Foundations for Measuring Volume Rendering Quality

Peter L. Williams[†] and Samuel P. Uselton[‡]

[†]Vassar College
Poughkeepsie, N.Y. 12601

[‡]MRJ Inc.
NASA Ames Research Center
Moffett Field, CA 94035-1000

Abstract

The goal of this paper is to provide a foundation for objectively comparing volume rendered images. The key elements of the foundation are: (1) a rigorous specification of all the parameters that need to be specified to define the conditions under which a volume rendered image is generated; (2) a methodology for difference classification, including a suite of functions or metrics to quantify and classify the difference between two volume rendered images that will support an analysis of the relative importance of particular differences. The results of this method can be used to study the changes caused by modifying particular parameter values, to compare and quantify changes between images of similar data sets rendered in the same way, and even to detect errors in the design, implementation or modification of a volume rendering system. If one has a benchmark image, for example one created by a high accuracy volume rendering system, the method can be used to evaluate the accuracy of a given image.

1 INTRODUCTION

As the availability and use of scientific visualization software has increased, the need to improve methods of evaluating this software has become apparent. In the beginning, having a visualization of data was so much better than not having one that much could be forgiven.

The visualization programmers were the domain scientists or worked directly with them. If there were errors in the visualization, they would be obvious to the scientist and so could easily be fixed. But now as the developers of visualization software become more removed from the users, the importance of verifiable accuracy grows. Visualizations can be very misleading and the errors are easy to miss. It is essential to be able to distinguish between the results of the visualization that reflect the data and the results due to the visualization process itself [19]. This concern is evident in the visualization community from the recent proliferation of panels and papers on the subject [9, 19, 20, 23, 38, 51, 63, 76].

For scientific visualization, the best evaluation of an image's accuracy and quality is its ability to support a viewer's accomplishment of particular tasks — so the ideal method for evaluating visualization software is by user testing. But user testing is difficult, time consuming and expensive; and the results don't generalize easily. Therefore, we would like to have computable functions for evaluating visualizations which at least roughly predict human performance of tasks dependent on the visualization.

In this paper, we focus on the evaluation of volume rendered images because this area is sufficiently narrow to be tractable and sufficiently broad to be useful. The methods used for direct volume rendering and the results generated are sufficiently similar across applications to raise hopes that the same design process, if not exactly the same functions, will be broadly applicable.

Methods for volume rendering rectilinear or voxel data sets, such as scanned medical data, have been intensively investigated [8, 10, 25, 27, 29, 30, 33, 34, 40, 52, 55, 59, 62, 67, 77]. As a result of this research, and by exploiting the geometric regularity of the data samples, volume rendering systems now exist that can interactively generate useful images of such data, even when the data sets are very large.

On the other hand, volume rendering systems for nonrectilinear (curvilinear, unstructured, or scattered) data sets are relatively slow due to the complex geometric configuration of the underlying mesh [5, 11, 12, 13, 15, 28, 35, 37, 40, 55, 57, 64, 65, 68, 69, 74, 78]. For large unstructured data sets, it is currently necessary either to wait a substantial time for an image to be generated or to introduce various approximations into the volume rendering process to reduce the time. Some approximation methods are justifiable and can result in useful images for visualization.[1] However, it is important to know exactly how much these approximations are distorting the volume rendered image. Therefore, this class of applications is in especial need of methods for evaluating image quality and accuracy.

The goal of this paper is to provide a foundation for objectively comparing volume rendered images. The two key elements of the foundation are: (1) a exhaustive list of all the parameters that need to be specified to define the conditions under which a volume rendered image is

[1]The question "Why not resample an unstructured data set to a rectilinear mesh?" is frequently encountered. Unfortunately, there are theoretical and practical difficulties associated with this. These issues are addressed in Appendix A.

generated; (2) a methodology for difference classification, including a suite of functions or metrics to quantify and classify the difference between two volume rendered images that will support an analysis of the relative importance of particular differences.

A complete specification of the parameters is necessary to properly compare the results from different software rendering "the same" image, and to be able to isolate the effects on an image of particular parameter choices. In addition, such a list of all the possible parameters available for selection may bring to mind new applications and possibilities that had not occurred to the visualization scientist. Some of these possibilities are discussed in Section 3.

In designing metrics, we are specifically interested in measuring the differences between two images that affect the interpretation of the data. Some differences, such as a small amount of random noise or the overall brightness of the image, may have no significant effect on what the user notices. Other differences, such as an edge or a bright spot, could seriously mislead a scientist who is looking for patterns or structure in a data set. Our approach is to develop: (a) a methodology for difference classification, (b) metrics which can be objectively calculated and that yield a numerical measure of the various classes of difference, and (c) techniques for analyzing the content of each of these categories of difference and, if possible, finding the source of the differences. The differences are separated into three classes, each of which has a distinct perceptual impact. The importance of a particular class of difference will depend on the visualization task being performed.

The difference metrics have several potential applications. The results of this method can be used to study the changes caused by modifying particular parameter values, or to compare and quantify changes between images of similar data sets rendered in the same way, for example air flow past the same aircraft but at different velocities, or changes in images of different time slices from the same time-varying data set. They can even be used to to detect errors in the design, implementation or modification of a volume rendering system. If one has a benchmark image, for example one created by a trusted system of high precision, the metrics can be used to evaluate the accuracy of a given image and to measure the effect on images when approximations are introduced into the volume rendering process.

This paper provides a complete specification of the input parameters for a volume rendering system. In addition, we present a methodology for analyzing the difference between two volume rendered images. A companion paper will report on the application of the methodology developed herein to evaluate images produced by several existing volume rendering systems for a suite of nonrectilinear data sets. Several benchmark scenes will be included. The design and implementation of a high accuracy volume rendering system for generating benchmark quality images of unstructured data sets is described by Williams, Max and Stein in [73].

The following section refines the problem statement and places the problem in context. Section 3 proposes a standard generation specification for volume rendered images. Section 4 describes a set of metrics to classify and quantify the differences between two volume rendered images.

2 PROBLEM STATEMENT

The best test of the quality of a visualization is whether it enables the user to accomplish the intended task more accurately, more rapidly and more easily than alternatives. Such an evaluation requires careful specification of the task and precise measures of accuracy in accomplishing it. It also requires extensive testing with an appropriate user population and a carefully defined testing procedure. The results of such testing are dependable but expensive and narrowly applicable. If the task to be accomplished or the population using the tool are varied or if some characteristics of the visualizations or input data differ, then the previous results do not necessarily apply. Volume rendering has an extremely wide range of applications, the tasks in each application vary dramatically, and many techniques are available for generating the images. Once many narrow studies have been done, it may be possible to formulate and test more general models, but these models will not be available for use soon.

We would like a set of analytical tests which provide an objective basis for comparing the quality of visualizations. Ideally, the results of these tests will be good predictors of human performance on a range of tasks when using these visualizations. These tests must measure differences in images in a way that corresponds to differences in users' perceptions and interpretations of the images.

Three examples illustrate the diversity of applications and tasks that can be addressed with volume rendering. In medicine, volume rendering is used to display CT and MR data. The tasks to be performed range from material classification of regions of the volume to estimating size and relative position of particular features. The users are highly trained and there is a clear standard of what to expect, for example a kidney has a typical size, shape and location. Variations from the norm are of primary interest.

In seismic exploration, the data collected resembles ultrasound medical data, and some of the tasks, particularly classification, are also similar. However, the context is quite different. The training of a seismic interpreter is very different from that of a physician, and there is a much wider range in the possible structures to discover. The kind of structures of interest also vary with the intent of the scientist, for example earthquake prediction as opposed to seeking oil.

In computational aerodynamics, a fluid is being modeled. Therefore material classification is not a significant part of the visualization as it is in the case of seismic or medical applications. Vector rather than scalar quantities are the primary focus. Scalar values are derived which highlight information about the vector field. A vortex in a flow field is a very different kind of feature from a kidney stone inside a human patient.

To develop tools that are useful across such a broad range of applications, the tasks to be accomplished must be considered very abstractly, and human perceptual abilities must be a major guide. Several sources [4, 14, 16, 49, 60, 61] have developed guidelines for producing good visualizations. These rules are based on human perceptual abilities and propensities but are generally focused on presentation of information to an audience, rather than on constructing visualizations for a scientist to explore new data.

A number of image quality metrics have been proposed and discussed in the literature. Barrett [2, 3] discusses the derivation of a "mathematical observer" in the context of radiological images, where the task is restricted to assigning the image to one of a fixed number of classes (eg "lesion present" or "lesion absent"). Image quality is then determined by the performance of the mathematical observer. We seek a method for comparing images that is less task specific. Neumann [43] describes an aggregate comparison metric in which an image is represented as a single vector of pixel attributes and the metric is calculated as the inner product of two normalized image-vectors. Neumann also discusses a non image-based feature scale method for measuring image quality. Wilhelms and Van Gelder [70] use the combination of RMS and maximum absolute difference as a metric. A good deal of attention has been given to investigating and developing metrics which include a consideration of human perceptual performance/experience. Several overall image quality measures based on the human visual system are discussed in [18, 22, 24, 36, 45, 47, 53, 56, 58]. However, these measures are complex and not entirely complete, and no universal agreement has been achieved despite years of research. We prefer to objectively classify the difference between two images into categories, each of which has a unique perceptual impact on the user, and then to apply a metric to each category individually.

3 STANDARD GENERATION SPECIFICATIONS

We want to measure the differences in volume rendered images in a way that will allow us to determine whether two systems given the same input really make the same picture. We also want to be able to measure the impact of changing rendering parameters or the effect of changes in the data (the scene content).

To make a useful comparison of images generated by different volume rendering systems requires great care in assuring that all the parameters (the scene, the viewing specifications, the data set, etc) are appropriately matched. Similarly, a study of the effects of changing a single parameter requires the careful control of all the other parameters. So a complete listing of all the parameters that require specification is an important prerequisite for using the metrics. The set of specific instances of all these parameters that is used to generate an actual image is called the *generation specification* for that image.

Certain image generation information may be specified in different but equivalent ways, e.g. a viewing position may be stated as a series of rotations and translations, or by an eye position, a look-at position and an up-vector. We will choose one and assume that conversions can be computed.

Most volume rendering systems will not have explicit inputs for controlling all the possible parameters. Many parameter choices are made as design decisions during system development; other choices may not have even occurred to the developers. These hardwired parameters must also be determined in order to do a thorough job of understanding a comparison.

Sub-Section A gives a complete and detailed list of all the parameters that must be specified to precisely define the conditions under which a volume rendered image is generated. In Sub-Section B, the parameters are described in further detail.

A. Parameter Listing

The following summarizes the general categories of information needed to specify the conditions under which a volume rendered image is generated.

<u>General Specification Categories</u>

- Scene Description
  - Data Set
  - Transfer Functions
  - Background
  - Lighting
  - Derived and Embedded Objects
- Viewing Parameters
- Optical Model
- Image Specification

The information within each of the above categories is enumerated below.

<u>Scene Description:</u>

<u>Data Set:</u>

- geometry
- data field name(s), data values, and precision
- time step
- data location (at vertices, or face- or cell-centered)
- mathematical operations on the field values resulting in derived values
- intra-cell interpolation information
- modeling transformations for multiple grids
- symmetry transformations

<u>Transfer Functions:</u>

- mapping from scalar data value to color emission
- mapping from scalar data value to extinction coefficient

Background:

- background color and/or texture map

Lighting:

- intensity, location and direction of lights
- intensity of ambient light
- illumination model

Derived and Embedded Objects:

- surface or procedural description
- material properties and shading model for surfaces

Viewing Parameters:

- projection type (orthographic/perspective)
- eye position, look-at position, up vector
- viewplane window (*right, left, top, bottom*)
- distance from eye to viewplane
- z-near, z-far clipping planes
- viewing coordinate system handedness

Optical Model:

- underlying theoretical optical model or light intensity differential equation
- precise definition of optical properties of volume density

Image Specification:

- image size in pixels
- image format

B. Explanations, Preferences and Assumptions

In this section, the descriptions of certain parameters listed in the previous section are amplified and/or qualified. Any related assumptions as well as our preferences are stated.

The Scene Description:

The Data Set:

The data set geometry must be specified — this includes the following information: the coordinate system (Cartesian, cylindrical, spherical, etc.) in which the data is defined, in the case of a Cartesian system, whether it is right or left handed, the number of vertices in the data set, and the coordinates of each vertex. If the data set is defined on an underlying grid or mesh, the following information should also be given: the number of cells of each type (tetrahedron, pyramid, prism, brick, etc.), the vertices which define each cell and each face of each cell. Currently, there is no standard format for these specifications. For unstructured data sets, we use the format given by Williams, Max and Stein in [73]. For structured CFD data sets, such as curvilinear meshes with hexahedral cells, the PLOT3D file formats, as updated in [66], seem useful.

Since a data set usually consists of several fields, each specified for one or more time steps, the name(s) of the specific data field(s) being visualized and their associated time step must be specified. (As described in the following section on transfer functions, more than one field may be used directly in creating the visualization.) If the field being visualized is itself derived from one or more of the raw data fields, the function to calculate this derived field must be specified. In addition, the precision of the data fields in the data set should be stated.

To allow the field to be correctly evaluated within a cell, interpolation functions need to be specified for each cell type. If applicable, the order of the cells should be stated; for example, finite element data sets may have linear, quadratic or cubic cells and the cell faces may be planar or curvilinear. To ensure maximum accuracy in the visualization, the scientist generating the data set should provide as much information as possible about the interpolation methods used in the simulation, including if applicable, the actual shape or parameter functions used to generate the data. For scattered data, Nielson [44] and Ruprecht and Muller [50] discuss appropriate interpolation methods. Either the use of locally bounded multiquadric radial basis functions, or the triangulation of the points with a Delaunay or a regular triangulation [32] and then use of linear interpolation seems preferable.

If a data set has multiple grids, each with its own coordinate system, the modeling transformation must be provided for each grid so as to specify the relative positions of the grids. Any symmetry transformations used to visualize symmetric areas of the field that are not actually part of the data set should be specified.

Transfer functions:

Transfer functions specify the mappings from scalar field values to color and extinction coefficient values. These mappings need to be clearly specified. One method for doing this is described below. In addition, for non-gray-scale maps, the color model (RGB, HSV, etc.) should be specified. Sometimes it is useful to map one data field to color and a different field to opacity (and even to show iso-surfaces of a third field value); these mappings need to be stated.

Transfer functions are commonly specified in a piecewise linear fashion; there are a number of benefits from doing this. When the scalar field varies linearly within a cell, the differential equation for the light intensity along a ray through the volume can be integrated exactly if the transfer functions are linear or piecewise linear [75] (assuming the use of the RGB color model). In addition, the use of linear or piecewise linear transfer functions allows certain efficiencies in the volume rendering process.

In selecting the transfer functions, it should be recognized that a C^1 discontinuity occurs at the junction of two adjacent linear portions of a piecewise linear transfer function. This discontinuity can introduce artifacts into the volume rendered image. For example, inside a linear tetrahedra, the field varies linearly therefore the intensity in the image should too; but if within the cell the field values straddle a discontinuity in the transfer function, a nonlinear variation in the intensity will be introduced. To eliminate this effect would require the transfer functions to be linear over their entire range (constant or a ramp). This is a rather impractical limitation and usually it suffices just to minimize the number of breakpoints. For maximum accuracy, the transfer functions should be applied after the data is interpolated within a cell, rather than interpolating the color and extinction coefficient.

An unambiguous specification of the transfer functions is a key element in the generation of volume rendered images. One useful method for specifying the functions is a table with entries for the data field value, the color and the extinction coefficient, s_i, c_i, e_i, $1 \leq i \leq n$, where $s_i \leq s_{i+1}$. For $s < s_1$ the color and extinction coefficient are c_1, e_1; similarly for $s > s_n$ they are c_n, e_n. For $s_i < s < s_{i+1}$, color and extinction coefficient are linearly interpolated between c_i and c_{i+1}, and e_i and e_{i+1}, respectively. A discontinuity in the transfer function can be specified by giving two different sets of color and extinction coefficient for the same scalar value, so for successive indices i and j, $s_i = s_j$, s_i, c_i, e_i and s_j, c_j, e_j would be specified. The idea being that one piece of the transfer function runs from s_{i-1} to s_i and another piece from s_j to s_{j+1}, thus allowing a discontinuity at $s_i = s_j$.

The Background:

The background may be a solid color specified as a pixel value, or a pattern specified as an image or as a procedural mapping from either the data set coordinate system or the viewing coordinate system to a pixel value. For the purpose of image comparison, a solid background, preferably black, eliminates the problem of a slight misalignment in background patterns. However, for general use, background patterns, such as a black and white checkerboard, are an excellent aid in understanding semitransparent images.

Lighting:

If the optical model requires lighting, for example to produce shaded contour surfaces without actually constructing surface polygons as in Max's single scattering model [41], or if lighting is required to shade the surfaces of embedded or derived objects, it is necessary to specify the direction, position and intensity of the external light(s), the intensity of the ambient light, and the illumination model, for example Lambert, Phong, Blinn-Phong, etc.,

Derived and Embedded Objects:

It can be very useful to embed surfaces or solid objects in a visualization to serve as a physical reference and give visual cues, for example an airfoil in a flow field or the biopsy needle in an image from ultrasound data. Often this information is given as a set of polygons in the data set. Derived objects, such as isosurfaces, cutting planes, stream surfaces, etc. should be fully specified, including the name of the scalar field from which they are derived and the criteria used to calculate them, such as the contour value, rake position, etc.

The color of these objects should also be specified. If the surfaces of these objects are shaded, the shading method, for example Gouraud or Phong, and the relevant material properties (ambient, diffuse and specular reflectivity, shininess, transparency, etc.) appropriate for the illumination model need to be specified. In addition, the outward direction of the surfaces should be specified, e.g. for a contour surface, whether the normal points in the direction of the gradient or in the opposite direction.

Viewing Parameters:

The viewing parameters described in Section 3 are complete. However any alternative specification may be used if it can be unambiguously converted to that shown here. We assume the view plane is perpendicular to the line between the eye position and the look-at position as is usual in volume rendering. If this is not the case, this should be stated. We also assume that all parameters are given in world coordinates.

Optical Model:

A number of different optical models suitable for volume rendering are described by Max in [41]. The optical model specification must include both (a) a precise definition of the optical properties: color emission (or glow energy if monochromatic) and/or absorption (optical density or extinction coefficient)[2], and (b) the differential equation for the light intensity along a ray through the volume in terms of these optical properties. Examples of such differential equations are given by equations (1), (4), and (6) in [41]. The differential equation for the model should be specified in its basic form before any simplifications or assumptions (such as assuming the color or scalar field is constant within a cell, or that the interpolation function is linear). This should be done even if simplifications or assumptions are used to actually generate the image. This allows a benchmark system to create an accurate image for comparison based on the unsimplified differential equation. Failure to specify any aspect of the optical model can result in ambiguity. An actual example of ambiguity in a visualization caused by an incomplete specification of the underlying optical model is given in Appendix B.

Image Specification:

Formally, an *image* is a two dimensional function $f(x, y)$, where x and y denote spatial screen coordinates and the value of f at any point (x, y) is the pixel value of the image at that point. For our purposes, a *pixel value* p is an n-dimensional vector $p \in \Re^n$ specifying light intensity. For a gray-scale image, n is 1. For a tristimulus system such as RGB or HSV, n is 3. Unless otherwise specified, an image refers to a volume rendered image.

For benchmark comparisons, pixel values should have the maximum available precision. This

[2] Optical density ρ is the term used in [75], it is equivalent to the extinction coefficient τ used in [41].

of course will depend not only on the compiler, but on the rendering method (hardware or software) and the algorithm used. If the pixel values are generated as floating point numbers and then converted to limited precision (e.g. 8-bit per channel) for display, it is preferable to use the floating point values for comparison purposes. The pixel values should be the raw values, before any adjustments such as for ratio-based intensity or gamma correction. Compression should not be used unless it is lossless. For the work herein, the pixel values are single precision floating point numbers in the range $[0.0, 1.0]$ (except for difference images which are introduced and described in the next section).

The image size is specified in pixels, as *xres* pixels horizontally by *yres* pixels vertically. The pixel array storage convention should be clearly specified. The location of the point at which the pixel value is calculated within the overall area of the pixel should be specified, for example at the center of the pixel area or at the lower left hand corner, etc.

4 DIFFERENCE METRICS

A *metric* is a measure of the distance between two objects. In mathematical analysis, metrics are required to be nonnegative, independent of the order in which the objects are specified, and abide by the triangle inequality. In addition, if the distance between objects is zero, they must be the same object. The metrics we define are intended to be a measure of the difference between two images. A difference of zero should surely mean the images are the same. Making the measurements nonnegative and symmetric is easily done. Only the verification of the triangle inequality may be difficult. Because we classify the difference and compute measures of each class separately, calling these measurements metrics may be abusing the term slightly, but it reminds us of the ideal we would like to have.

We may use the metric to compare an image generated by a newly constructed volume renderer with a benchmark image, or to measure the effect on an image when a rendering parameter is varied. We first focus on metrics for comparing a single color component of two images. These may be two gray-scale images or a single channel of two multichannel images. Other methods for multichannel image comparison are discussed below. For the remainder of this paper, we assume all images have a solid black background.

Let the *difference image* of two images A and B be the image formed by subtracting image B from image A on a pixel by pixel basis; this is written as $\mathit{diff}(A, B)$. For the purposes of this paper, pixel values of a difference image are in the range $[-1.0, 1.0]$. A difference image can be displayed in a number of ways, for example: (a) display the absolute value of each pixel value, (b) clamp the pixel values at ± 0.5 and then add 0.5 to each pixel value, (c) divide the pixel values by 0.5 and then add 0.5 to each pixel value, or (d) convert the difference image to a color image where the positive pixel values and the negative pixel values are each mapped to different colors.

Any of the following operations applied to the absolute value of all pixels in the difference image is a valid metric on A and B: the sum, the maximum value, the arithmetic mean, the mid-mean (the arithmetic mean of the half-list obtained by dropping out the highest quarter and lowest quarter), the geometric mean, the median, or the root-mean-square (RMS).

In calculating the above metrics, only those pixels in the difference image which correspond to a nonblack (nonbackground) pixel in either image A or image B are counted. Pixels which are black in both images A and B do not count towards the total number of pixels when calculating these metrics. We call the pixels that do count, the *nonbackground* pixels in the difference image. Therefore, each pixel in the difference image needs to be tagged as nonbackground or not nonbackground. A separate shadow array can be used for the tags.

The standard deviation and the variance can be used to measure the dispersion of the distribution of differences around the above central measures. The dispersion can be visualized by displaying a histogram of the pixel values in the difference image, or by displaying the difference image itself.

For multichannel images, a separate analysis of each channel seems preferable since this permits the detection of a difference or an error occurring in only one channel. It is possible, however, to use a *vector norm* on $\Re^n$, a measure of distance in a vector space, to compare a pixel $p_1 \in \Re^n$ of image A with a pixel $p_2 \in \Re^n$ of image B. Usually the Euclidean norm $l_2(p_1, p_2)$ or the ∞-norm $l_\infty(p_1, p_2)$ are used for this purpose [17]. Certain color spaces may support the comparison of multichannel images better than others. Glassner [17] argues that taking a difference in a perceptually uniform color space such as L-a-b space or $L^*u^*v^*$ space may be most appropriate. More research in this area is needed.

Of the metrics described above, the RMS measure is most commonly used for image comparison purposes. However, there is rather widespread agreement that RMS error is not a good measure of overall image differences or of perceptual distortion. For example, Teo [58] in an image comparison study, shows that it is possible for images with less visible distortion to have a greater RMS error. Indeed none of the above metrics are completely satisfactory for overall image comparison.

Figure 1[3] shows a volume rendered image of coolant velocity magnitude from a finite element simulation of coolant flow inside a component of the French Super Phoenix nuclear reactor. This image will serve as a benchmark image; it was created by the high accuracy volume rendering system described by Williams, Max and Stein [73]. Several different types of errors were added to the benchmark image; these erroneous images are shown in Figures 2, 3, and 4. The RMS difference between each of these erroneous images and the benchmark image is the same (0.044); yet the visual effect of the error is significantly different in each of these images. For most visualization purposes, the image shown in Figure 3 is just as useful as the benchmark image in Figure 1; whereas the image in Figure 4 is clearly misleading. Figure 5

[3]The figures can be found at http://www.nas.nasa.gov/NAS/TechReports/NASreports/NAS-96-021/NAS-96-021.html

shows the cumulative effect of adding all three types of error at once to the benchmark image. This image will be used as an example in explaining our metrics and the process for computing them.

Existing approaches to image quality metrics are discussed at the end of Section 2. Our approach is to decompose the overall or *aggregate* difference between two images into categories, each of which has a distinct perceptual impact, and then measure and analyze each of these categories. The categories are carefully chosen so that each category of difference is responded to quite differently by the human visual system. When applied to these individual categories of difference, the metrics described above give a more useful representation of how the images differ.

We partition the difference between two images into three categories: bias, noise and structured difference. These categories are defined in the following sections. We refer to the overall difference image of two images A and B, $\mathit{diff}(A, B)$, as the *aggregate difference image*. The aggregate difference image, *diff*(Figure 1, Figure 5), is shown in Figure 6. (Method (c), described above, is used to display all difference images in this paper.) Our goal then is to use numerical statistical methods as well as image processing techniques: (a) to separate the aggregate difference into noise, bias and structured difference, (b) to quantify with mathematically computable metrics the amount of difference in each of these categories, and (c) to analyze each category of difference to determine its composition, and if possible, to find its sources.

The importance of a particular component or subcomponent of difference will depend on the intended use of the image. Some differences, such as a small component of low level random noise or a small difference in the overall brightness of an image, may not have a significant effect on the user. Whereas a difference in the form of a small bright blob or a halo could be very misleading depending on the specific visualization task being performed. Separating the difference into different general categories allows us to discriminate between differences that are perceptually less significant and those that can seriously affect the interpretation of the data. Given a specific task, the metrics can then be used to predict the relative value of two images for accomplishing that task.

The following sections define and discuss each of these classes of difference, investigate the implications of each on visualization, and describe methods to isolate, measure and analyze each category of difference.

A. Noise

Noise in a signal is an uncorrelated modification to the original signal. For the purpose of analyzing the differences between two images, we assume that random or spurious differences in pixel values that are not part of any larger pattern are noise. Sources of noise in a volume rendered image can include numerical or roundoff error, stochastic sampling, and numerical overflow during the computation of the image. Numerical overflow can occur if a limited number of bits, e.g. 8 bits, is used to accumulate pixel intensities; a slight increase or decrease in intensity can cause an extreme change in the pixel value, e.g. from 255 to 0. Generally, numerical error will cause low level random noise with a fairly high spatial frequency and a normal distribution of differences whose mean will be close to zero. Stochastic sampling and overflow can give rise to spike noise or pops; this form of noise will generally have a low amplitude uniform distribution of differences.

Because of its lack of spatial correlation, noise in an image generally has a higher spatial frequency spectrum than the more structured components of the aggregate difference image. Therefore low-pass spatial filtering can be effective for removing noise. There are several techniques for image smoothing including neighborhood averaging and low-pass filtering [21, 48]. Typical smoothing filters perform some form of moving window operation that may be a convolution or other local computation in the window. Linear smoothing can be performed either by convolution or by Fourier filtering. There is no difference in the results; however, Fourier domain processing often gives more insight into the nature of the noise and so is useful in designing a noise filter. Fourier filtering methods are described in [42, 48].

It is easy to smooth an image and thereby filter out noise; the problem is how to do so without blurring interesting features. Typically edges have a high spatial frequency so smoothing filters tends to attenuate or blur the sharpness of edges in an image. This blurring of edges will also result in incorrectly classifying a portion of the aggregate difference image as noise. For this reason, we emphasize edge-preserving filters.

Typical windowing filters set the value of the center pixel in a neighborhood or window to a function of the other pixels in the window. This function may be the mean, the weighted mean, the median, the mode, or the mean of the k pixels in the window whose values are closest to the center pixel. Lee and Redner [31] describe a class of nonlinear filters called *alpha* filters. In these filters, the samples in the neighborhood are sorted in increasing order, a percentage α of the samples are removed from both ends of the sorted list, and then the remaining samples are averaged. To improve the filtering, the alpha filter can be repeatedly applied with different values of alpha.

Windowing filters can have different shapes. Some of the more useful window shapes are square, plus-sign and octagonal. Typically the sizes of the windows range from 4 to 24 of the nearer neighbors. Small sized windows are preferable since features or artifacts smaller than about 3×3 to 5×5 pixels can often be safely ignored.

Other window filtering methods are based on gradients, homogeneity, iteration, cascaded filtering or conditional filtering [48]. A quantitative comparison of some of these methods is given by Chin and Yeh in [6]. In addition, optimal filtering can be used for noise suppression [26]. Shapiro and Silverman [54] present a method to characterize the amount of noise in a signal prior to its removal. A noise filter may work reasonably well on one type of image but not on another so care must be taken in selecting a filter. The nature of the noise and the attributes of features of interest, such as horizontal or vertical alignment, should be incorporated into the design of the filter — one way to approach to this is to use Fourier analysis as referred to above.

The median, the k nearest neighbors and the alpha filter with a plus-sign shaped window seem to be the best edge-preserving filters. The blurring effect of neighborhood averaging can also be reduced by using a threshold procedure [21]. The idea here is to leave unchanged those values whose difference from their neighborhood values do not exceed the specified value of the threshold. This technique can produce an equivalent reduction in noise to non-threshold smoothing, but can leave the edges much sharper.

We calculate a *noise-filtered difference image* by applying one of the above edge-preserving filters to the aggregate difference image. Let the aggregate difference image be D and the noise-filtered difference image be NF, then we define the *noise image* X to be $\mathit{diff}(D, NF)$. A metric for the noise in the aggregate difference image D is given by $\mathcal{M}_{noise} = \sum_{all\_pixel\_values} |X|$. This metric is an approximation since the noise-filtered difference image may contain a component of structured difference: (a) a fine grained texture or pattern whose detail is smaller than the pixel smoothing window or (b) the effects of edge blurs resulting from the filtering process. Isolating these components from the noise-filtered difference image before calculating the noise metric will result in a more accurate measure of noise. Methods for isolating and removing fine grained texture in an image are given in [1, 48]. One method to identify edge blurs is described next.

The effects of edge blurs in the noise image can be detected by an analysis of the distribution of the nonbackground pixel values in the noise image. A histogram is very useful for doing this. Pure noise will usually have either a uniform or a normal distribution of differences. The degree of deviation from either of these distributions is a measure of edge blur in the noise image. For a more accurate analysis, the noise image can be successively subdivided into smaller tiles and the distribution of difference analyzed for each tile. This procedure is also valuable for determining if the noise is uniformly distributed spatially over the image or is constrained to certain regions.

The human visual system is remarkably tolerant of noise [7, 72]. Therefore unless the signal to noise ratio is low, noise has little effect on the use of the image. When an image A is being compared with a benchmark image B, a signal to noise ratio can be calculated for image A as $(\sum_{all\_pixel\_values} B)/\mathcal{M}_{noise}$, where $\mathcal{M}_{noise}$ is the noise metric for the aggregate difference image $\mathit{diff}(A, B)$.

Figure 7 shows the noise-filtered difference image after filtering the aggregate difference image shown in Figure 6. To create the noise filtered difference image, the aggregate difference image was filtered with one pass of a median filter, followed by a single pass of an alpha filter with $\alpha = 0.5$. A 3x3 plus-sign shaped window, which included five pixels, was used for both filters. The noise metric for the image shown in Figure 6 is $\mathcal{M}_{noise} = 3023.42$, and the signal to noise ratio is $\frac{27707.10}{3023.42} = 9.16$.

The noise should be removed from the aggregate difference image before proceeding to isolate the bias and structured difference as described in the following two sections.

B. Bias

If one of two images being compared is uniformly brighter than the other, but has no other difference, the aggregate difference metrics will show a significant difference even though the information content of both images could be perceptually equivalent. In statistics, bias is defined as the expected value of the difference between a value and an estimate of that value. We use the term *bias* to refer to a (relatively) uniform difference in intensity between images across the pixels of the image. The effect of bias may not be serious if the pixel intensities lie in the linear portion of the response curve of the monitor and the human visual system.

The difference of the medians of the nonbackground pixel values of each of the images being compared is an excellent metric for bias, $\mathcal{M}_{bias} = Median(A) - Median(B)$, where A and B are the two images being compared, B is the benchmark image if applicable, and background pixels, as determined from the difference image $diff(A, B)$, are ignored. In order to have a measure of the bias that can be compared with the noise and structured difference metrics, which are cumulative measures rather than per pixel measures, we define a cumulative bias metric $\mathcal{M}_{bias\_overall} = n * \mathcal{M}_{bias}$ where n is the number of nonbackground pixels. The bias metric computed by subtracting the median of the nonbackground pixels in the benchmark image from the median of Figure 5 after the removal of noise is $\mathcal{M}_{bias} = +0.063$. This indicates that each pixel in the error image is 6.3% brighter than in the benchmark image. The cumulative bias metric is $\mathcal{M}_{bias\_overall} = 8446.03$.

The *mode* of a distribution $f(x)$ is the value of x where $f(x)$ is maximum. The mode of a distribution of pixel values can be determined by an analysis of a histogram of the pixel values using a limited number of buckets. Provided the mode of the distribution of pixel values in a noise-filtered difference image is sufficiently distinct, it can also be a metric for bias, $\mathcal{M}_{bias2} = Mode(noise\_filtered\_difference\_image)$.

The bias should be subtracted from the noise-filtered difference image before analyzing the structured difference as described in the next section.

C. Structured Difference

Structured difference is that part of the aggregate difference image which consists of objects which have a coherent structural form, such as edges, halos, lines, bright or dark spots, blobs, textures or patterns. We refer to these objects collectively as *structured objects*. The presence of structured difference means the images being compared have a meaningful difference in content in the context of scientific visualization. These differences may be features that are introduced or enhanced or are lost or attenuated; they may or may not represent features inherent in the data set. In any case, the class of difference which is most important to focus on when comparing and/or evaluating volume rendered images is structured difference. In visualization, the scientist is looking for patterns or structure in the data, therefore structured difference is highly significant as it could mislead the scientist.

After noise and bias have been removed from the aggregate difference image, the resulting image, the *structured difference image*, can be measured. Figure 8 shows the structured difference image after removing the bias from the noise-filtered difference image, Figure 7. One metric for structured difference is $\mathcal{M}_{struc\_diff} = \sum_{all\_pixel\_values}(|structured\_difference\_image|)$. The structured difference metric for Figure 8 is $\mathcal{M}_{struc\_diff} = 693.63$. Alternatively, any of the metrics described at the beginning of Section 4 such as RMS or mean could be applied to the structured difference image. The key factors we want to measure are the number of pixels involved, the magnitude of the difference and the spatial context. Which is the best metric will depend on the visualization application.

The structured difference image should then be analyzed for content using the feature extraction techniques outlined below. Statistics can be gathered on the number and size of the various coherent objects: e.g. lines, blobs, edges, halos, in this image. The features can optionally be classified as to alignment: as grid or cell aligned, view aligned, background aligned, or unaligned.

An analysis of the structured difference image may yield important clues as to the source of the differences. This may lead to the identification of errors in the design or implementation of the volume rendering system — for example, grid-aligned differences may indicate improper treatment of degeneracy or the differences may be as innocent as a slight change in viewing parameters or that pixel values were computed at pixel corners rather than at pixel centers. Once a structured object has been isolated in the structured difference image, the images being compared should be analyzed to find the source of this object.

Feature extraction techniques include thresholding, Fourier transform analysis, image enhancement techniques, edge detection, gradient finding, shape analysis reconstruction and region growing algorithms. For example, blobs, spots, edges or halos in an image can be isolated by the following method. A threshold filter [1, 26] is performed on the absolute value of a difference image for several different threshold values. In each resulting image all pixels whose difference is greater than or equal to the threshold are set to 1, the remainder are set to 0. Next, a region growing algorithm, such as blob coloring or segmentation [21] is used to find contiguous blobs or shapes whose difference is binary 1. The patterns so found can be displayed visually and/or the number of pixels comprising each blob can be calculated for each threshold. Further details on feature extraction techniques can be found in [1, 17, 21, 26, 48].

It can be very helpful to graphically display the structured difference image alongside the images being compared and visually explore them using simple graphics and image processing tools, such as a magnifier, filter or histogram. Often, prominent structured objects can be located in the difference image and then traced to their source in one of the original images. However to locate more subtle features, it is usually necessary to apply the feature extraction techniques described above.

5 SUMMARY

The goal of this paper is to lay a foundation for objectively comparing volume rendered images, leading to objective evaluation of their accuracy and quality. In order for an image comparison to be useful the images should be created with equivalent parameters or at least with any differences known. In order to generate a benchmark image for comparison purposes, the parameter specifications must be provided. So an essential prerequisite for image comparison is a complete list of all possible parameters that must be specified. We provide such a listing and also state preferred formats and/or values for many of the parameters which will result in maximizing the effectiveness of the comparison process, especially for benchmark comparisons.

When comparing two images it is not enough just to look at a measure of the overall or aggregate difference between the images. Rather it is necessary to measure specific differences that affect the interpretation of the images. To dramatize this, we showed a benchmark image and a set of three erroneous images each of which had a different category of error added to it. The RMS difference between the benchmark image and each error image is identical. As can be seen in these figures, the error has a very different perceptual impact in each of the three cases. Therefore, in this paper we focus on analyzing and decomposing the difference before measuring it. To do this, we develop: (a) a methodology for classifying the difference between two images as noise, bias or structured difference, (b) a suite of metrics to objectively quantify each class of difference, and (c) techniques to analyze the nature of each class of difference and relate it to its source.

The difference metrics can be used to measure the effect on images when approximations are introduced into the volume rendering process, and they can lead to the detection of errors in the design or implementation of a volume rendering system. The metrics can also be used to determine the effect on an image when a rendering parameter is changed, to compare images generated by two different volume rendering systems, and to detect and quantify changes between images of similar data sets or images of different time steps in time-varying data sets.

Although the methodology described herein was developed primarily for evaluating images of nonrectilinear data sets, most of this work is also relevant to uniform grids. This research was not designed to exclude voxel data, but rather to be as generally applicable as possible. This work may also be applicable to benchmarking studies of images created by general purpose radiosity or ray tracing systems.

The difference classification methods and the metrics described herein are currently being used to evaluate images produced by several existing volume rendering systems for a suite of nonrectilinear data sets. A companion paper will report on this work and also will include several volume rendering benchmarks. Each of these benchmark scenes will consist of an image generated by a high accuracy volume rendering system and its complete generation specification, as defined in Section 3B.

Work in this area is just beginning, so additional work is needed on a wide variety of topics. The development of additional categories of differences and better metrics will improve the usefulness of this work. Methods for analysis of structured difference are likely to be application dependent, but also quite useful. Noise filters for Gaussian noise (and other distributions) can be developed. And finally, testing the accuracy of our hypothesis that structured differences are what really matters (by measuring the performance of people performing specific tasks) will be useful and enlightening.

ACKNOWLEDGMENTS

We are grateful to David Kenwright, Dan Asimov and Nelson Max for reading an early draft of this paper and offering many valuable comments. Nelson Max and Allen Van Gelder provided helpful discussions relating to Appendix B. Charlie Smith suggested the use of the median as a measure of bias. Keki Burjorjee, with support from the Undergraduate Research Summer Institute at Vassar College, implemented some of the image processing tools described herein. The data set used to generate the images in this paper was provided by Bruno Nitrosso and Electricité de France. The first author gratefully acknowledges summer support and equipment loans from NASA Ames Research Center

APPENDIX A

The question is often asked, "If volume rendering curvilinear and unstructured grids is so slow, why not resample the data to a regular grid?" There are both theoretical and pragmatic difficulties with this suggestion.

Assume the original data is completely specified by the locations of the original points, the values at these original data points and some interpolation function between these points. The degree of the interpolation function and the geometry of the original mesh determine a lower limit on the size of the smallest feature that can be present. This limit and the Nyquist sampling theorem determine the required sampling density to capture all features; at least two samples per interval of this feature size are needed in each dimension. For simplicity, assume linear interpolation across tetrahedral grids, and trilinear interpolation across hexahedral cells of curvilinear grids. These assumptions force all extrema of the field to be at sampled locations, so no feature is smaller than the smallest distance between original grid points. Sampling at twice the density in each dimension of the smallest cell in a curvilinear grid of hexahedral cells results in an eightfold increase in the number of samples required, assuming all the cells are roughly the same size. If a volume renderer for regular grids runs between ten and one hundred times faster than one for unstructured or curvilinear grids, this trade-off may seem advantageous. However there are problems with this approach.

A strictly pragmatic problem is that fast volume renderers usually require all the data to fit in memory. But nonrectilinear data sets often are very large. Resampling such a data set makes it at least eight times larger and probably far too large to fit in memory. Then the fast regular grid renderer slows to a rate limited by disk paging.

A much more severe, but still pragmatic, difficulty arises when confronted by real data sets. The reason fluid dynamics computations are performed on curvilinear grids is, in part, because the size of the features of interest varies drastically across the volume of the computation. The grid cell sizes vary correspondingly to capture these features. The aeroscientists spend a great deal of time and effort designing these grids to capture features of interest as precisely as possible with a fixed budget of nodes. This limit is determined by the memory available on the machine performing the computational fluid dynamics (CFD) calculation. Typical grid sizes for problems run recently at NASA's Numerical Aerodynamics Simulation Center are between several hundred thousand and a few tens of millions of nodes.

The ratio of the longest grid cell edge to the shortest edge in typical aeronautical fluid dynamics calculations ranges between one thousand and one million. If the entire volume is sampled at a uniform density, determined by the sampling requirements of the smallest grid cell, the data set size is multiplied by several thousand or million, not just eight. This regular grid may not fit even on a workstation disk, let alone in main memory. The appropriate response to this difficulty would seem to be a multiresolution representation. A straightforward multiresolution representation using axis aligned samples at integral multiple step sizes might get back to the eight-fold increase. But rendering this multiresolution structure will not be as fast as rendering a regular grid.

The theoretical difficulties arise in dealing with the external boundaries of the volume to be resampled, or with cell boundaries where there may be discontinuities in the data values. Computational scientists choose curvilinear and unstructured grids not only so they can capture features of varying size at different locations in the field, but also because these grids can capture the unique shape of the volumes over which the computations are performed. Aircraft fuselages and engine combustion chambers, to give two examples, are extremely complex shapes. The boundaries of the computational grids are designed to provide a good polygonal approximation to these shapes, often using a great many very small cells on these boundaries. Resampling using only axis aligned cells, even with multiple resolutions, inevitably either misses some of the volume of interest or places regular grid nodes outside the original input volume. Missing some of the volume is unacceptable, particularly since the regions near these boundaries are usually the most interesting part of the flow.

Cells with some points inside the aircraft body suffer another, even more unpleasant difficulty; the field being approximated is discontinuous at the aircraft boundary, that is, there is no flow inside the wing, fuselage or turbine blade. This discontinuity translates into a

Nyquist requirement for an infinitely dense sample in the neighborhood of this boundary. In structural modeling, similar discontinuities may exist between interior cells, for example at an interface between two different materials.

There is one more argument used to support the possibility of regular resampling of this data. This argument is that a digital image is a regularly sampled grid of the influence of the data set, so it must be possible to limit sampling by what is visible. This statement is true, as far as it goes. However to reap the benefits of a regular grid, the conversion via resampling must be done only once, or at least very infrequently; it is the resampling involved in creating the image that takes so long and drives us to consider converting to a regular grid. Changes in the viewing direction or distance force resampling, invalidating the benefits one might have expected.

APPENDIX B

Volume rendering systems are based on a theoretical optical model for light interacting with a volume density or cloud. This appendix gives an example of how an incomplete description of the underlying optical model can result in an ambiguity in the generation specification for a volume rendered image. The model used for illustration is the *absorption plus emission* model [41, 75]; every point in the volume density both absorbs light and also emits light (glows). We will assume monochromatic light; however, the example generalizes to colored light. The differential equation for the light intensity along a ray towards the eye through the volume density is:

$$dI(s) = g(s)ds - I(s)\tau(s)ds \tag{1}$$

where s is a length parameter along the ray and $I(s)$ is the light intensity at s. The extinction coefficient $\tau(s)$ is a function of position in the volume density and is usually assigned by a transfer function based on the scalar field $f(s)$ being visualized. In the limit as ds tends to zero, $\tau(s)ds$ is the fraction of light coming from behind s that is occluded or absorbed at s. When a point s in the cloud is totally transparent, $\tau(s) = 0$; when it is totally opaque, $\tau(s) = \infty$. The term $g(s)$ is the source or glow term; it has the meaning that in the limit as ds tends to zero, $g(s)ds$ is the intensity of light emitted at s.

At this point it might seem that the model is completely specified. However, there are at least two ways to treat the glow term $g(s)$. Wilhelms and Van Gelder [71] treat $g(s)$ as a physical property of each point in the cloud. Whereas, Williams and Max [75] consider $g(s) = \kappa(s)\tau(s)$, where the chromaticity $\kappa(s)$ is a physical property of each point in the cloud. They rationalize that if a point in the cloud has zero extinction coefficient then it should also have zero glow energy. In the former case, the color transfer function specifies $g(s)$, whereas in the latter case it specifies $\kappa(s)$. (Porter and Duff [46] allude to this difference in their discussion of premultiplying color by opacity in the context of subpixel accumulation and in their description of an *opaque* operator for dealing with luminescent objects which adds color information without obscuring the background.)

In the context of this appendix, the important point is that, with the same input data, a volume renderer based on the first assumption about $g(s)$ can generate quite different images than one based on the second assumption. Van Gelder calls the model that treats $g(s)$ as a physical property of the cloud the Los Angeles neon and smog model. In this model, when the extinction coefficient is zero, the cloud is completely transparent yet glows with an energy $g(s)ds$ — like a neon[4] filled bulb, or to further the analogy, like looking at something through colored glasses. When the extinction coefficient at a point tends toward infinity, the light intensity goes to zero and the image becomes black at that point — hence the smog. On the other hand, in the Williams and Max model where $g(s) = \kappa(s)\tau(s)$, when the extinction coefficient goes to zero, the glow energy also goes to zero; when the extinction coefficient at a point s tends to infinity, the light intensity at s is $\kappa(s)$.

The advantages of the first model are: (a) It allows the modeling of a transparent glowing cloud, (to do this with the second model would require an infinitely bright $\kappa(s)$). (b) The glow energy can be specified in terms of a different scalar field than the one used for the extinction coefficient.

The advantages of the second model are: (a) It can be more intuitive for some volume rendering applications. For example, (i) increasing the extinction coefficient transfer function makes the surface color more dominant, rather than making the image darker and ultimately black, and (ii) if the extinction coefficient at a point in the volume is very small, the glow energy will also be very small. (b) The extinction coefficient does not appear in the denominator of the cumulative light intensity equation as it does in the first model, see Equation (8) in [71], so the cumulative light intensity is well behaved for very small or zero extinction coefficient.

The top image in Figure 9 is a volume rendering of the Super Phoenix data set using the Williams and Max treatment of the glow energy. The bottom image is a volume rendering of the same data set with identical viewing parameters and transfer functions except that the Wilhelms and Van Gelder treatment of glow energy is utilized. Both images were created using the *absorption plus emission* model. The bottom image is brighter since the color is not premultiplied by the extinction coefficient which in this case is always less than one. The white in the bottom image is caused by the saturation of all three channels due to the brighter colors. Both images were generated by the volume rendering system described in [73].

One other difference is worth mentioning — the interpolation properties of the two models. If the scalar field and the transfer functions are linear on a ray segment, then the first term on the right hand side of Equation 1 will be linear for the neon and smog model, whereas in the case of the Williams and Max model that term will be quadratic.

[4] a transparent light-emitting material

When images from volume rendering systems based on each of these versions of the same theoretical optical model were first compared side-by-side, there was some initial confusion. The images were quite different, yet it was thought that the underlying models were equivalent. The lesson was clear. It is important to clearly understand all aspects and implications of the optical model on which a volume renderer is based.

References

[1] D. H. Ballard and C. M. Brown, "Computer Vision," Prentice-Hall, 1982.

[2] H. H. Barrett, "Objective Assessment of Image Quality: Effects of Quantum Noise and Object Variability," *J. Optical Soc of Amer*, vol. 7, no. 7, pp. 1266–1278, July 1990.

[3] H. H. Barrett, J. Yao, J. P. Rolland and K. J. Myers, "Model Observers for Assessment of Image Quality," *Proc. Natl. Acad. Sci.*, vol. 90, pp. 9758–9765, Nov. 1993.

[4] J. Bertin, "Semiology of Graphics," 1967, reprinted by University of Wisconsin Press, Madison, WI, 1983.

[5] J. Challinger, "Parallel Volume Rendering for Curvilinear Volumes," *Proc. Scalable High Performance Computing Conference*, April 1992.

[6] R. T. Chin and C.-L. Yeh, "Quantitative Evaluation of Some Edge Preserving Noise Smoothing Filters," *Computer Vision, Graphics, and Image Processing*, no 23, 1983.

[7] R. L. Cook, "Stochastic Sampling in Computer Graphics," *ACM Trans. Computer Graphics*, vol. 5, no. 1, pp. 51–72, Jan. 1986.

[8] R. A. Drebin, L. Carpenter and P. Hanrahan, "Volume Rendering," *Computer Graphics*, vol. 22, no. 4, pp. 65–74, Aug. 1988.

[9] F. D. Fracchia, R. D. Kriz, B. Mones-Hattal, L. Jorgenson, B. Rogowitz, "Panel: Is Visualization Struggling under the Myth of Objectivity?" *Proc. Visualization '95*, pp. 412–4415, Oct. 1995.

[10] H. Fuchs, M.S. Levoy, and S. Pizer, "Interactive Visualization of 3D Medical Data," *Computer* pp. 46–50, Aug. 1989.

[11] R. Gallagher and J. Nagtegaal, "An Efficient 3D Visualization Technique for Finite Element Models and Other Coarse Volumes," *Computer Graphics*, vol. 23, no. 3, pp. 185–192, July 1989.

[12] M. P. Garrity, "Raytracing Irregular Volume Data," *Computer Graphics*, vol. 24, no. 5, pp. 35–40, Dec. 1990.

[13] C. Giertsen, "Volume Visualization of Sparse Irregular Meshes," *Computer Graphics*, vol. 12, no. 2, pp. 40–48, Mar. 1992.

[14] C. Giertsen, "Creative Parameter Selection for Volume Visualization," *Jrn. Visualization and Comput. Anim.*, vol. 3, pp. 1–11, 1992.

[15] C. Giertsen and A. Tuchman, "Fast Volume Rendering with Embedded Geometric Primitives," *Visual Computing - Integrating Computer Graphics with Computer Vision*, T.L.Kunii (ed), Springer Verlag, pp. 253–271, 1992.

[16] C. Giertsen, "Direct Volume Rendering of Multiple Scalar Fields," *Jrn. Visualization and Comput. Anim.*, vol. 5, pp. 69–84, 1994.

[17] A. S. Glassner, "Principles of Digital Image Synthesis," Volume 1, Morgan Kaufmann Publishers, 1995.

[18] W. Glenn, "Digital Image Compression Based on Visual Perception and Scene Properties," *SMPTE Journal*, pp. 392–397, May 1993.

[19] A. Globus and S. Uselton, "Evaluation of Visualization Software," *Computer Graphics*, vol. 29, no. 2, pp 41–44, May 1995.

[20] A. Globus and E. Raible, "Fourteen Ways to Say Nothing with Scientific Visualization," *IEEE Computer*, vol 27, no. 7, pp. 86–88, July 1994.

[21] R. C. Gonzalez and R. E. Woods, "Digital Image Processing," Addison-Wesley, 1992.

[22] D. J. Granrath, "The Role of Human Visual Models in Image Processing," *Proc. of IEEE*, vol. 69, no. 5, pp 552-561, May 1981.

[23] N. Greene and M. Kass "Error-Bounded Antialiased Rendering of Complex Environments," *Proc. Siggraph 94*, Orlando, pp. 59–66, July 1994.

[24] C. Hall and E. Hall, "A Nonlinear Model for the Spatial Characteristics of the Human Visual System," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-7, no. 3, pp. 161–170, Mar. 1977.

[25] P. Hanrahan, "Three-Pass Affine Transform for Volume Rendering," *Computer Graphics*, vol.24, no. 5, pp.71–78, Nov. 1990.

[26] B. K. P. Horn, "Robot Vision," MIT Press, 1986.

[27] A. Kaufman, D. Cohen, and R. Yagel, "Volume Graphics," *Computer*, vol.26, no.7, pp. 51–64, July 1993.

[28] K. Koyamada, "Volume Visualization for the Unstructured Data," *SPIE Vol. 1259 Extracting Meaning from Complex Data: Processing, Display, Interaction*, 1990.

[29] P. Lacroute and M. Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," *Computer Graphics*, vol. 28, no. 4, pp. 451–458, Aug. 1994.

[30] D. Laur and P. Hanrahan, "Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering," *ACM SIGGRAPH Comput. Gr.*, vol. 25, no. 4, pp. 285–288, July 1991.

[31] M. E. Lee and R. A. Redner, "A Note on the Use of Nonlinear Filtering in Computer Graphics," *IEEE Computer Graphics & Applications*, vol 10, no. 5, pp. 23–29, May 1990.

[32] C. Lee, Regular Triangulations of Convex Polytopes. *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift* P. Gritzmann and B. Sturmfels, eds. Amer Math. Soc., Providence, RI, 1991.

[33] M. Levoy, "A Hybrid Ray Tracer for Rendering Polygon and Volume Data," *IEEE Computer Graphics and Applications*, ppp. 33–40, Mar. 1990.

[34] M. Levoy, "Efficient Ray Tracing of Volume Data," *ACM Trans on Graphics*, vol. 9, no. 3, pp. 245–261, July 1990.

[35] B. A. Lucas, "A Scientific Visualization Renderer," *Proc. Visualization '92*, Boston, pp. 227–234, Oct. 1992.

[36] Mannos and Sakrison, "The Effects of a Vision Fidelity Criterion on the Encoding of Images," *IEEE Trans. Information Theory*, vol. 20, pp. 525–536, July 1974.

[37] X. Mao, L. Hong and A. Kaufman, "Splatting of Curvilinear Volumes," *Proc. Visualization '95*, pp. 61–68, Oct. 1995.

[38] S. R. Marschner and R. J. Lobb, "An Evaluation of Reconstruction Filters for Volume Rendering," *Proc. 1994 Symposium on Volume Visualization*, Washington, DC, pp. 100–107, Oct. 1994.

[39] G. A. Mastin, "Adaptive Filters for Digital Image Noise Smoothing: An Evaluation," *Computer Vision, Graphics, and Image Processing*, vol. 31, no. 1, pp 103–121, July, 1985.

[40] N. Max, P. Hanrahan and R. Crawfis, "Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions," *Computer Graphics*, vol. 24, no. 5, pp. 27–33, Dec. 1990.

[41] N. Max, "Optical Models for Direct Volume Rendering," *IEEE Trans. on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 99–108, June 1995.

[42] R. Nathan, "Spatial Frequency Filtering," in *Picture Processing and Psychopictorics*, B. S. Lipkin and A. Rosenfeld, eds., Academic Press, New York, pp. 151–164, 1970.

[43] U. Neumann, "Volume Reconstruction and Parallel Rendering Algorithms: A Comparative Analysis," PhD thesis, University of North Carolina at Chapel Hill, 1993.

[44] G. Nielson, T. Foley, B. Hamann and D. Lane, "Visualizing and Modeling Scattered Multivariate Data," *Computer Graphics*, vol. 11, no. 3, pp. 47–55, May, 1991.

[45] N. B. Nill, "A Visual Model Weighted Cosine Transform for Image Compression and Quality Assessment"' *IEEE Trans. Communications*, vol. com-33, no. 6, pp. 551–557, June 1985.

[46] T. Porter and T. Duff, "Compositing Digital Images," *Computer Graphics*, vol. 18, no. 4, pp 253–260, July 1984.

[47] C. Poynton. "Gamma and It's Disguises: The Nonlinear Mappings of Intensity in Perception, CRTs, Film and Video," *SMPTE Journal*, pp. 1099–1108, Dec. 1993.

[48] W. K. Pratt, "Digital Image Processing," John Wiley & Sons, 2nd ed., 1991.

[49] B. E. Rogowitz and L. A. Treinish, "An Architecture for Perceptual Rule-based Visualization," *Proc. Visualization '93*, pp. 236–243, Oct. 1993.

[50] D. Ruprecht and H. Muller, "Image Warping with Scattered Data Interpolation," *Computer Graphics*, vol. 15, no. 2, pp. 37–43, Mar. 1995.

[51] H. Rushmeier, M. Botts, S. Uselton, J. Walton, H. Watkins and D. Watson, "Panel: Metrics and Benchmarks for Visualization," *Proc. Visualization '95*, pp. 422–426, Oct. 1995.

[52] P. Sabella, "A Rendering Algorithm for Visualizing 3D Scalar Fields," *Computer Graphics*, vol. 22, no. 4, pp. 51-58, Aug. 1988.

[53] Saghri, Cheatham, and Habbi, "Image quality measure based on a human visual system model," *Optical Engineering*, vol. 28, no. 7, pp. 813–818, July 1989.

[54] H. R. Shapiro and R. A. Silverman, "Alias-free Sampling of Random Noise," *Journal Soc. Ind. and Applied Math.*, vol. 8, no. 2, pp. 225–248, 1960.

[55] P. Shirley and A. Tuchman, "A Polygonal Approximation to Direct Scalar Volume Rendering," *Computer Graphics*, vol. 24, no. 5, pp. 63–70, Dec. 1990.

[56] H. Snyder. Image Quality: Measures and Visual Performance. in *Flat-Panel Displays and CRT's*, L. E. Tannas, ed., Van Nos Reinhold, 1984.

[57] C. Stein, B. Becker and N. Max, "Sorting and Hardware Assisted Rendering for Volume Visualization," *Proc. 1994 Symposium on Volume Visualization*, Washington, DC, pp. 83–90, Oct. 1994.

[58] P. C. Teo and D. J. Heeger, "Perceptual Image Distortion," *Proc. First International Conf. on Image Processing*, Austin, pp. 982–986, Nov. 1994.

[59] T. Totsuka and M. Levoy, "Frequency Domain Volume Rendering," *Computer Graphics*, pp. 271–278, 1993.

[60] E. R. Tufte, "The Visual Display of Quantitative Information," Graphics Press, 1983.

[61] E. R. Tufte, "Envisioning Information," Graphics Press, 1990.

[62] C. Upson and M. Keeler, "V-BUFFER: Visible Volume Rendering," *Computer Graphics*, vol. 22, no. 4, pp. 59-64, July 1988.

[63] S. P. Uselton, G. Dorn, C. Farhat, M. Vannier, K. Esbenson and A. Globus, "Validation, Verification and Evaluation of Visualization Software," *Proc. Visualization '94*, Washington, DC, pp. 414-418, Oct. 1994.

[64] S. P. Uselton, "Volume Rendering on Curvilinear Grids for CFD," *AIAA 94-0322, AIAA 32nd Aerospace Sciences Meeting* Reno, Nev., Jan. 1994.

[65] A. Van Gelder and J. Wilhelms, "Rapid Exploration of Curvilinear Grids Using Direct Volume Rendering," *Proc. Visualization '93*, pp. 70–77, Oct. 1993.

[66] P. P. Walatka, J. Clucas, R. K. McCabe, T. Plessel and R. Potter, "FAST User Guide," *Tech. Report RND-93-010*, NASA Ames Research Center, July 1993.

[67] L. Westover, "Interactive Volume Rendering," *Proc. 1989 Workshop Volume Visualization*, Chapel Hill, pp. 9-16, May 1989.

[68] J. Wilhelms, J. Challinger, N. Alper, S. Ramamoorthy and A. Vaziri, "Direct Volume Rendering of Curvilinear Volumes," *Computer Graphics*, vol. 24, no.5, pp. 41–47, Dec. 1990.

[69] J. Wilhelms, "Pursuing Interactive Visualization of Irregular Grids,", *The Visual Computer*, vol. 9, pp. 450–458, 1993.

[70] J. Wilhelms and A. van Gelder, "Multi-Dimensional Trees for Controlled Volume Rendering and Compression," *Proc. 1994 Symposium on Volume Visualization*, Washington, DC, pp. 27–34, Oct 1994.

[71] J. Wilhelms and A. van Gelder, "A Coherent Projection Approach for Direct Volume Rendering," *Computer Graphics*, vol. 25, no. 4, pp. 275-284, July 1991.

[72] D. R. Williams and R. Collier, "Consequences of Spatial Sampling by a Human Photoreceptor Mosaic," *Science*, vol. 221, pp. 385–387, July 22, 1983.

[73] P. L. Williams, N. Max and C. Stein, "A High Accuracy Volume Rendering System for Unstructured Data," manuscript, Jan. 1996.

[74] P. L. Williams, "Interactive Splatting of Nonrectilinear Volumes," *Proc. Visualization '92*, Boston, pp. 37–44, Oct. 1992.

[75] P. L. Williams and N. A. Max, "A Volume Density Optical Model," *Proc. 1992 Workshop Volume Visualization*, Boston, pp. 61–68, Oct. 1992.

[76] C. M. Wittenbrink, A. T. Pang and S. Lodha, "Verity Visualization: Visual Mappings," *Tech. Report UCSC-CRL-95-48*, University of Calif., Santa Cruz, 1995.

[77] R. Yagel, "Realistic Display of Volumes," *SPIE Proc. Medical Imaging VI: Image Capture, Formatting, and Display*, vol. 1653, pp. 470–476, Feb. 1991.

[78] R. Yagel, D. S. Ebert, J. N. Scott and Y. Kurzion, "Grouping Volume Renderers for Enhanced Visualization in Computational Fluid Dynamics," *IEEE Trans. on Visualization and Computer Graphics*, vol. 1, no. 2, pp. 117–132, June 1995.

The figures for this paper can be found at http://www.nas.nasa.gov/NAS/TechReports/NASreports/NAS-96-021/NAS-96-021.html