raycasters that use bricking strategies for memory management [24] or point-based techniques that are fragment-bound [1], would also benefit from our acceleration technique.

6 Conclusions

In this paper, we have presented an acceleration technique for unstructured grid volume rendering that operates in image-space. By rendering small images and upsampling them with a smart filter, we have measured performance improvements of up to 30 times. Our upsampling strategy based on joint bilateral upsampling results in a high quality approximation that avoids the high frequency noise common in existing acceleration techniques based on rendering reduced representations of the data. The major advantages of our algorithm are that it is simple to implement, it is flexible enough to be included as a post-process to virtually any direct volume rendering algorithm, and it can easily be used in combination with existing acceleration techniques.

In the future, we would like to explore using additional information in the reference image to preserve internal features. If transfer functions remain static, internal boundaries may be captured by rendering the geometry of an isosurface selected based on the transfer function. This would allow the method to be applied to volume renderers for structured grids as well.

Acknowledgments. The authors thank Jens Krüger and Peter Shirley for useful discussion, the Stanford University Computer Graphics Laboratory for the Dragon dataset, O'Hallaron and

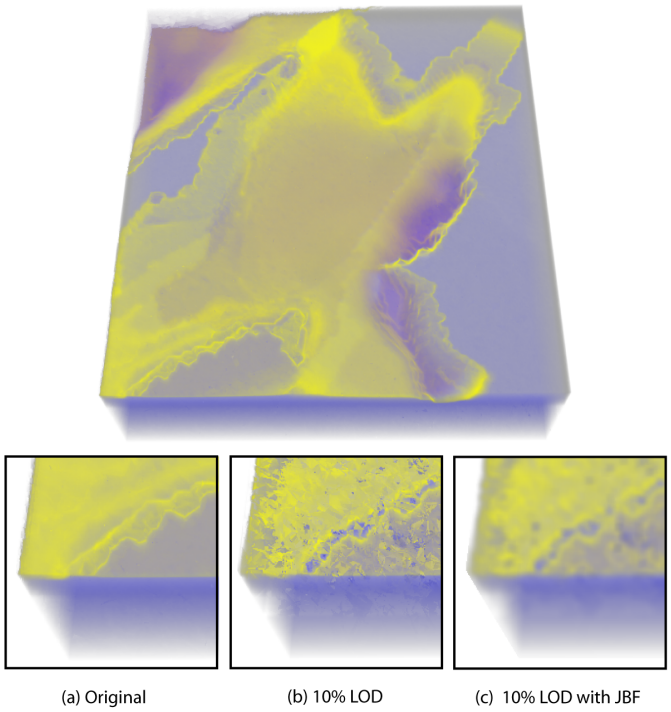


Fig. 7. The San Fernando Earthquake dataset (1.4 million tetrahedra) rendered using HAVS [7] with sample-based simplification [6]. (a) The full quality image is rendered at 1.7 fps compared with (b) sampling 10% of the geometry (20 fps) and (c) sampling 10% of the geometry then using the joint bilateral filter without upsampling to remove the high frequency error while still preserving boundary features (15 fps).

Shewchuk (CMU) for the earthquake dataset, Notrosso (electricite de France) for the SPX dataset, Haimes (MIT) for the F117 dataaset, and Hung and Buning (NASA) for the Blunt Fin dataset. This work is funded by the Department of Energy under the ASCI VIEWS program and the MICS office, the National Science Foundation (grants CCF-0401498, EIA-0323604, OISE-0405402, IIS-0513692, CCF-0528201), Sandia National Laboratories, Lawrence Livermore National Laboratory, an IBM Faculty Award, a University of Utah Seed Grant, and a University of Utah Graduate Research Fellowship.

REFERENCES

[1] E. W. Anderson, S. P. Callahan, C. E. Scheidegger, J. Schreiner, and C. T. Silva. Hardware-assisted point-based volume rendering of tetrahedral meshes. In *Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI)*, pages 163–170, 2007.

[2] L. Bavoil and K. Myers. Deferred rendering using a stencil routed k-buffer. In *ShaderX 6 - Advanced Rendering Techniques*. to appear.

[3] F. F. Bernardon, C. A. Pagot, J. L. D. Comba, and C. T. Silva. GPU-based tiled ray casting using depth peeling. *Journal of Graphics Tools*, 11(3):23–29, 2006.

[4] P. Bunyk, A. Kaufman, and C. T. Silva. Simple, fast, and robust ray casting of irregular grids. In *Proc. of Dagstuhl, Scientific Visualization*, pages 30–36, 1997.

[5] B. Cabral, N. Cam, and J. Foran. Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *IEEE/ACM Symposium on Volume Visualization*, pages 91–98, 1994.

[6] S. P. Callahan, J. L. D. Comba, P. Shirley, and C. T. Silva. Interactive rendering of large unstructured grids using dynamic level-of-detail. In *IEEE Visualization*, pages 199–206, 2005.

[7] S. P. Callahan, M. Ikits, J. L. Comba, and C. T. Silva. Hardware-assisted visibility sorting for unstructured volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 11(3):285–295, 2005.

[8] P. Cignoni, L. D. Floriani, P. Magillo, E. Puppo, and R. Scopigno. Selective refinement queries for volume visualization of unstructured tetrahedral meshes. *IEEE Transactions on Visualization and Computer Graphics*, 10(1):29–45, 2004.

[9] J. Danskin and P. Hanrahan. Fast algorithms for volume ray tracing. In *Workshop on Volume Visualization*, pages 91–98, 1992.

[10] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 21(3):257–266, 2002.

[11] F. Durand, N. Holzschuch, C. Soler, E. Chan, and F. X. Sillion. A frequency analysis of light transport. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 24(3):1115–1126, 2005.

[12] E. Eisemann and F. Durand. Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 23(3):673–678, 2004.

[13] C. Everitt. Interactive order-independent transparency. White paper, NVidia Corporation, 1999.

[14] S. Fleishman, I. Drori, and D. Cohen-Or. Bilateral mesh denoising. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 22(3):950–953, 2003.

[15] M. Garland and Y. Zhou. Quadric-based simplification in any dimension. *ACM Transactions on Graphics*, 24(2), Apr. 2005.

[16] C. Hansen and C. Johnson. *The Visualization Handbook*. Academic Press, 2004.

[17] J. Kopf, M. Cohen, D. Lischinski, and M. Uyttendaele. Joint bilateral upsampling. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 26(3), 2007.

[18] E. LaMar, B. Hamann, and K. Joy. Multiresolution techniques for interative texture-based volume visualization. In *IEEE Visualization*, pages 355–361, 1999.

[19] J. Leven, J. Corso, J. D. Cohen, and S. Kumar. Interactive visualization of unstructured grids using hierarchical 3D textures. In *IEEE Symposium on Volume Visualization and Graphics*, pages 37–44, 2002.

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

[21] M. Levoy. A hybrid ray tracer for rendering polygon and volume data. *IEEE Computer Graphics and Applications*, 2(4):33–40, 1990.

[22] M. Levoy. Volume rendering by adaptive refinement. *The Visual Computer*, 6(1):2–7, 1990.

[23] K.-L. Ma, J. Blondin, J. H. Chen, M. Rast, and R. Samtaney. Meet the scientists. In *IEEE Visualization Panels*, 2007.

[24] P. Muigg, M. Hadwiger, H. Doleisch, and H. Hauser. Scalable hybrid unstructured and structured grid raycasting. *IEEE Transactions on Visualization and Computer Graphics (Proc. of Visualization)*, 13(6):1592–1599, 2007.

[25] ParaView. http://www.paraview.org.

[26] S. Parker, M. Parker, Y. Livnat, P.-P. Sloan, C. Hansen, and P. Shirley. Interactive ray tracing for volume visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):238–250, 1999.

[27] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, 23(3):664–672, 2004.

[28] B. Preim and D. Bartz. *Visualization in Medicine*. Morgan Kaufmann, 2007.

[29] H. S. Sawhney, Y. Guo, K. Hanna, R. Kumar, S. Adkins, and S. Zhou. Hybrid stereo camera: an IBR approach for synthesis of very high resolution stereoscopic image sequences. In *ACM SIGGRAPH*, pages 451–460, 2001.

[30] C. T. Silva, J. L. D. Comba, S. P. Callahan, and F. F. Bernardon. GPU-based volume rendering of unstructured grids. *Brazilian Journal of Theoretic and Applied Computing (RITA)*, 12(2):9–29, 2005.

[31] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

[32] M. Weiler, M. Kraus, M. Merz, and T. Ertl. Hardware-based ray casting for tetrahedral meshes. In *IEEE Visualization*, pages 333–340, 2003.

[33] M. Weiler, P. N. Mallón, M. Kraus, and T. Ertl. Texture-Encoded Tetrahedral Strips. In *Symposium on Volume Visualization*, pages 71–78, 2004.

[34] M. Weiler, R. Westermann, C. Hansen, K. Zimmerman, and T. Ertl. Level-of-detail volume rendering view 3D textures. In *IEEE Symposium on Volume Visualization*, pages 7–13, 2000.

[35] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA, 1990.