A Methodology for Comparing Direct Volume Rendering Algorithms Using a Projection-Based Data Level Approach

Kwansik Kim and Alex Pang

Computer Science Department
University of California, Santa Cruz, CA 95064
ksk@cse.ucsc.edu, pang@cse.ucsc.edu

Abstract. Identifying and visualizing uncertainty together with the data is a well recognized problem. One of the culprits that introduce uncertainty in the visualization pipeline is the visualization algorithm itself. Uncertainties introduced in this way usually arise from approximations and manifest themselves as artifacts in the resulting images. In this paper, we focus on comparing different direct volume rendering (DVR) algorithms and their artifacts as a result of DVR algorithm selections and their associated parameter settings. We present a new data level comparison methodology that uses differences in intermediate rendering information. In particular, we extend the traditional image level comparison techniques to include data level comparison techniques. In image level comparisons, quantized pixel values are the starting point for comparison measurements. In contrast, data level comparison techniques have the advantage of accessing and evaluating the intermediate 3D information during the rendering process. Our data level approach overcomes limitations of image level approaches and provide capabilities to compare application dependent details as well as general rendering qualities. One of the key challenges with our data level comparison approach is finding a common base for comparing the rich variety of DVR algorithms. In this paper, we present how a projection algorithm can be used as a base for comparing other DVR algorithms. In addition, a set of projection-based metrics are derived to quantify the comparison measurements among DVR algorithms. The results presented in this paper complement our earlier findings where a ray-based approach was used as the base for comparing other DVR algorithms.

Key Words and Phrases: Scientific visualization, direct volume rendering, uncertainty, error, difference, similarity, metrics.

1 Introduction

Although a large number of visualization methods have been developed, few provide the foundations and methodologies to compare and evaluate them against each other. This shortcoming has been raised as a critical issue for the objective interpretations of scientific data [2, 9, 21, 22]. Direct volume rendering (DVR) is one of the most popular

methods for visualizing 3D scalar data sets. These methods have been extensively investigated resulting in a rich variety of algorithms. Some of these can be found in [5, 6, 8, 10, 13, 15–17]. Unfortunately, this plethora of DVR methods produces images that are different from each other. In critical applications, it can be very disconcerting to have even slight differences in images rendered by various volume rendering algorithms. It is therefore necessary for both users and developers to be able to do in-depth study on the differences. Important differences stem from varying degrees of approximation in reconstruction, material classification (e.g. transfer functions), and accuracy in physical simulation of light and material interactions. Because it is a perceptual issue and DVR algorithm variations are often arbitrarily non-linear, it is very difficult to quantify errors (or uncertainties) that are introduced in the final rendered image. In addition, unlike quality issues in the image synthesis and image processing communities, criteria for measuring the quality often depend on the purpose of the particular visualization. They are not necessarily about how realistic or aesthetically satisfactory the final images are.

Fortunately, more and more DVR papers address the important issue of volume rendering qualities and comparisons [9, 11, 20, 21]. In those that address the issue, the norm is to use image level comparisons, and sometimes at the image summary level at best. Combinations of image analysis methods and summary statistics have been used to compare rendered images. For example, wavelet based image metrics [1] have been proposed to help determine perceptual similarities between volume rendered images. However, there are limitations to image metrics such as summary statistics and perceptual metrics as well. Williams and Uselton [21] first described some of the difficulties and limitations of image comparison. They presented the need for providing rigorous specifications of the volume rendering parameters, a set of image difference classifications and corresponding metrics for each category. However, there are inherent limitations to image level comparisons. While image level comparisons can provide information as to the location and degree by which two images differ, they often do not provide enough information as to why the two resulting images differ in general.

This paper addresses this shortcoming by proposing the use of data level comparison techniques. The goal is that if two images differ, then we want to provide an explanation for the causes for such difference. The name *data level comparison* was inspired by the work of Pagendarm and Post [12]. Data level methods incorporate intermediate and auxiliary information in the rendering process and use this information to generate a data level comparison image. Another distinguishing factor is that image level comparisons quantize data values then compare, while data level comparison methods compare data values then quantize thereby resulting in a greater dynamic range of comparison values. As in the Figure 1(a), we can compare volume data, intermediate rendering data, or final rendered images. In this paper, we use data level comparison to take advantage of any intermediate information in the volume rendering pipeline before the final image is generated.

Because of the wide variety of DVR algorithms, some of them have drastically different approaches and therefore difficult to obtain registered, intermediate rendering information for comparisons. Hence, we compare algorithms by first mapping them to a base or reference algorithm and then deriving metrics natural to the base algorithm (Figure 1(b)). We consider this mapping as invertible and thus we can experiment with

multiple base algorithms and develop their corresponding metrics. In this paper, we use the projection algorithm as our base algorithm, and focus our attention on DVR algorithms applied to regularly gridded data.

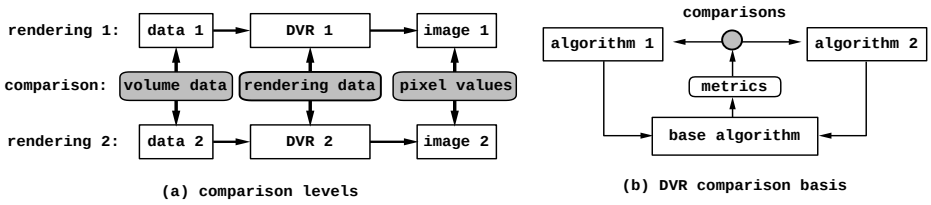

Fig. 1. Types of comparisons (a) and basis for comparing algorithms (b). (a) highlights three different areas where one can perform comparison: data, rendering information, and pixel values. Data level comparison includes comparison of data and rendering information, while image level comparison works with pixel values. (b) shows the architecture for comparing two different DVR algorithms via a common base algorithm.

2 Previous Work

In our earlier work [3, 4], we described how DVR algorithms can be simulated using ray casting as a base or reference algorithm (Figure 1(b)). For example, projection-based algorithms were simulated by intersecting the ray with the set of projected polygons of each cell. We also developed several ray-based metrics such as the distance the ray traveled, the number of samples along each ray, and similarity measures for sample colors and data along each ray. Here, we complement our earlier work by using a projection-based algorithm as our base or reference algorithm. This involves mapping or simulating other DVR algorithms, such as ray casting, into projection-based DVR algorithms.

3 Data Level Approach

Figure 1(a) delineates image level comparison (using pixel values) from data level comparison (using volume and rendering data). In DVR, the intermediate information may include items related to the rendering process. Examples include: distribution of opacities, values that contribute to a pixel, and minimal or maximal sample values along the viewing direction. Intermediate information may also include items related to the volume rendering algorithm. Transfer functions, sampling locations and frequency, interpolation functions, opacity threshold, and projection filters are examples of information related to the volume rendering algorithm. It should be noted that in some cases this distinction is blurred. In either case, these information and others can be used to generate metrics for data level comparison.

There are two different approaches for collecting these intermediate information. One approach is to collect them directly from the different DVR algorithms. The other

approach is to first map other DVR algorithms to a base or reference algorithm, then collect the metrics from this base algorithm. Both approaches have their advantages and disadvantages. The main advantage of the first approach is the accuracy of the intermediate information since we do not need to simulate or map the algorithms to a base algorithm and thereby possibly introducing some errors. The disadvantages of this approach are: (a) difficulty in collecting registered and meaningful intermediate information that can be compared across different DVR algorithms, and (b) the need to put data collection code in different DVR implementations. On the other hand, the main advantages of the second approach are: (a) ability to compare different DVR algorithm on the same set of intermediate data as collected from the reference algorithm, and (b) multiple choices for the reference algorithm since the mapping is invertible. That is, as demonstrated by this paper and our earlier work [3], a ray-based DVR algorithm can be mapped to a projection-based DVR algorithm, and vice versa. The second approach also has a number of disadvantages: (a) difficulty of completely specifying all the rendering parameters to precisely control the mapping of some DVR algorithms to the base algorithm, and (b) the need to map to the base algorithm or simulate other DVR algorithms. The critical thing to address the first difficulty of the latter approach is that rigorous specifications of all rendering parameters are necessary in order to faithfully simulate a given DVR algorithm. In the absence of this rigor, emphasis should be placed on specifying the more important rendering parameters. Our basic assumption is that the major differences from different DVR images result from those important rendering parameters.

4 Projection as a Base Algorithm

A popular model for computing the color intensities as the light passes through a sequence of translucent material is to assume that the light is attenuated by the opacities of the material. This is often described by the compositing equation below:

$$C_{out,i} = \alpha_i C_i + (1 - \alpha_i) C_{in,i} \tag{1}$$

α_i is the opacity and C_i is the color of the ith object to be composited. $C_{in,i}$ is the color intensities before the ith object is rendered (or composited) and $C_{out,i}$ is the result of rendering (or compositing) the ith object. Integrating this equation yields the so called *volume rendering integral*:

$$C(a,b) = \int_a^b E(s) e^{-\int_a^s \delta(x)dx} ds \tag{2}$$

$C(a,b)$ is the color intensity contributions through a line from position a to b. E is the color emission function and δ is the differential opacity function. One may view different DVR algorithms as variations on how to approximate the emission function E and the solution of the volume rendering integral. For example, Eqn. 2 has a closed form solution if we assume constant color emission and opacity between the interval of integration. Image order algorithms, like ray casting, discretize the volume rendering integration process and compute multiple compositions (Eqn. 1) of the integrated colors

($C(a,b)$ in Eqn. 2) of small adjacent sampling intervals. The simplest approximation is simply $C(a,b) = E(a)$ or $E(b)$ at each sampling point. Our challenge then is to make a projection-based algorithm general enough to facilitate simulations of the different variants of DVR algorithms.

The structure of our base algorithm is the same as that of any object order algorithm like *coherent projection* [17] or *splatting* [16, 5]. The basic idea of these object order algorithms is to compute the contributions of the sub-volumes and composite them to the images in the proper order. We conveniently collect any desired metrics as the algorithm calculates contributions to the final rendered image. The structure of our comparison algorithm is:

Procedure **project**:

1. Determine the order of volume cells or voxels to be projected.
2. For each cell or voxel, in the appropriate order
   (a) Compute the contribution of the cell or voxel and composite.
   (b) Collect *metrics.*

The term *cell* refers to the cube formed by the 8 neighboring data points in a regular grid volume. The term *voxel* refers to the region around a data point. The shapes of the voxel depend on the data model. In *splatting* algorithms, the shape (and thus its 2 dimensional contribution to the final rendered image) is defined as a function of the distance from the data point.

Projection-based algorithms [17, 14] usually take advantage of hardware polygon shading found in modern workstations. However, the projection algorithms are general enough to be extended and simulate effects of other DVR algorithms if they are implemented in software using *volume scan conversion*. Therefore, in step (2a) of procedure **project**, we implemented volume scan conversion to process a set of projected polygons in software. This allows us to:

- simulate other variations of DVR algorithms (e.g. raycasting) by varying the volume integration method at each pixel in the volume scan conversion procedure.
- derive projection-based *metrics* to compare a wide range of algorithms on a common basis.

As pointed out in [19], scan conversion of surface polygons has been studied extensively, but the process in volume rendering is a different problem. In scan converting surfaces, the surface (or polygon) has the color properties to be scanned. However, in scan converting a volume cell (or voxel), we need to process the material (or data) between volume cell faces that has the color properties. In the following description of the procedure **composite_sub-volume**, the details of our volume scan converting is explained. The structure of our base algorithm is the same as other volume scan converting procedures but it carries additional information such as relative locations of front and back points for simulations of other algorithms (at lines in 2(c)(2) of procedure **composite_sub-volume** below).

Procedure **composite_sub-volume**:

1. Decompose the given sub-volume into a set of projected polygons
2. For each polygon **P**,
   (a) Get the range **R** of the scanlines that **P** occupies in screen space and set up the edge table.
   (b) Empty the active edge table.
   (c) For each scanline of the range **R** in bottom-to-top order,
      (1) Update active edge table.
      (2) For each pixel from the left edge to the right edge,
         * Update front and back pixel information.
         * Compute the contribution between the front and back sub-volume at the pixel.

For projection based algorithms such as *Coherent Projection* [17] or *splatting* [5, 16], the footprint of a cell is often approximated as a set of polygons. The typical usage of these projection algorithms can be easily simulated with our base algorithm by using surface scan conversion only. In this case, the contribution from a cell (or voxel) is simply a color intensity interpolated within the footprint polygon. Sampling and reconstruction of volume data and color mappings are done only at the vertices of projected polygons as in [5, 16, 17].

Image order algorithms, such as variations of ray casting, can be simulated by changing the sampling patterns and the approximations to the volume rendering integral (i.e. last line of Procedure **composite_sub-volume**). Figure 2 (color plate 1) illustrates the simulation of variants of ray casting while projecting a volume cell. The figure shows a cross sectional view perpendicular to the projection plane. For each pixel, the base algorithm updates the information needed to project a volume cell, such as data values and relative locations of the front and back points within the cell. Using the specifications of rendering parameters, such as sampling patterns, we can calculate the color contributions of the cell along the ray and composite them to the final image. Pixel (a) of figure 2 shows values used by a ray casting algorithm that samples at cell face intersections. The pink colored squares represent the sample points. The color contribution for pixel (a) is calculated using Eqn. 2 assuming a constant color value between the sampling interval (often the average of front and back sample colors is used) and composited to the screen (using Eqn. 1). Pixel (b) shows the values used by another ray casting algorithm that uses regular sampling. Pixels (c) and (d) together show yet another variation as used by volume slicing. Here, it is simulated as a regular sampling pattern with the first sample point starting from the plane closest to the screen. For all the sample points in the figure 2, either their data can be reconstructed and colors mapped from the transfer function, or their sample colors are interpolated using the pre-mapped colors of the eight cell corners.

In the context of scan converting polygons for projection-based DVR algorithms, we use the following to classify several popular DVR algorithms:

1. **Data model** – this distinguishes whether data is defined at vertices or at voxel centers. An associated interpolation or distance function is also specified for each.
2. **Interpolation value** – this distinguishes whether data values are reconstructed or color values are being interpolated at the polygon vertices or sample locations. That is, the color intensities (E) in Eqn. 2 are calculated either by

  - interpolating the data values and then evaluating the transfer function, or
  - evaluating the transfer function first, then interpolating the colors.
3. **Scan conversion procedure** – this specifies whether polygon or volume scan conversion is used to render the polygonal decompositions of volume cells. In polygon scan conversion, a software Gouraud shading is used to render polygons.
4. **Sampling strategy** – this distinguishes whether samples are taken regularly or only at cell face intersections, and how they should be distributed throughout the entire volume data.

Based on this four level classification scheme, we can identify several DVR algorithms that can be simulated as projection-based algorithms. This classification is not meant to be exhaustive but simply illustrates how different DVR algorithms can be viewed in terms of their variants. For complete specifications of DVR algorithms, more detailed rendering parameters within each criterion should be specified. Used in this manner, Table 1 shows how some algorithms are distinguished by their data model, interpolation value, scan conversion procedure, and sampling pattern.

Data Model	Interpolation Values	Scan conversion	Sampling Pattern	DVR Algorithm
Cell model with Tri-linear Interpolation	data or color	volume	regular	ray casting
		volume	cell face	ray casting
	color	volume	regular	volume texture
		volume	cell face	shear-warp
	color	polygon	cell face	coherent projection
Voxel model with distance function	color	polygon	irregular (depends on distance function)	splatting

Table 1. Illustration of how different DVR algorithms can be expressed in terms of projection-based algorithm using different combinations of the data model, value being interpolated, scan conversion type, and sampling pattern.

The data model in column (1) comes with either an interpolation function or distance function. When the data model assumes values are defined at vertices (*cell model*), an interpolation function is often used. In such situations, tri-linear interpolation seems to be the interpolation method of choice in many DVR implementations. However, other possibilities include higher order interpolations or adaptive reconstruction [7]. When the data model assumes values are defined at voxel centers (*voxel model*), a distance function is often used to model how data vary within the confines of the voxel. Other distance functions, as well as simpler voxel modeling using nearest neighbors, can also be incorporated.

The simulations of various DVR algorithms that we just described have varying degrees of accuracy. More precise specifications need to be made if an exact simulation is desired. For example, it is important to note that different methods of polygonaliza-

tion may lead to different looking images (see Figure 3, color plate 1). Thus, the complete polygonalization policy must also be specified if a faithful simulation is desired. Another thing to note is that projection-based algorithms often rely on hardware scan conversion of the polygons. Therefore, it is also possible to notice some differences, especially along the boundaries of projected polygons.

We verified our approach and implementation with a renderer called *mdh* [17, 18] which has multiple choices of algorithms. We made the rendering parameters of our simulation as identical as possible to those for the algorithms in *mdh* and made sure that the differences in the intermediate rendering information are within a given tolerance. That is, we took differences in colors, locations and data values for each front and back vertices of the projected polygons of all volume cells between our simulations and the projection algorithm of *mdh* and made sure that the differences are negligible (less than 10^{-7} in scale of 0 to 1.0 for each color channel).

5 Metrics for Projection-based DVR Comparisons

In this section, we present a set of data level metrics derived from our projection algorithm basis and proposed for comparing DVR algorithm simulations. These metrics should reveal information about the volume data (or color intensities) as well as the behavior of DVR algorithms. The idea is to identify differences in algorithms that may not be revealed from image level metrics alone. Note that there are numerous other useful intermediate information (e.g. gradient and normal calculations, and surface classification) that can also be collected and used as metrics.

- **Threshold-based Metrics**
  The following metrics are obtained to examine the behavior of a DVR algorithm for a given threshold condition. Each metric is measured at each pixel when the accumulated (or sample) color components reaches the threshold condition. While opacities are often used for specifying threshold levels, other color components can also be used. In our current implementation, a user can give a threshold condition that combines color and opacity threshold values.
  1. **Number of cells**
     Different algorithms use different rendering parameters and thus each algorithm may require a different *number of cells* to satisfy the given threshold condition.
  2. **Distances**
     It is useful to measure the distance traveled into the volume before reaching a specified threshold condition at each pixel. Distance can be measured from the user's eye position (*eye distance*) or from the bounding box of the data volume in the viewing direction (*volume distance*).
- **Contribution Metrics**
  The following metrics measure the contribution of each cell to the final rendered image. The user specifies a pixel in the image to probe, then metrics are measured and visualized to show which cells contributed to the selected pixel and by how much. These metrics can be used with or without specifying an opacity or color

threshold condition. Contribution metrics for each cell may either be *absolute* or *additive*. Absolute contribution is the amount contributed by a data cell to a pixel as if there is nothing between the cell and the image plane. Additive contribution is the actual amount of contribution by a data cell to a pixel because its absolute contribution is attenuated by the accumulated opacity so far. Looking at the front-to-back composition equation,

$$C_{acc,new} = C_{acc,old} + (1 - \alpha_{acc,old})C_{cell,i} \tag{3}$$

where $C_{acc,new}$ is the new color intensity after composition, $C_{acc,old}$ is the accumulated color before composition with the ith cell, $C_{cell,i}$ is the color contribution by the ith cell, and $\alpha_{acc,old}$ is the opacity component of $C_{acc,old}$. The *absolute* contribution metric uses the term $C_{cell,i}$. On the other hand, the *additive* contribution metric uses the term $(1 - \alpha_{acc,old})C_{cell,i}$.

1. **Pixel Probe**
   This measures the amount of color intensity contributed by each cell to the pixel being probed.
2. **Cell Probe**
   This is similar to the pixel probe but shows other information (e.g. averages, minimum, maximum, and standard deviation) by the contributing cells to the target pixel. These statistics are collected based on how each data cell contributed to the pixels in the final image. For example, when the contribution from each cell is distributed unevenly across several pixels, a measure of spread can be calculated for that cell.

– **Data Probe**
Similar to cell probe, except the user selects a particular data cell and is shown the contribution made by that cell on the different pixels of the DVR image. Note that this is different from the projection filter.

– **Difference Metrics**
While *threshold metrics* and *contribution metrics* probe how each algorithm individually behaves, differences of these metrics can show where and how two or more algorithms differ. For example, in addition to the difference and statistical measures (e.g. average, minimum, maximum, and standard deviations) between two algorithms, differences of the *cell probe* metric includes the correlation of color intensities generated by the two algorithms for the pixels that are covered by the selected cell.

6 Examples

In this section, we show some examples of applying our comparative visualization methods and discuss the applicability of our metrics. Figure 4 (color plate 1) shows volume rendered images generated by two different algorithms. Both algorithms render a 64^3 Hipip (High Potential Iron Protein) volume data. The image (a) is generated by ray casting (simulation with our base algorithm) with sampling and reconstruction of data at the cell faces. The image (b) shows an image generated using a polygon projection algorithm. The image in (c) shows the absolute differences between image (a) and

image (b). Color intensities in the difference image (c) are magnified to show the difference clearly (each color intensity is multiplied by 10). This image based comparison method can show location and magnitudes of differences but not much more.

Figure 5 (color plate 2) demonstrates how our data level metrics can provide more insight. Image (a) shows the colormapped visualization of the *number of cells* needed to reach an opacity level of 0.11 using the ray casting algorithm that generated image (a) of Figure 4. It shows that the regions around the red and blue molecules require more number of cells to be examined before reaching the given opacity. Image (b) shows the number of cells needed to reach an opacity level of 0.2 using the polygon projection algorithm that produced image (b) of Figure 4. Image (c) shows differences in the number of cells to reach opacity 0.15 between the two algorithms. It shows a higher difference near the boundary regions of the red and blue molecules. Aside from the opacity threshold, users can also try threshold conditions composed of other color channels. Users can confirm that one source for the differences in Figure 4 is due to the different number of cells used by each algorithm. The users can further compare algorithms using metrics such as *pixel probe* for specific pixels of interests. Images (d) and (e) of Figure 5 show absolute and additive color contributions from all the cells contributing to a specific pixel. In particular, the data cells that contributed to the red component of the selected pixel are highlighted. The absolute contributions in (d) show that the red intensities composited were prominently higher in a small region in the volume data, but the additive contributions in (e) are more widely spread. The yellow arrows show the viewing directions and the front-to-back traversal of the algorithm.

Figure 6 (color plate 2) illustrates a hypothetical case study using our metrics. The two DVR images are generated using our scanline simulation of a ray-based DVR method, but with different transfer functions. The volume data is from a CAT scan of a human head. In column (a), the location of a hypothetical tumor in the brain is identified to be within the box region. The upper image shows the tumor but the lower image does not show it. The visualizations in column (b) show the *volume distance* metric associated with each image. The opacity threshold is set to 0.31 in both cases. Black color is assigned to pixels where the opacity does not reach the opacity threshold. In the lower image of column (b), near the region of interest, an almost flat, blue wall indicates that the pixels accumulated enough opacity without traveling through many layers of data cells. On the other hand, the upper image of column (b) shows an almost uniformly black region where the blue wall used to be. With the exception of the tumor, the region around it did not produced sufficient opacity. This tool is especially useful if the person understands how the DVR algorithms work and how they are affected by the different rendering parameters. However, other metrics can also help users to understand and verify the rendering methods. Columns (c) and (d) show data level analyses using contribution-based metrics. A pixel of the tumor is first selected to be examined. In column (c), the absolute amount of opacity contributions are visualized for all sub-volume cells that contribute to the given pixel. In column (d), the additive opacity values are shown. The visualizations show that the transfer function for the lower row produced opacity values that are too high to reach the tumor. On the other hand, on the top row, both absolute and additive contributions are highest where the hypothetical tumor is located. From the comparisons of the final rendered images alone,

it may not be obvious whether the opacity for brain tissue is set too high or the data range for the tumor tissues is not set properly to be detected by the transfer function. This case demonstrates that our comparative visualization techniques can provide more insight into why two rendered images are different and how rendering parameters (such as transfer function) affect the resulting images.

7 Conclusion

We presented a new data level framework that solves difficulties of comparing different DVR algorithms and demonstrated how different DVR algorithms can be simulated using the projection algorithm basis. From this base or reference algorithm, several new data level comparison metrics were then presented that highlight different aspects of the volume data and the DVR algorithms. These metrics, used individually and in combination, provide additional information beyond how two different DVR images are different – they seek to provide clues as to *why* and *how* the two images may be different. We also gave examples of using our metrics and a hypothetical case study that demonstrates the applicabilities of the metrics. Our new methodology overcomes the limitations of conventional image level comparisons and helps us to perform more in-depth comparisons which are closely related to the purpose of a particular visualization application as well as general rendering quality comparisons. These results are important to scientists to help them interpret different visualization results objectively.

Acknowledgements

We would like to thank Craig Wittenbrink and Suresh Lodha for collaborative work on uncertainty visualization and Sam Uselton for comments and feedback on image level comparisons. We would also like to thank Chang Sung Jeong and Carol Mullane for help with proofreading this paper. This projected is partially supported by NSF grant IRI-9423881, DARPA grant N66001-97-8900, NASA NCC2-5281, ONR grant N00014-96-1-0949, and LLNL/DOE grant B347879.

References

1. Ajeetkumar Gaddipatti, Raghu Machiraju, and Roni Yagel. Steering image generation with wavelet based perceptual metric. In *Computer Graphics Forum*. Eurographics, September 1997.
2. A. Globus and S. Uselton. Evaluation of visualization software. *Computer Graphics*, pages 41–44, May 1995.
3. Kwansik Kim and Alex Pang. Ray-based data level comparison of direct volume rendering algorithms. Technical Report UCSC-CRL-97-15, UCSC Computer Science Department, 1997.
4. Kwansik Kim and Alex Pang. Ray-based data level comparative visualization of direct volume rendering algorithms. In *to appear in Scientific Visualization, Dagstuhl Workshop Proceedings*. Springer, 1998.

5. David Laur and Pat Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. In *Computer Graphics (ACM Siggraph Proceedings)*, volume 25, pages 285–288, July 1991.
6. Marc Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, 9(3):245–261, July 1990.
7. R. Machiraju and R. Yagel. Reconstruction error characterization and control: A sampling theory approach. *IEEE Transactions on Visualization and Computer Graphics*, pages 364–378, December 1996.
8. Tom Malzbender. Fourier volume rendering. *ACM Transactions on Graphics*, 12(3):233–250, July 1993.
9. S. R. Marschner and R. J. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 100–107. ACM, October 1994.
10. Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, June 1995.
11. Torsten Moller, Raghu Machiraju, Klaus Mueller, and Roni Yagel. Classification and local error estimation of interpolation and derivative filters for volume rendering. In *Symposium on Volume Visualization*, pages 71–78, San Francisco, CA, October 1996. ACM/IEEE.
12. Hans-Georg Pagendarm and Frits H. Post. Comparative visualization – approaches and examples. In M. Gobel, H. Muller, and B. Urban, editors, *Visualization in Scientific Computing*, pages 95–108. Springer-Verlag, 1995.
13. P. Sabella. A rendering algorithm for visualizing 3D scalar fields. In *Computer Graphics*, pages 51–58, August 1988.
14. P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. In *1990 Workshop on Volume Visualization*, pages 63–70, San Diego, CA, December 1990.
15. Allen Van Gelder and Kwansik Kim. Direct volume rendering with shading via 3D textures. In *ACM/IEEE Symposium on Volume Visualization*, pages 22–30, San Francisco, CA, October 1996.
16. L. Westover. Footprint evaluation for volume rendering. In *Computer Graphics*, pages 367–376, August 1990.
17. Jane Wilhelms and Allen Van Gelder. A coherent projection approach for direct volume rendering. In *Proceedings of SIGGRAPH 91*, pages 275–284, 1991.
18. Jane Wilhelms and Allen Van Gelder. Multi-dimensional trees for controlled volume rendering and compression. In *Proceedings of the Symposium on Volume Visualization*, pages 27–34, color plate 125, Washington, D.C., 1994.
19. Jane Wilhelms, Allen Van Gelder, Paul Tarantino, and Jonathan Gibbs. Hierarchical and parallelizable direct volume rendering for irregular and multiple grids. In *Proceedings of IEEE Visualization '96*, pages 57–64, 1996.
20. Peter L. Williams, Nelson L. Max, and Clifford M. Stein. A high accuracy volume renderer for unstructured data. Technical Report UCRL-JC-126942, Lawrence Livermore National Laboratories, September 1997.
21. Peter L. Williams and Samuel P. Uselton. Foundations for measuring volume rendering quality. Technical Report NAS-96-021, NASA Numerical Aerospace Simulation, 1996.
22. Craig M. Wittenbrink, Alex T. Pang, and Suresh Lodha. Verity visualization: Visual mappings. Technical Report UCSC-CRL-95-48, Univ. of Cal. Santa Cruz, 1995.

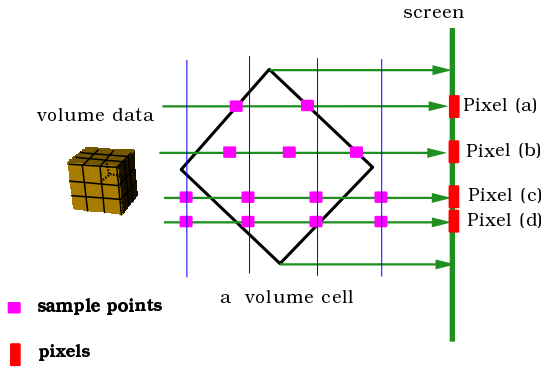

Fig. 2. Simulations of image based algorithms using our projection base algorithms. The black square and the green vertical line illustrates a volume cell and its projection to the screen. For each pixel, the color contributions are computed by taking sample points and compositing their values along the viewing direction. The blue vertical lines indicate a volume slicing plane shared by pixels (c) and (d).

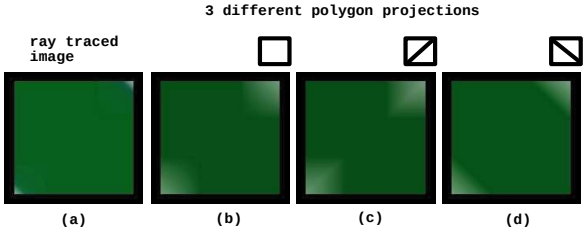

Fig. 3. Ray casting and projection algorithms with different polygonalization. The volume data is 4^3 with uniform values except at the two corners. The viewing direction is orthogonal to the front face. Above image (b), (c) and (d), we show the three types of polygonalization: (b) a square, (c) and (d) two different triangulations of a square. The images show that different triangulations have different effects on Gouraud shading.

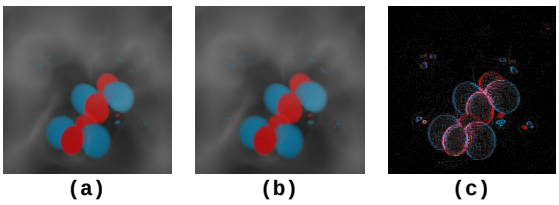

Fig. 4. Volume rendering and image level comparisons of Hipip. Images (a) and (b) are generated using our simulations of (a) ray casting with data samplings and reconstructions at cell faces, and (b) polygon projection such as coherent projection. Image (c) shows the absolute differences between (a) and (b). MSE (Mean Square Errors) and RMSE (Root Mean Square Errors) of actual difference intensities are (4.882487, 1.574969, 2.983297) and (2.209635, 1.254978, 1.727222) for each red, green and blue channel respectively. All image sizes are 256×256.

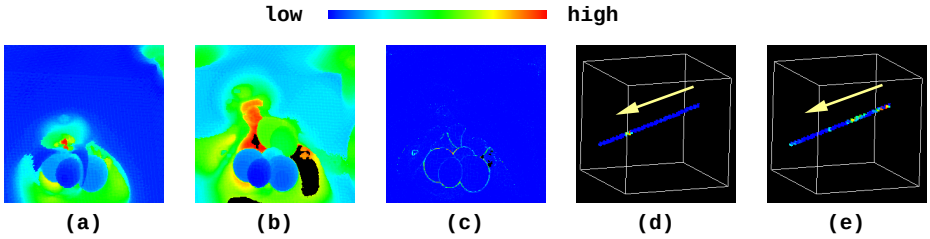

Fig. 5. Data level comparisons of algorithms used in Figure 4. (a) shows the number of cells needed to reach opacity of 0.11 with with ray casting simulation, (b) shows the number of cells needed to reach opacity of 0.21 with the polygon projection algorithm, (c) shows differences in the number of cells needed to reach opacity of 0.15 for both algorithms. Colors are used to indicate relative values of the metric. Black indicates regions that did not reach the threshold. The pixel probe visualizations shows the absolute (d) and additive (e) red intensity contributions of data from the volume to the pixel marked by the cross hair.

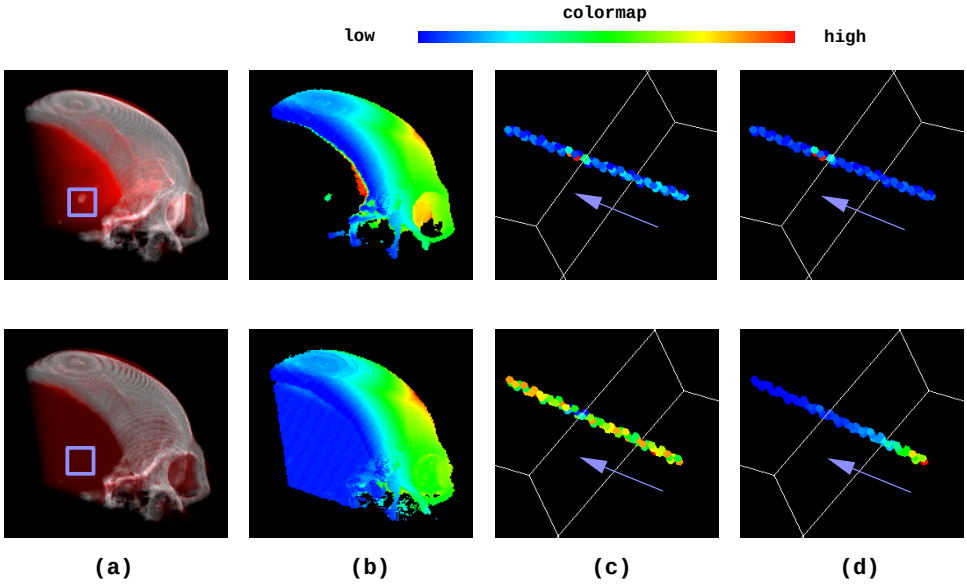

Fig. 6. Case study illustrating a hypothetical tumor and how the *volume distance* metric (b) and the *pixel probe* (c) and (d) can shed more insight. Visualizations of column (c) and (d) show differences in absolute and additive color contributions from volume cells.