# Ray Casting Image Quality of Different Solutions

Ingmar Bitter
Computer Systems Laboratory
HP Laboratories Palo Alto
HPL-96-130(R.1)
March 17, 2006*

ray casting, volume rendering, volume visualization, resampling, classification, accumulation buffer, color interpolation

Ray casting is one of the standard approaches for rendering images from volumetric data sets. There are multiple ways to map ray casting into an algorithmic solution. The goal of this paper is to show the different impacts in image quality due to those different realizations of the ray casting technique.

* Internal Accession Date Only
© Copyright 2006 Hewlett-Packard Development Company, L.P.

# *Ray Casting Image Quality of Different Algorithmic Solutions*

*Ingmar Bitter*

August 30, 1996

## Abstract

Ray casting is one of the standard approaches for rendering images from volumetric data sets. There are multiple ways to map ray casting into an algorithmic solution. The goal of this paper is to show the different impacts in image quality due to those different realizations of the ray casting technique.

## 1 Introduction

*Volume visualization* is a method of extracting information from volumetric datasets through the use of interactive graphics and imaging, and is concerned with the representation, manipulation and rendering of these datasets [Kau90]. Most commonly the data represents a continuous 3D function sampled on a regular, rectilinear 3D grid of volume elements called *voxels*. There are many different approaches to render images from these volume datasets. One technique is to extract an intermediate polygonal surface representation from the volume data and then project those polygons to the screen (e.g. Marching Cubes in [LC87] ). Another one is to apply thresholding to values at each voxel. The remaining voxels are then treated as small 6 faced cubes and projected to the screen [HL79]. Splatting convolves each voxel with a 3D filter and accumulates the voxels contribution on the image plane [Wes91]. A more physically based approach is to simulate how light is reflected and attenuated along the line of sight. In ray casting [Lev88] for every pixel in the screen image the sight ray is followed through the volume. At all intersection with at least partially visible voxels the local surface and its normal are approximated to use in an illumination equation which returns the brightness of that sample location. The final pixel color is the accumulation of all shaded samples along the ray. In volumetric ray tracing [SK94, Sob94] the ray casting approach is extended to also include the effect of the sight rays being reflected off the approximated surfaces. Finally there is the light field [LH96] or lumigraph [GGSC96] approach in which the 4D light flux field on a sphere surrounding the volume is computed as an intermediate representation. After that images are rendered as 2D resampled cuts from the 4D light flux field. To generate the light flux field a set of images is needed which can be generated using any of the previous approaches.

The highest quality images are produced with ray casting and volumetric ray tracing. As the second one is orders of magnitude more expensive to compute and the three dimensional structure of the data is equally well represented in both techniques ray casting is often the technique of choice.

To implement ray casting one has to discretize the rays traversing the volume. This is done by point samples along the ray (see figure 1). In the easiest case the rays are parallel, aligned with one of the major axes and the ray sample locations lie on the volume grid positions. Even in this case the data sample has to be mapped to color and opacity of the ray sample. The opacities could be preassigned due to a previous segmentation process. Otherwise a function taking the voxel value and the local gradient magnitude as parameters are used to assign the opacity. This is called classification. The color for each voxel is normally taken from a function taking only the voxel value as a parameter. This color is then attenuated by the result of the illumination equation which computes how much light from the present light source(s) is reflected from the local surface into the direction of the sight ray. This is called shading. Those shaded samples are then composited along the ray to yield the final pixel color.

Viewing from an arbitrary angle causes the ray samples to fall between the voxel grid positions (see figure 1). In order to compute the shaded samples in this case one can either interpolate the neighboring voxel values to get an approximation of the data value at the sample location and then do the classification and color assignment or do the color and opacity assignments at the voxel grid positions followed by an interpolation of those values to the ray sample position (see figure 2). The classification first approach which is used in Levoy's [Lev88] volume rendering paper has the advantage that with fixed light sources classification and shading can be precomputed while in the resampling first approach which is used in the Cube4 architecture [PWK95] classification and shading have to be recomputed for each new viewing direction.

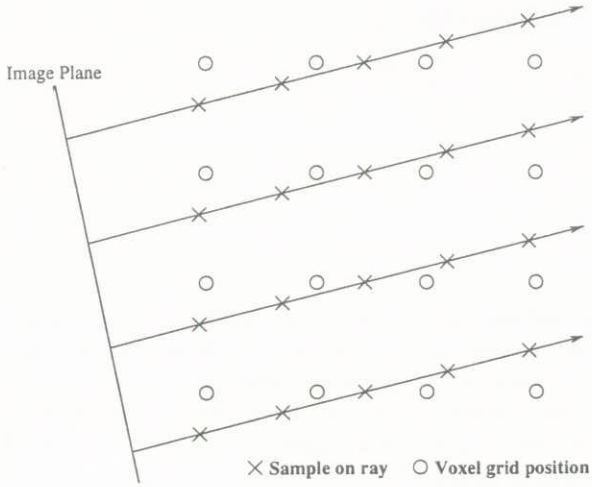


Figure 1: Ray casting in 2D: The data is present at the voxel grid, but needed at the sample locations along the rays.

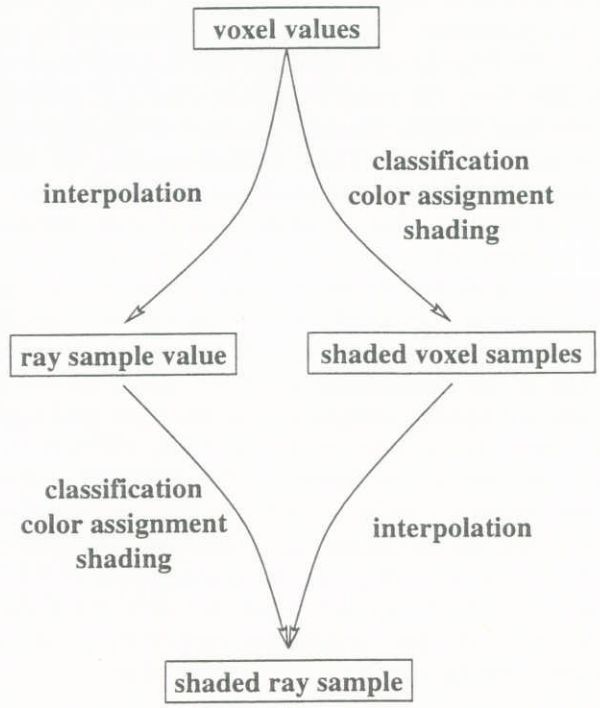


Figure 2: The two possible orders to compute the shaded ray samples.

# 2 Resampling vs. Classification order

## 2.1 Resampling rate

The higher the sampling rate along the ray, the higher the resulting image quality. From a certain sampling rate on, a further increase doesn't introduce noticeable changes in the image any more. This maximum frequency depends on the dataset and the classification function. The worst case occurs, when neighboring voxels differ greatly in their density values and the classification function is very steep. (This means high frequencies in the volume data and the classification function.)

The possible maximum increase in image quality depends heavily on how the resampling is done. Figure 3 shows a dataset in which the real world data consists of tilted layers of continuously increasing density (including a layer of density 70). During the process of resampling this data is discretised resulting in density values only at the voxel positions (in this case there are vo voxels of density exactly 70). When a classification function is used that is supposed to show the surfaces of density 70 very sharply this slice of density 70 might be missed completely. In this example all that is necessary is that the width of the nonzero opacity assigning interval $I_o$ of the classification function is less than 2. Assuming it is set to [69.2, 70.8] classifying before resampling assigns all voxels to be invisible and the samples along will be invisible, too. If the resampling is done before the classification, then the interpolated density values might be – as in this example – closer to the desired range. So the two samples in the big circles would be classified to be visible and the slice of density 70 would not be missed. If the interval $I_o$ would be even smaller and both approaches would yield only invisible samples, but in the case of resampling first one can just increase the sampling frequency accordingly. So for a interval $I_o$ of half the width twice the sampling frequency will show the slice of density 70 again. Increasing the sampling density in the classification first approach doesn't increase the visibility of very thin structures.

A worst case test sequence of exponentially increasing sampling rates from 0.5 to 32 samples per voxel showed:

classification ⇒ resampling: 1 sample per voxel or more always rendered the same voxels visible (no skipping) but had still many shading artifacts and the image looked fuzzy. From 6 or more samples per voxel on the shading was smooth and the image was sharp. $f_{max} = 6$ See figure 4.

resampling ⇒ classification: 5 samples per voxel or more showed the correct connectivity. From 20 samples per voxel on the shading was smooth. But up to 32 samples per voxel there where still small differences in images generated with slightly different sample rates. $f_{max} = 20$ See figure 5.

Using a thin, rectangular window as a classification function (which has an infinite extend in the frequency spectrum) with the intention to visualize an iso surface causes these artifacts:

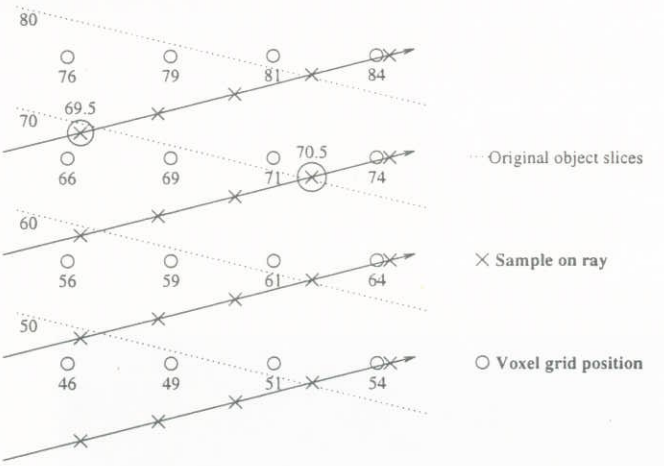


Figure 3: A dataset in which the real world data consists of tilted layers of continuously increasing density. During the process of resampling this data is discretised resulting in density values only at the voxel positions. The resampling process tries to reconstruct the information between the grid locations.

classification ⇒ resampling: The iso surface appears to have many holes or in worse cases consists of a set of disconnected voxels. This happens if the width of the classification rectangle is smaller than the difference in adjacent voxel values, such that the complete, accepting interval is missed quite often on the voxel grid locations. See figure 6.

resampling ⇒ classification: The iso surface appears similar to height contour lines. The areas between the connected visible surface lines correspond to the pixels in which the classifying interval fits into the gaps between the samples along the ray. Doubling the sampling rate usually halves the size of the gaps. So with enough samples per voxel, the gap size can be pushed below pixel size and the resulting iso surface appears smooth and connected. See figure 7.

## 2.2 View Dependency

classification ⇒ resampling: This order implies view independent classification results. Rotations around any axis appear as smooth movements of a consistent image (one always sees the same voxels). See figure 8.

resampling ⇒ classification: This order implies that for each step in a rotation sequence new sample points along the new rays have to be classified. A classification function with high frequencies emphasizing very thin structures will cause those structure surfaces to flicker during the rotation. If the structure is thinner than the sampling distance, then for some view directions there will be one sample in the structure

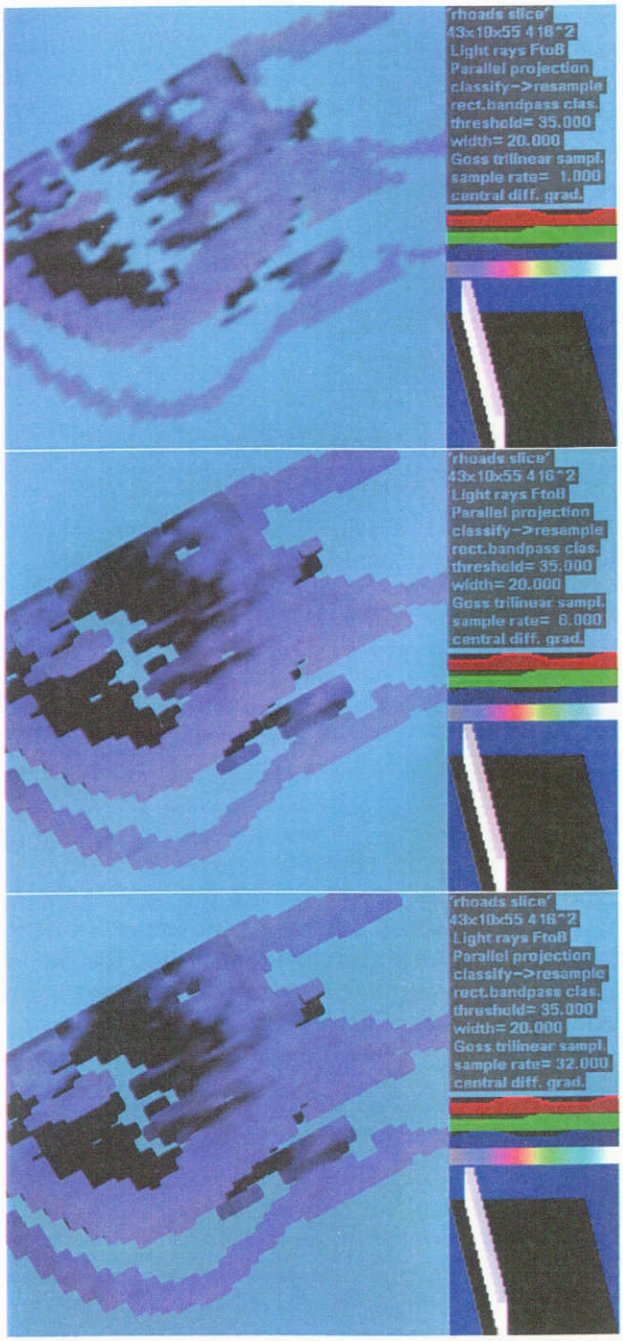


Figure 4: Classifying before resampling always renders the same voxels visible but improves shading and picture clarity with increasing sampling rate.

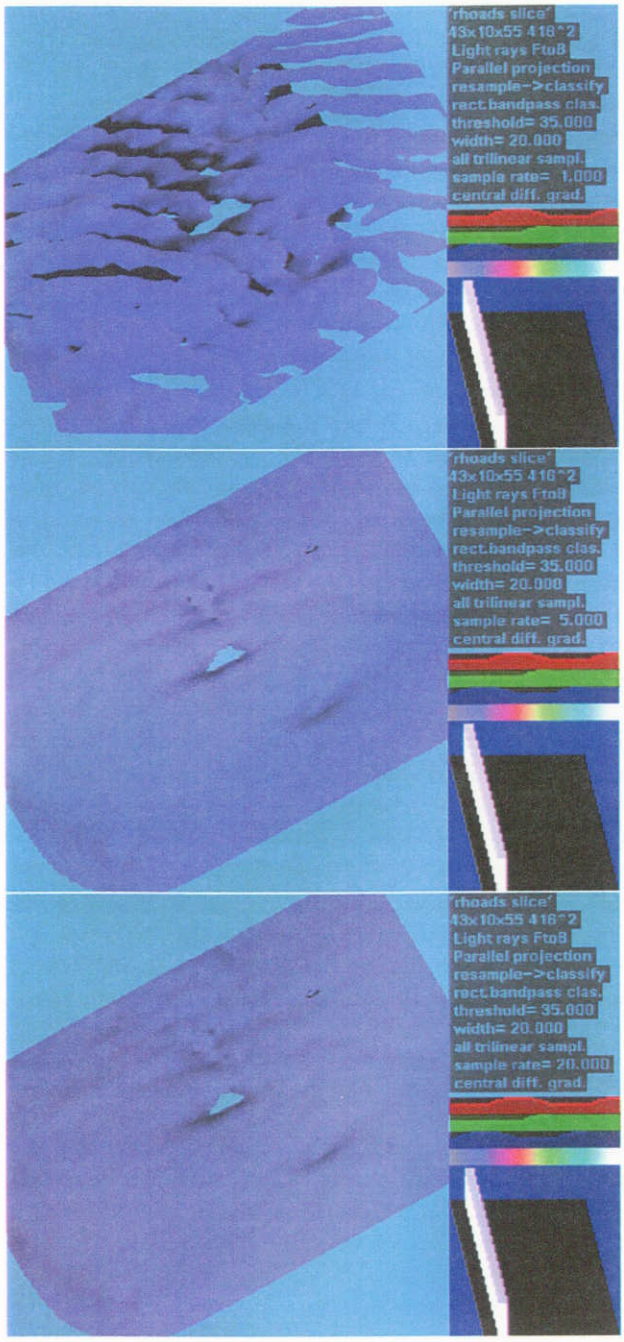


Figure 5: Resampling before classifying results in connected surface strings which might have gaps between them. With higher sampling rates the gaps vanish, but even at very high sampling rates a further increase in the sampling rate still results in visible differences on the object edges.

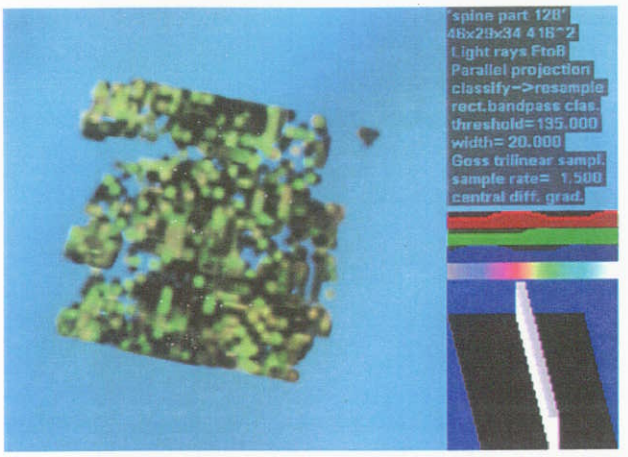


Figure 6: Classifying before resampling together with a very sharp classification function can create a disconnected set of visible voxels.

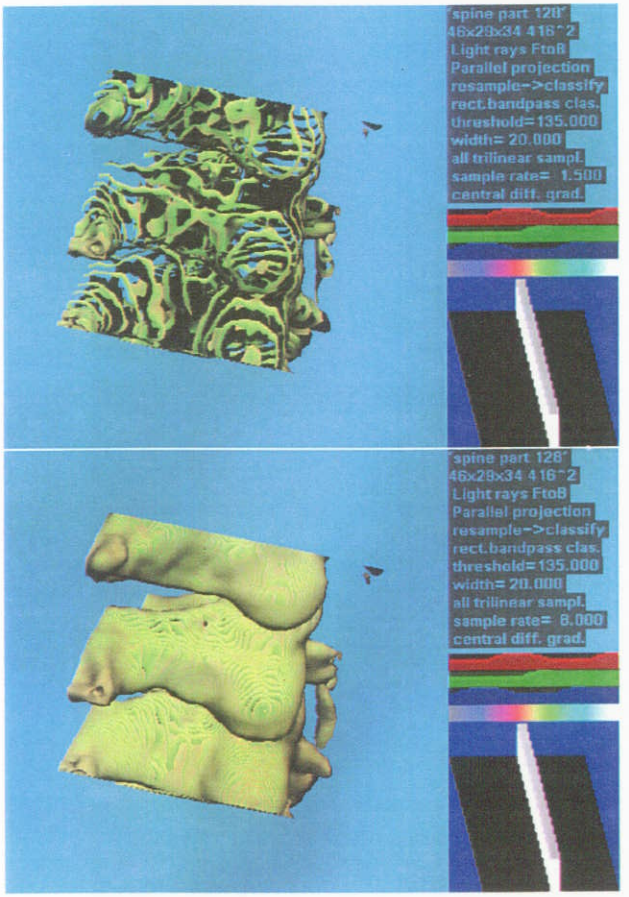


Figure 7: Resampling before classifying together with a very sharp classification function can create gaps between connected surface strings. The gaps shrink with increasing sampling rate.

and for other directions there will be one before and one behind, thus missing the structure. See figure 9.

The comparison between both versions shows that the resampling first order conveys more information about the three dimensional structure. On the other hand, it introduces flickering if there are structures thiner than the current sampling distance, but in these cases one still gets a better idea of the connectivity of these thin structures than in the classify first approach.


Figure 8: Classifying before resampling creates the same object appearance independent of the viewing angle (here changed by 10 degrees).

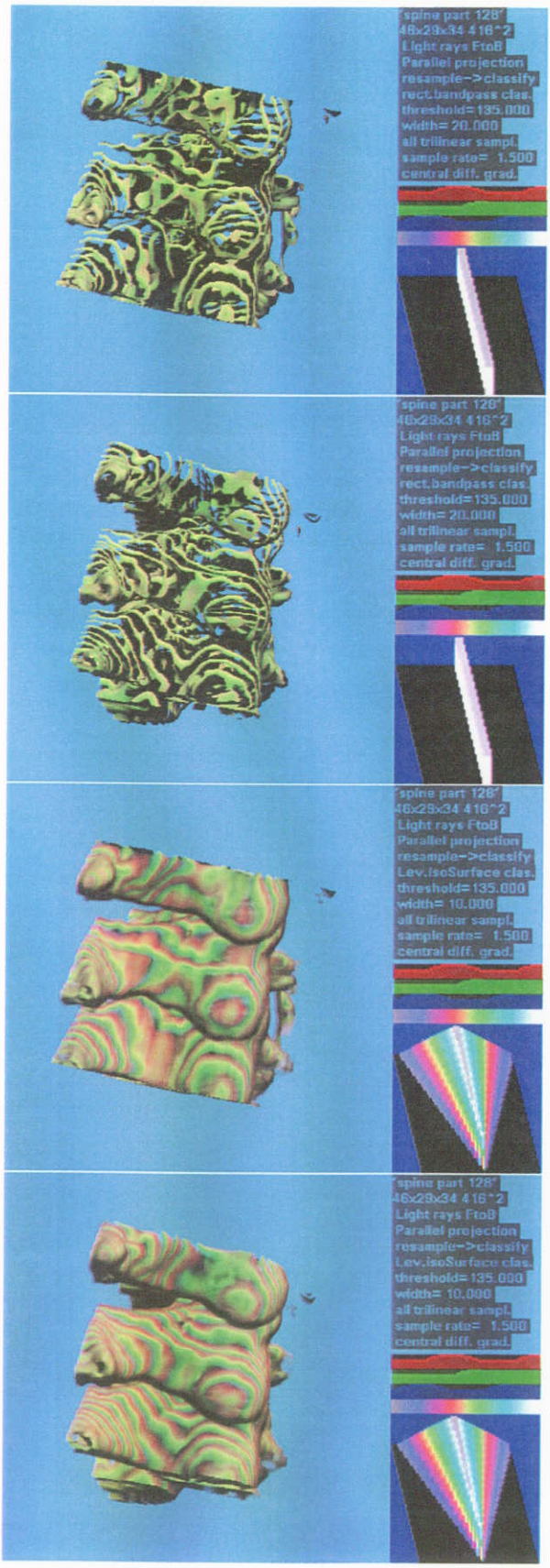


Figure 9: Resampling before classifying might result in a change of the appearance of the object, depending on the viewing angle (here changed by 10 degrees).

# 3 Accumulation Buffer Depth

The last step in the rendering pipeline is the accumulation of the shaded samples yielding the final pixel value. If the intermediate results after adding the contribution of a new sample are stored in a buffer for all rays in the image, then the buffer might get quite big if double precision is used. This section describes how many bits are needed, if a fix point format is chosen.

The two possible accumulation approaches are back-to-front and front-to-back compositing.

## 3.1 back-to-front compositing

The algorithm to do the compositing back-to-front is :

$$\text{for } k = n \ldots 1$$
$$C = C \cdot (1 - \alpha_k) + c_k \cdot \alpha_k$$

$c_k$ = *color of sample k*
$\alpha_k$ = *opacity of sample k*
$C$ = *accumulated color*

The worst case scenario here appears if all $\alpha_k = c_k =$ .0000 0001 (binary fix point values). Then $c_k \cdot \alpha_k =$ .0000 0000 0000 0001, thus all intermediate results have to be computed with precision $p_i$ which must be at least twice the precision $p_\alpha = p(\alpha_k) = p(c_k)$. In the above example of $p_\alpha = 8$ bits accumulating $n = 2^8 = 256$ samples or more lets the least significant bit of the $p_i$ precision intermediate results sum up to a change in the least significant bit in the original $p_\alpha$ 8 bit range. Therefore the buffer for the intermediate results cannot allow any rounding to coarser precision than $p_i = 2 * p_\alpha$.

## 3.2 front-to-back compositing

The algorithm to do the compositing front-to-back is :

$$\text{for } k = 1 \ldots n$$
$$C = C + c_k \cdot \alpha_k \cdot (1 - A)$$
$$A = A + \alpha_k \cdot (1 - A)$$

$c_k$ = *color of sample k*
$\alpha_k$ = *opacity of sample k*
$C$ = *accumulated color*
$A$ = *accumulated opacity*

The complete summation results in the same analytic sum as in the back-to-front approach, therefore the same worst case applies.

## 3.3 sub & super sampling along a ray

The opacity classification of voxels is usually represents the factor of how much the intensity of the ray intersecting with the complete voxel is decreasing. To keep this decrease constant, even if there are multiple samples within a single voxel, the opacities have to be adjusted. Bentum [Ben95] derives the following approximation for the adjustment assuming peace wise homogeneous materials and sampling frequency $f$ :

$$\alpha' = 1 - \sqrt[f]{1 - \alpha}$$

For super sampling the worst case appears if an $\alpha_k = \varepsilon =$ .0000 0001 opacity has to be remaped to an $\alpha' > 0$. To guarantee that $\alpha'$ is nonzero when $\alpha_k$ is nonzero the number of bits $b_{f,\varepsilon}$ for $\alpha'$ has to be at least $b_{f,\varepsilon} = \lceil |\log_2(1 - \sqrt[f]{1 - \varepsilon})| \rceil$.

For sub sampling the worst case appears if an $\alpha_k = 1 - \varepsilon$ opacity has to be remaped to an $\alpha' < 1$. To guarantee that $\alpha' < 1$ when $\alpha_k < 1$ the number of bits $b_{f,\varepsilon}$ for $\alpha'$ has to be at least $b_{f,\varepsilon} = \lceil |\log_2(1 - \sqrt[f]{1 - [1 - (1 - \varepsilon)]})| \rceil = \lceil |\frac{1}{f} \log_2 \varepsilon| \rceil$.

## 3.4 light fog

Often 8 bits are used for r, g, b and opacity $\alpha$. In case that the intend is to visualize an object being partially occluded by light fog, than the fog representing voxels have to have very small opacities. Using only 8 bits 1/256 is the smallest nonzero opacity. In an $1024^3$ data set with uniform sampling or an $256^3$ data set with four samples per voxel there are 1024 samples which have to be accumulated. After each fog sample the remaining ray intensity is reduced by a factor of 255/256. So after 1024 samples the ray intensity is down to $\frac{256}{256}^{1024} = 0.02$, thus the light fog will totally dominate the rendering. Using 16 bits for the opacities improves the remaining opacity after 1024 samples through very thin fog to be $1 - \frac{1}{256^2}^{1024} = 0.98$. So to achieve the desired result, one has to either use more bits for r, g, b and $\alpha$ or has to interpret the bits differently. One idea is to handle the opacities as if they where the square roots of the real opacities. So then can be stored at 8 bits and then during computations the 16 bit square is used. This nonlinear mapping automatically expands from 8 bits to 16 bits and gives highest resolution in the near zero region where small changes can make big differences.

## 3.5 compositing artifacts

Violating the above constraints has two kinds of consequences. If the classification function which assigns the individual opacities only generates very opage and completely transparent samples, then using limited precision will hardly affect the resulting image, as only one or two samples along a ray are actually contributing to the final color. The other extreme is when nearly samples along the ray have very small, but nonzero opacities. In this case using limited precision will result that all those samples get rounded to zero and incorrectly don't contribute to the final pixel color. So an object at the end of a foggy tunnel would appear sharp and bright instead of half hidden behind grey shade. See figure 13.

These effects are more obvious in the case of super sampling where small opacities need extra precision than in the case of sub sampling where the near complete opage samples need extra precision. In the former case the fog might disappear while in the later a sample making up

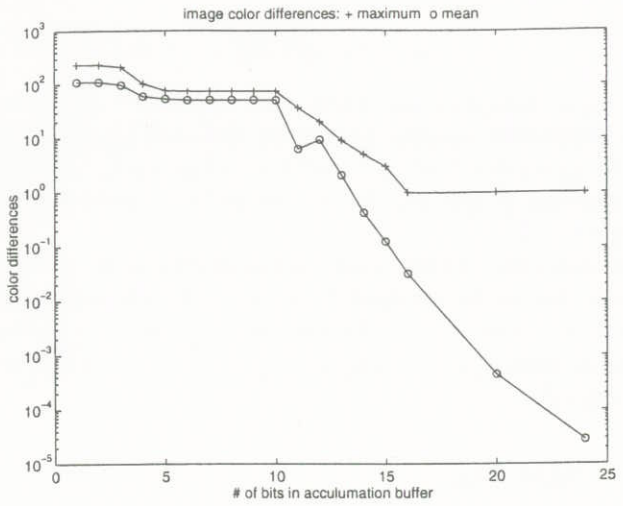


Figure 10: maximum and mean error dependence on accumulation buffer size. From 16 bits on the maximum error lies within the least significant bit and can be neglected.

more than 99% might make up 100% of the final pixel color.

## 3.6 experimental results

The following images show the differences between double precision accumulation and limited fix point accumulation buffers. The classification function is clipped to always stay inside the decimal interval [0.00···001,1], so most of the data appears as 0.00···001 opage fog.

Figure 11 shows the visual results of fog having an opacity equal to the last significant bit of varying fix point precision encodings such as 6,8,10,12,14 and 16 bits. One sees clearly that 8 bits are not enough. 12 to 14 bits give nice visual results.

In figure 12 and 13 the images on the left side are rendered using fog encoded at 12 bits precision and the limited accumulation buffer (the number before the word bits in the third line of parameter informations on each image stands for the number of bits used for that particular image accumulation buffer). The images on the right side are the difference images of the one left of it and the one using double precision. Up to 14 bits compositing buffer size the image quality improves noticeably — beyond 14 bits the errors are neglectable.

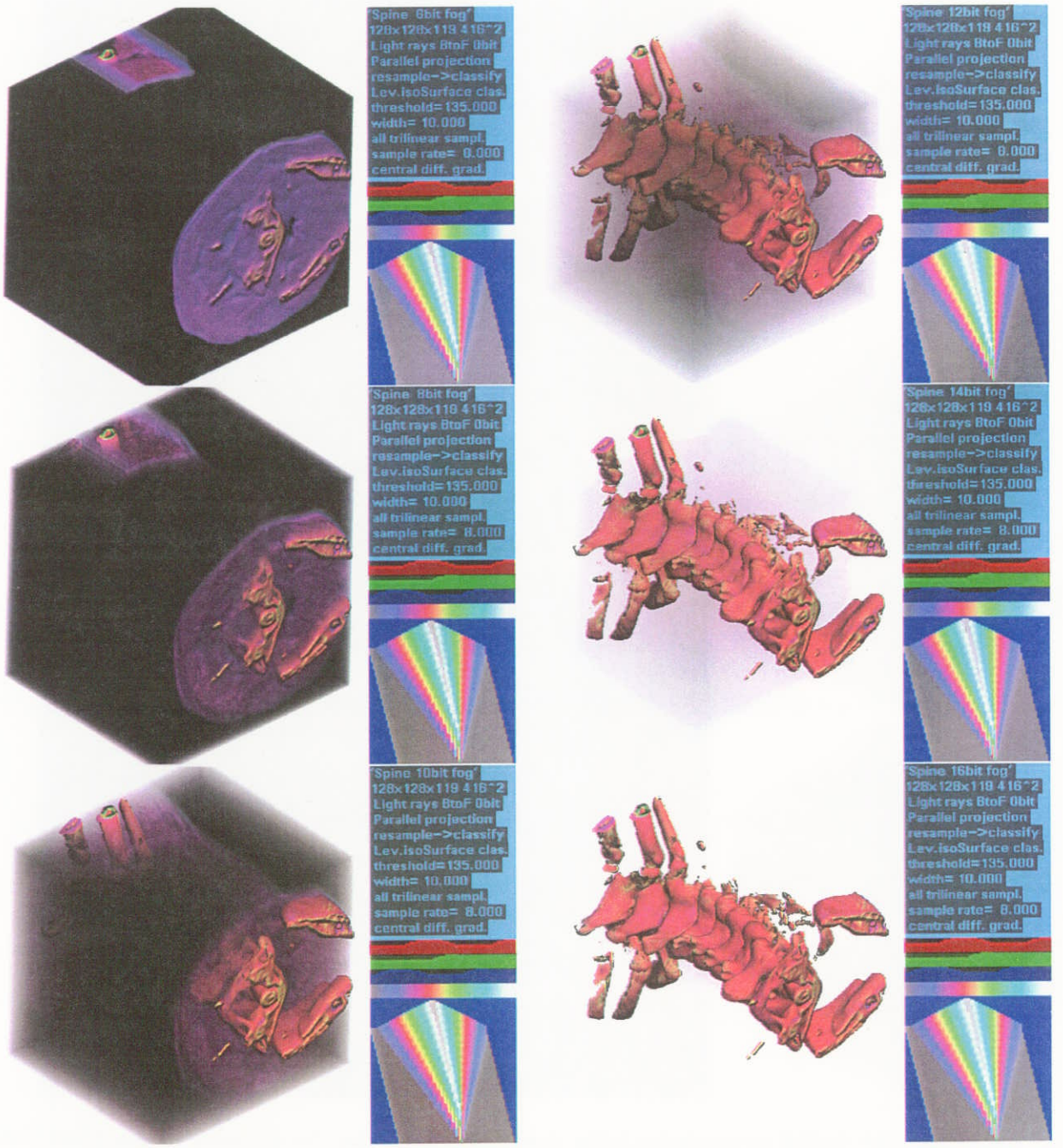


Figure 11: Representing Fog at 6,8,10,12,14 and 16 bits precision.

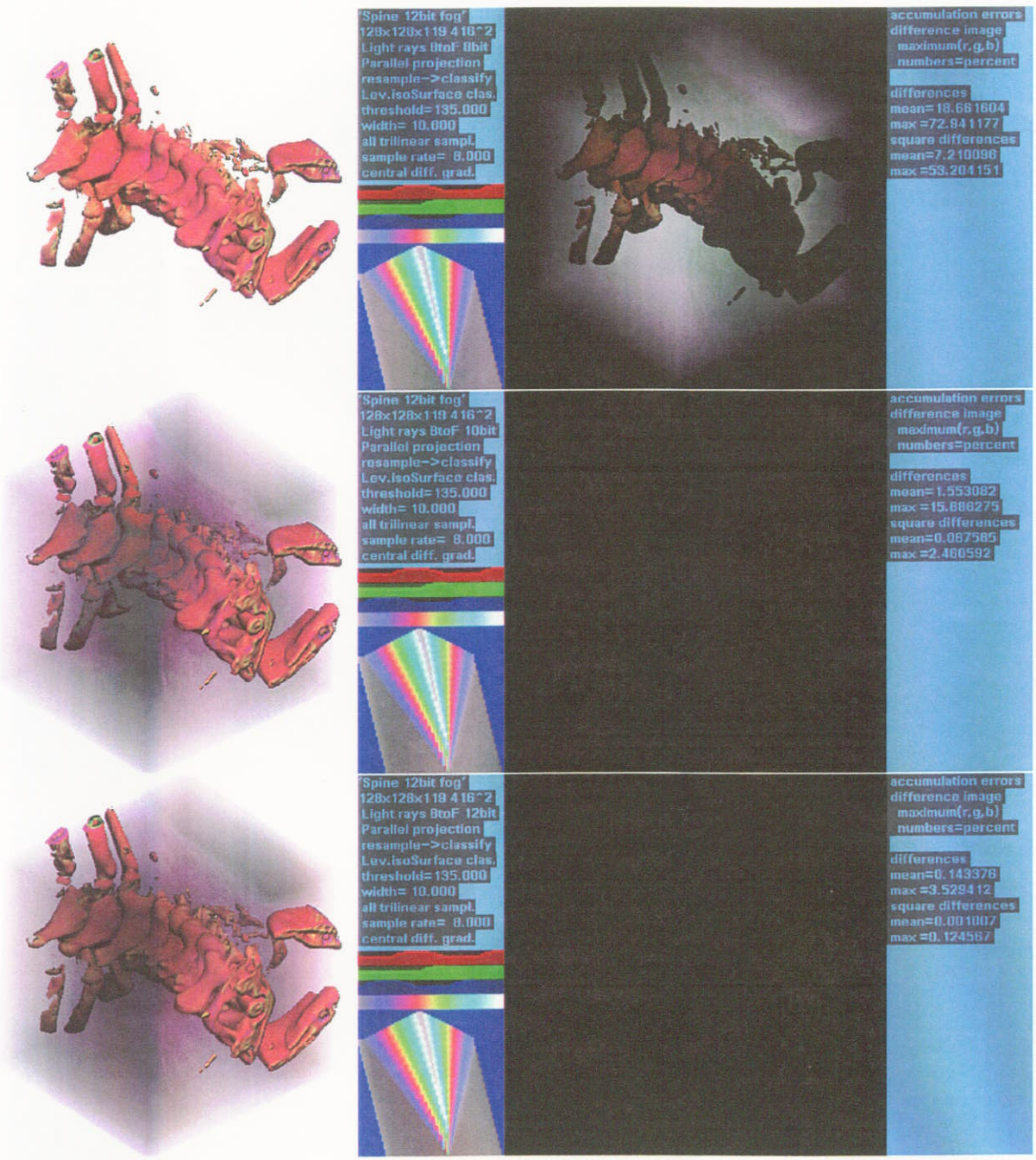


Figure 12: Renderings with limited accumulation buffer sizes 8, 10 and 12 and the difference images towards the double precision accumulation buffer. The images on the left side are rendered using the limited accumulation buffer and the number before the word bits in the third line of parameter informations stands for the number of bits used for that particular image. A zero there stands for double precision. The images on the right side are the difference images of the one left of it and the one using double precision.

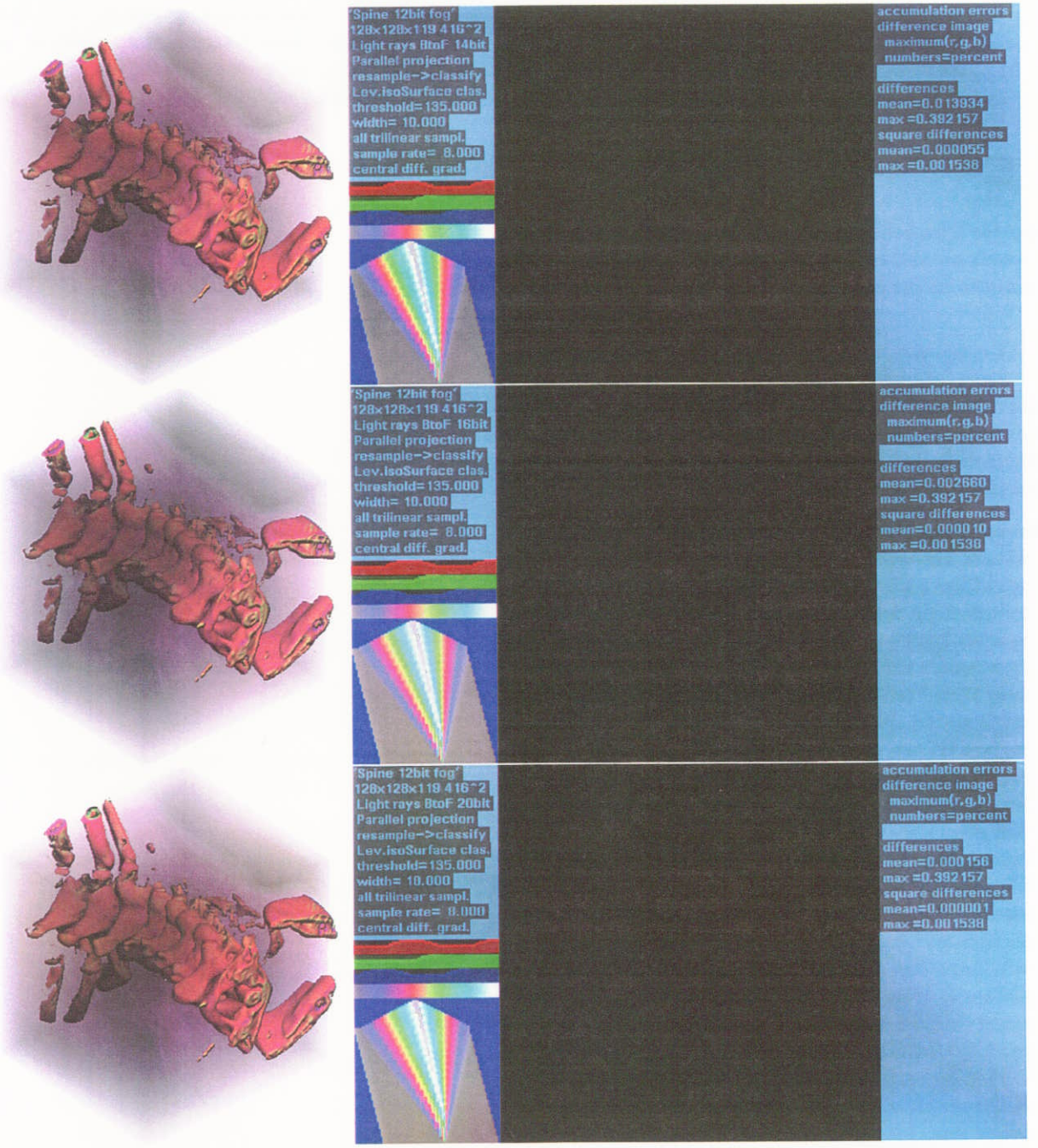


Figure 13: Renderings with limited accumulation buffer sizes 14, 16 and 20 and the difference images towards the double precision accumulation buffer. The images on the left side are rendered using the limited accumulation buffer and the number before the word bits in the third line of parameter informations stands for the number of bits used for that particular image. A zero there stands for double precision. The images on the right side are the difference images of the one left of it and the one using double precision.

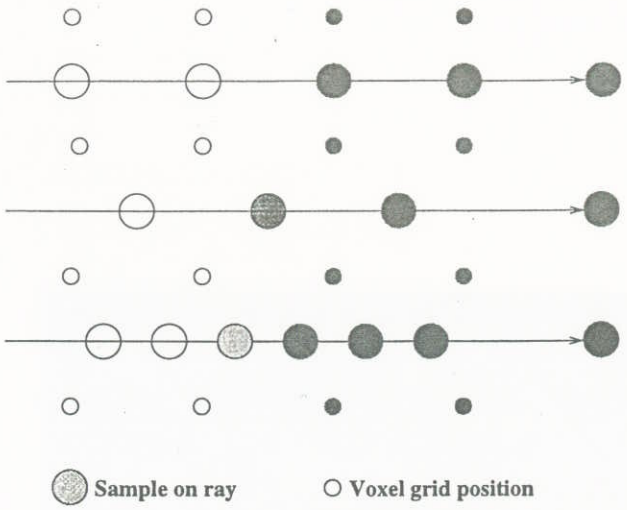


Figure 14: The Self Occlusion Effect: As Invisible classified voxels on the left contribute and incorrectly darken the samples on the border to the bright and visible voxels.

# 4 Goss vs. trilinear interpolation

In case that the classification ⇒ resampling approach and first order interpolation is chosen, there are at least two solutions to find the color and opacity values for the sample locations along each ray:

trilinear interpolation: This is the standard tradeoff winner between speed and quality. Using only multiplications and additions, a cascade of seven linear interpolations finds the approximations for colors and opacity between the voxel grid points. If the data and the classification function contain high frequencies, then "self occlusion" might appear. This effect refers to the situation in which some of the surrounding voxels are classified as outside (opacity 0, black color) and some are classified as inside (opacity 1, bright color) . (See figure 14.) The interpolation will generate an intermediate sample which has black mixed into the bright color. This effect is highly visible as this darkened sample is usually the first visible sample along the ray. Viewing from an arbitrary direction and looking at a flat surface, adjacent rays will have the darkened sample at slightly different distances from the real surface, therefore, the darkening will be worse for the samples a little further from the surface. In the complete image one can see darkening stripes running across the surface.
If the outside voxel classification resulted in a color other than black, then the self occlusion turns from darkening to color shifting. (See figure 15a).)

Goss interpolation: To overcome the self occlusion, Michael Goss suggested not to interpolate the colors, but to compute the weighted and normalized sum of colors where each adjacent grid color is weighted by the opacity at that location and the normalization is performed by dividing the resulting color sum by the sum of all adjacent opacities. The weighting of colors removes the influence of the outside classified voxels, thus avoiding the self occlusion. The normalization is needed to keep the ability to represent thin bright clouds. Without the normalization all nearly transparent objects would also be nearly black as the near 0 opacities would darken the corresponding colors severely. See figure 15b).

Comparing both methods with the results from the resampling ⇒ classification approach (see figure 15c)) shows that the Goss interpolation yields a consistent image while the trilinear interpolation either introduces darkening or color shifts.


Figure 15: a) Classifying before resampling together with standard trilinear interpolation introduces self occlusion or color shifts. b) Classifying before resampling together with Goss style interpolation is consistent with the reference image c) below. c) Reference image: resampling before classifying together with trilinear interpolation.

# 5 Conclusions

I have shown that with sharp transitions in data, classification function and color assigning functions one has to expect rendering artifacts. As the resampling before classification approach allows to minimize these artifacts by taking more samples along each ray (and thus also taking longer rendering time) I prefer this order even though it produces view dependent images.

The second result is that pre-classified or pre-colored datasets which are intended to show fog or gel like materials have to be represented with more than 8 bits precision. 12 bits seems to be the lower boundary for acceptable visual results. In oder to composite the samples along a ray through those datasets one has to have even more precision in the compositing buffer to overcome the rounding to zero problem for the individual low opacity samples. My preliminary experiments show 14 bits as a lower boundary for the accumulation buffer size.

Finally, my studies made it very obvious that in the case of classification before resampling one should do trilinear interpolation for the opacities, but alpha weighted interpolation for R, G and B.

*Ingmar Bitter*
415-857-7462 (o)
415-324-4703 (h)
415-852-3791 fax
bitter@hpl.hp.com
ingmar@cs.sunysb.edu
www.cs.sunysb.edu/~ingmar

# References

[ACM96] ACM. *Computer Graphics, SIGGRAPH'96*, Anual Conference Series, New Orleans, LA, August 1996.

[Ben95] Mark Bentum. *Interactive Visualization of Volume Data.* PhD thesis, Twente Univertity, Enschede, December 1995.

[GGSC96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Computer Graphics, SIGGRAPH'96* [ACM96], pages 43–54.

[HL79] G. T. Herman and H. K. Liu. 3D display of human organs from CT. *Computer Graphics and Image Processing*, 9(1):1–21, January 1979.

[Kau90] Arie Kaufman, editor. *Volume Visualization.* IEEE Computer Society Press, Los Alamitos, CA, 1990.

[LC87] William E. Lorensen and Harvey E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. In *Computer Graphics, SIGGRAPH'87*, volume 21(4), pages 163–169, Anaheim, CA, July 1987.

[Lev88] Marc Levoy. Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications*, 8(5):29–37, May 1988.

[LH96] Marc Levoy and Pat Hanrahan. Light field rendering. In *Computer Graphics, SIGGRAPH'96* [ACM96], pages 31–42.

[PWK95] H. Pfister, F. Wessels, and A. Kaufman. Cube-4: A scalable architecture for real-time volume rendering. Technical Report TR.95.01.15, State University of New York at Stony Brook, Computer Science Department, Stony Brook, NY 11794-4400, January 1995.

[SK94] L. Sobierajski and A. Kaufman. Volumetric ray tracing. *Volume Visualization Symposium Proceedings*, pages 11–19, October 1994.

[Sob94] L. Sobierajski. *Global Illumination Models for Volume Rendering.* PhD thesis, State University of New York at Stony Brook, Computer Science Department, Stony Brook, NY 11794-4400, August 1994.

[Wes91] L. A. Westover. *Splatting: A Parallel, Feed-Forward Volume Rendering Algorithm.* PhD thesis, The University of North Carolina at Chapel Hill, Department of Computer Science, July 1991. Technical Report, TR91-029.