Volume Rendering Methods for Computational Fluid Dynamics Visualization

David S. Ebert† Roni Yagel‡ Jim Scott§ Yair Kurzion‡

†Computer Science Department, University of Maryland Baltimore County, Baltimore, MD 21228*

‡Department of Computer and Information Science, The Ohio State University, Columbus, Ohio 43210

§Department of Aeronautical and Astronautical Engineering, The Ohio State University, Columbus, Ohio 43210

Abstract

This paper describes three alternative volume rendering approaches to visualizing computational fluid dynamics (CFD) data. One new approach uses realistic volumetric gas rendering techniques to produce photo-realistic images and animations from scalar CFD data. The second uses ray casting that is based on a simpler illumination model and is mainly centered around a versatile new tool for the design of transfer functions. The third method employs a simple illumination model and rapid rendering mechanisms to provide efficient preview capabilities. These tools provide a large range of volume rendering capabilities to be used by the CFD explorer to render rapidly for navigation through the data, to emphasize data features (e.g., shock waves) with a specific transfer function, or to present a realistic rendition of the model.

1 Introduction

This paper describes three alternative approaches for volumetric visualization of CFD data. One new approach uses a realistic surface and volumetric rendering and animation system which will allow scientists to obtain accurate photo-realistic images and animations of their simulations, containing more information in a more comprehensible format than current tools available to the researchers. The second approach is based on ray casting without the modeling of self shadows. It is mainly intended to support the interactive design of transfer function. Currently this tool supports transfer function mapping voxel value (density) and gradient to opacity and mapping of voxel values to colors. The third approach is mainly intended for interactive preview of the data. It is based on an extended implementation of the *template based ray casting* [1] that supports screen and volume supersampling. To support interactivity, this renderer is based on a very simple illumination model. Combined, these tools are shown to provide effective visualization of scalar CFD data.

1.1 Background and the state of CFD visualization

Computational fluid dynamics is an active research area. CFD research involves large three-dimensional volumes of data. The data from CFD simulations often contains many data values per three-dimensional location (e.g., velocity, pressure, density, energy, temperature). Recently, scientific visualization systems have been developed to aid CFD researchers in the interpretation of these computational data sets, including commercial systems such as Data Visualizer, AVS, and Iris Explorer^{TM}, and non-commercial systems such as FAST by NASA Ames Research Center. Techniques used in these systems include isosurface rendering, stream-lines, contour plots, and three-dimensional volume visualization.

Current visualization systems for CFD simulations are lacking in the following way:

> Many current systems fail to accurately display the entirety of the three-dimensional data from the CFD simulations. CFD simulations model three-dimensional flow phenomena. Surface based rendering techniques, stream-lines, and contour plots can only capture a small fraction of the data in a three-dimensional simulation, resulting in important information being hidden or not easily discernible in the resulting images. Volume visualization is an important tool for discerning three-dimensional data, especially three-dimensional scalar data. A comparison of Figures 1, 3 and Figure 2 reveals the quantity of information that can be lost in traditional visualization systems. Figure 2 uses a standard ray-marching volume rendering algorithm [2] to produce results similar to a isosurface renderer; whereas, Figure 1 uses our gaseous CFD visualization system and Figure 3 is rendered by our ray caster which employs a specially designed opacity transfer function.

*e-mail: ebert@cs.umbc.edu

There has been some recent work on addressing some of these shortcomings. However, these systems only overcome some of the shortcomings of current CFD visualization systems. Surface particles [3], and a combination of volume rendering, vector fields, and texturing [4] have been used to capture the dynamics and massive information content in a CFD simulation.

While visualization systems for CFD have concentrated on simple illumination and shading models, recent advances in computer graphics for realistic rendering of gases and fluids have produced dramatic and near photo-realistic images of water, steam, fog, and smoke [5, 6, 7, 8, 9]. These techniques use physically-based rendering techniques to display these natural phenomena realistically. These photo-realistic rendering techniques can be incorporated into volume visualization to provide important perceptual cues for comprehending complex three-dimensional data.

Our systems for CFD visualization combine several volume rendering techniques which provide the scientist with the option to trade time for image quality. One technique employs a sophisticated illumination model to render our test data in approximately 83 seconds. The second technique simplifies illumination and emphasizes material modeling (via transfer functions) to render the data in approximately 64 seconds. The third technique is mainly intended for speedy rendering based on a basic illumination model and material modeling which produce an image in approximately 27 seconds. All reported times in this paper where taken on a HP 9000 Series Workstation model 735.

1.2 Turbo-jet compressor visualization

The development of advanced aerospace propulsion systems relies heavily upon the management of critical flow features which influence performance and operating characteristics. In order to understand the critical flow behavior, numerical simulation techniques must be used to complement experimental studies. Among the propulsion system components in which some of the most complex flow behavior is encountered are the turbomachinery components. In the compressor, the flow may be entering a stage at a supersonic velocity in which case a complex system of shocks and expansion waves form through the blade passage. This system of shocks and expansions not only produce losses themselves, but also contribute to other flow features which degrade performance. In order to control such flow features in a manner to optimize performance and minimize losses, numerical solution of the time-dependent Navier-Stokes equations is necessary. Such solutions provide the velocity field, along with pressure, density and temperature variation in time and space through the compressor rotor blade passages. To understand the nature of these phenomena and their influence on performance, it is desirable to be able to have a visual representation of the computed flow data which can be compared directly with experimental flow visualization. This type of data is particularly valuable in the analysis of shock waves for which the temporal and spatial variation can have a significant impact on the mass flow rate of air passing through the compressor. The use of interactive visualization will provide the engineer a means of analyzing the behavior of such critical flow features as he controls the flow entering the compressor stage under investigation. Specifically, this represents control of the operating condition through interactive variation of the flow pressure, velocity, or temperature, all of which influence performance characteristics. Other flow features which are of specific interest and play crucial roles in the overall performance and operation of turbomachinery devices include: vortices which form around the blade leading edges and along the corners at the blade - hub junction, and boundary layer growth and separation. The analysis of these features and their influence is also significantly enhanced through the use of interactive flow visualization of the type demonstrated here.

We have applied the three visualization techniques of our system to this problem of visualizing the flow between two blades of the compressor of a turbo-jet engine [10]. The computational mesh for the simulation is a three-dimensional curvilinear grid with dimensions of $64 \times 46 \times 18$. For visualization purposes, the computational grid was resampled to a rectilinear grid with dimensions of $64 \times 64 \times 32$.

2 The gas visualization system

Our gaseous visualization system demonstrates the benefits of realistic volumetric rendering techniques for flow visualization.

A more complete description of the system can be found in [5, 11]. The system has the following features:

- Fast combination of surface-based objects and volume data.
- Fast, accurate, physically-based atmospheric attenuation and illumination models for low-albedo gases.
- Fast volumetric shadowing algorithm.

The system is a hybrid surface and volume rendering system, which uses a fast scanline a-buffer rendering algorithm for the surface-defined objects in the scene, while volume modeled objects are volume rendered. The algorithm first creates the a-buffer for a scanline containing a list for each pixel of all the fragments that partially or fully cover the pixel. Then, if a volume is active for a pixel, the extent of volume rendering needed is determined. The volume rendering is performed next, creating a-buffer fragments for the separate sections of the volumes. Volume rendering ceases once full coverage of the pixel by volume or surfaced-defined elements is achieved. Finally, these volume a-buffer fragments are sorted into the a-buffer fragment list based on their average Z-depth values and the a-buffer fragment list is rendered to produce the final color of the pixel.

The system supports volume rendering of both procedural and scientific data-defined volumes. The rendering techniques currently available in this system for volumes include gas-based rendering methods and traditional density-based volumetric rendering [2].

2.1 Gaseous volume rendering algorithm

The volume rendering technique used for gases in this system is similar to the one discussed in [2]. The ray from the eye through the pixel is traced through the defining geometry of the volume. For each increment through the volume sections, the density is determined from the CFD data. The color, density, opacity, shadowing, and illumination of each sample is then calculated. The illumination and densities are accumulated based on a low-albedo illumination model for gases and atmospheric attenuation. The following algorithm is the basic gas rendering algorithm:

```
for each section of gas
  for each increment along the ray
    get color, density, & opacity of
        this element
    if self_shadowing
       retrieve shadowing of this element
         from the solid shadow table
    color = calculate gas illumination
       using opacity, density and the
       appropriate model
    final_color = final_color + color;
    sum_density =sum_density +density;
    if( transparency < 0.01)
       stop tracing
    increment sample_point
  create the a_buffer fragment
```

In sampling along the ray, a Monte Carlo method is used to choose the sample point to reduce aliasing artifacts. In the gaseous model, we are approximating an integral to calculate the opacity along the ray [12]. Therefore, the opacity is the density obtained from evaluating a volume density function multiplied by the step-size. The approximation used is

$$opacity = 1 - e^{-\tau \sum_{t_{near}}^{t_{far}} \rho(x(t),y(t),z(t))\Delta t}$$

where τ is the optical depth of the material, $\rho()$ is the density of the material, t_{near} is the starting point for the volume tracing, and t_{far} is the ending point. The final increment along the ray may be smaller, so its opacity is scaled proportionally [13].

The volume density functions interpolate the values stored in the CFD grid and allow density scalars and power functions to be applied to enhance the visualization results. By applying a density scalar, more volumetric information can be seen. Applying a power function to the density allows the contrast in change of the densities to be increased. The following function is used to achieve these results:

$$density = (density * density\_scalar)^{power\_exponent}$$

2.2 Illumination algorithm

For the gaseous rendering, the following low-albedo illumination model is used, where the phase-function is based on the summation of Henyey-Greenstein functions as described in [14]:

$$\mathrm{B} = \sum_{\mathrm{t_{near}}}^{\mathrm{t_{far}}} \mathrm{e}^{-\tau * \sum_{\mathrm{t_{near}}}^{\mathrm{t}} \rho(\mathrm{x(u),y(u),z(u)})\Delta \mathrm{u}} \times \mathrm{I} \times \rho(x(t), y(t), z(t)) \times \Delta t,$$

where I is

$$\sum_i I_i(x(t), y(t), z(t)) phase(\theta).$$

$Phase(\theta)$ is the phase function, the function characterizing the total brightness of a particle as a function of the angle between the light and the eye [14]. $I_i(x(t), y(t), z(t))$ is the amount of light from light source i reflected from this element.

The system also features a fast, accurate table-based volume shadowing technique [5] to increase the visual realism of the resulting images. The system can, therefore, accurately display three-dimensional gases and the shadows they cast.

2.3 Results for gaseous rendering

A comparison of the results achievable through the three-dimensional gas rendering of our system and more traditional volumetric rendering techniques can be seen in Figures 1 and 2. Both figures show the visualization of the static pressure for the computation, illuminated with two light sources. Figure 1 shows the data visualized using the gaseous volumetric rendering techniques. Figure 2 shows the results achievable using more traditional volumetric rendering techniques, where the gradient of the change in the data is used to imply surfaces and a traditional Phong surface illumination model is used. Figure 2 is lacking in several respects. First, the lack of shadowing makes the understanding of the image more difficult. Second, the implied surfaces obscure the details of the flow data within. Third, additional flow detail is lost because of the small degree of change in the values in the data. In contrast, notice the details in the changes in the pressure that can be seen in Figure 1. Also notice the depth of information that can be discerned in this image. The image uses self-shadowing of the volume data, as well as shadowing of the volume onto the walls to increase the understandability of the data. As you can see, much more of the three-dimensional flow can be seen using our gas visualization technique.

The advantages of the gas visualization prototype system can be clearly seen from comparing these two images. These images were computed at a resolution of 512×341 and took 83 seconds each to compute. Lower resolution images can be combined with rendering optimizations and more powerful graphics workstations to achieve interactivity of the visualizations.

Figure 5 shows the flexibility of the system for visualization. This figure contains 9 images with varying density scaling and power exponent values used in the visualization. The following table has the parameter values for each image in the figure (image 1 is the top-left image, image 9 is the bottom-right image).

Image	*Density Scalar*	*Exponent*
1	1.0	3.0
2	1.0	1.84
3	1.0	0.98
4	0.91	3.0
5	0.91	1.84
6	0.91	0.98
7	0.73	3.0
8	0.73	1.84
9	0.73	0.98

From left to right in the image, the power exponent is decreased, increasing the density of the data. From top to bottom, the density scaling factor is decreased, increasing the transparency of the flow.

This sequence of images demonstrates how the variation of transparency/opacity can be used to investigate the spatial gradients of a particular flow quantity. In this case, the static pressure varies significantly throughout the flowfield and among the critical flow features. The most critical flow features, which arise as gradients in the static pressure, are shock and expansion waves. These waves appear as abrupt changes or nearly discontinuous changes in the static pressure. In the sequence of renderings shown here, the shock can be seen as the abrupt change from blue to white along the vertical part of what appears as a cutout region on the left hand side of the image. Once again, the shock is caused by the turning of the flow as it enters the blade passage. It is readily apparent that the strength of the shock varies significantly across the flowfield or along the blade leading edge (the blade's geometry-angle thickness and twist vary in the radial direction from hub to tip). This sequence allows the viewer to look into the flowfield and see where the most intense or sharpest gradients exist and what their influence is on the subsequent (downstream) flow. In this figure, the greatest pressure gradient exists at the region which appears as the cutout in the upper left hand region of the flowfield just after the flow enters the blade passage. This is indicative of the greatest shock strength. Examination of the successive images across and down the figure reveals that the shock strength decreases as it approaches the adjacent blade surface. This sequence also gives an indication of the three-dimensionality of the shock geometry and shock strength which can have a significant influence on the operating characteristics of the compressor. Such analysis is especially valuable when it is incorporated into animations which will provide a means of analyzing the time variation as well.

3 The ray caster

This renderer is a ray caster that is equipped with a versatile new mechanism for the design of color and opacity transfer functions. These are interactively defined by the user prior to rendering.

Processing commences with some initialization computations. We start by computing the normal vector at each voxel. Unlike traditional 6-neighborhood central difference based gradient calculation, our computation considers all 26 neighbors, attenuating by $\sqrt{2}$ the 18-neighbors and by $\sqrt{3}$ the 26-neighbors. The normal vector $(N_x,\ N_y,\ N_z)$ is used in shading and its magnitude $\nabla = \sqrt{{N_x}^2 + {N_x}^2 + {N_x}^2}$, is utilized to index our 2D opacity transfer function. We calculate the magnitude of the gradient at each voxel location and store them in a 3D byte array of the same size as the original data set.

Given a data set and light source information, we duplicate the original data set and shade the newly generated one. We apply diffuse lighting on the data set using the computed normal at each voxel $(N_x,\ N_y,\ N_z)$ and the raw densities as intensity values ρ. Given m light sources located at direction $L_i = (lx_i,\ ly_i,\ lz_i),\ \ i = 1,\ \ldots,m,$

we set the shaded voxel intensity S to:

$$S = \sum_{i=1}^{m} \rho * (lx_i * N_x + ly_i * N_y + lz_i * N_z + 1)/2$$

which is simply the intensity times cosine the angle between L_i and N (normalized to the range $[0, 1]$). Note that we assume that all light sources have intensity 1.

These preliminary calculations require two additional memory banks the size of the original data set for storing the shaded data set and the gradient information. Alternatively, these computations can be performed on-the-fly, which will save the need for the extra storage and allow view-dependent light effects such as specular light. Once this initialization is completed, the data is ready for rendering.

The ray caster traverses image space. For each sub-sample a ray is shot. Along the ray, at user controlled distances Δt, a trilinearily interpolated sample is taken in each of the three datasets at the sample points $x(t), y(t), z(t)$. These sampled values participate in the illumination calculation to produce the final intensity for that sample point, as described in the next section. After all the samples along a ray are collected and composited, the compositing buffer contains a floating point number representing the intensity at the corresponding image sub-sample. When all image rays are shot, the final image is derived by averaging all sub-samples in a pixel and transforming the floating point intensities into the integer range [0,255].

The ray caster is equipped with an interactive tool, as well as an interpreter that can be used to build 1D and 2D transfer functions. The language supported by this simple modeler contains various boolean operators (min, max, add, subtract, etc.) between several 1D and 2D linear and exponential primitives. Using this tool, the user can define such transfer functions as threshold rendering, iso-value rendering, gradient emphasis by opacity and so on. The user defined transfer functions are evaluated and converted into 1D and 2D tables that are used by the renderer as color and opacity lookup tables. These tables are indexed by raw voxel value, gradient magnitude, and the like, as we describe in the next section.

3.1 Illumination algorithm

Our illumination procedure takes as input a dataset consisting of one byte per voxel intensity and is capable of generating one or three band images. Each band in the image has a pair of transfer functions associated with it:

1. *Voxel value to intensity* transfer function which maps original intensities to desired ones (implemented as a 1D lookup table).
2. *Voxel value and gradient to opacity* which sets the transparency of each voxel during the ray casting process (implemented as a 2D lookup table).

Given a sample triple (ρ, S, ∇), where ρ is the raw intensity, S is the shaded intensity, and ∇ is the gradient, all sampled at $x(t), y(t), z(t)$, we calculate two new values for each color band λ: α_λ, the transparency value and $\hat{\rho}_\lambda$, the intensity value:

$$\begin{aligned} \alpha_\lambda &= TF2D(\nabla, \rho) \\ \hat{\rho}_\lambda &= TF1D(S) \end{aligned}$$

where TF are 1D and 2D transfer functions designed by the user. Although we currently use the above 1D and 2D functions, the same mechanism we developed can be used to functionally bind reflectivity to gradient, for example, and to develop transfer function of higher dimensionality.

The compositing scratchpad consists of a pair (I_λ, O_λ) of float numbers, where I_λ stands for intensity and O_λ for opacity (in each color band λ). Compositing of $(\alpha_\lambda, \hat{\rho}_\lambda)$ onto the compositing scratchpad is performed as follows :

$$\begin{aligned} I_\lambda &= I_\lambda + I_\lambda * \alpha_\lambda * O_\lambda \\ O_\lambda &= O_\lambda * (1 - \alpha_\lambda) \end{aligned}$$

After we composite a ray, I_λ contains the final pixel intensity.

3.2 Results for the ray caster

The dataset was shaded in 5 seconds and gradient calculation took 5 additional seconds. Rendering time for all images (256^2 resolution) is 35 seconds when one ray is traced from each pixel. The transfer function used for rendering Figure 3 was

$$\begin{aligned} TF1D(S) &= S^\mu \\ TF2D(\nabla, \rho) &= (\nabla \cdot \rho)^\nu \end{aligned}$$

In Figure 3a we assigned $\mu = 0.3$ and $\nu = 1.1$. In Figure 3b we assigned $\mu = 0.3$ and $\nu = 0.5$.

Figure 3 shows a typical rendering of the flowfield static pressure generated using the computed Navier-Stokes flowfield data between two adjacent compressor rotor blades. In this figure, the flowfield is oriented such that the flow is entering the blade passage from the left and proceeds to the right and out the back of the page as it exits the blade passage. It should be noted that the data set used in this rendering has been truncated in the vicinity of the blade trailing edge or in the region where the flow leaves the blade passage. The specific transfer function used in this rendering reveals various regions in the flow in which large gradients exist. These are the regions in which abrupt changes are present. Specifically, the rapid change from dark to light near the left hand boundary appears to capture the general shape of the shock wave produced as the supersonic flow is turned when it enters the blade passage. As the flow proceeds toward the exit, the variation in the shading along the boundaries bounded by solid surfaces might be associated with the presence of the boundary layer along the solid surfaces. However, there are other flow quantities, such as velocity, which will reveal much more of the detail of the boundary layer behavior, including its growth and separation.

In the vicinity of the blade trailing edge, there appears to be a region of separated flow which may well be associated with the important phenomena of vortex shedding and interaction. These flow features have been subjected to a preliminary examination using an interactive animation procedure to look at the three-dimensional nature of their behavior at a single instant in time. The extension of the three-dimensional rendering procedure to include the temporal variation will provide even greater detail which will be of value in understanding these flow features.

4 Template-based rendering

In parallel viewing, where the observer is placed at infinity, all rays have exactly the same form. Therefore, there is no need to reactivate a line algorithm for each ray. Instead, we compute the form of the ray once and store it in a data structure called a *ray-template*. All rays can then be generated by following the ray template. This approach has great advantages both in terms of performance and in terms of accuracy, as we show later.

The rays, however, differ in the appropriate portion of the template that should be repeated. We choose a plane, parallel to one of the volume faces, to serve as a base-plane for the template placement. The image is computed by sliding the template along that plane, emitting a ray at each of its pixels. This placement guarantees a complete and uniform tessellation of the volume by 26-rays.

The proposed algorithm is composed of three phases: initialization, ray casting, and 2D mapping. In the first phase, the base-plane is computed and a template in the viewing direction is constructed. In the second phase, a ray is traversed from each pixel inside the image extent by repeating the sequence of steps stored in the ray template. In the last phase, the projected image is mapped from the base-plane onto the screen-plane by employing a 2D image transformation. The proposed algorithm guarantees uniform sampling and is based on simple calculations that form the basis for an extremely efficient implementation. We now turn to describe, in more detail, each of the algorithm's three phases.

The basic algorithm can be slightly modified to support multiple rays emitted from the same pixel, but from different sub-pixel addresses [15]. Instead of having one type of ray (i.e., one template), we now have several templates, one for each relative position of the ray-origin in a pixel. For example, if we want a supersampling rate of four rays per pixel, for the pixel at coordinate (i, j), these four rays will originate from $(i \pm 1/4, j \pm 1/4)$. We observe that for all (i, j) the rays $(i + 1/4, j + 1/4)$ have the same form. Therefore we need only four different templates - one for each relative displacement from the pixel origin.

The template-based algorithm can also be modified to efficiently support multiple samples per voxel [15]. If we assume that rays are emitted from the centers of the pixels in a plane that is parallel to one of the volume faces (such as the base-plane), then all these sample points are in the same relative position inside the voxel; that is, they have exactly the same set of distances to the eight corners of the voxel. Therefore, we can employ a continuous line algorithm to generate a template of floating point steps. For each sample point, we can pre-compute the weight to be assigned to each of the voxel values participating in the sampling operation. This will save the need to compute, for each sample, the weights of all eight voxel values participating in the trilinear interpolation. This template based ray casting algorithm uses very few (eight, on the average) integer additions per sample, instead of the naive implementation which requires tens of floating-point additions and multiplications.

4.1 Illumination algorithm

To benefit from the template algorithm, illumination capabilities were kept to a minimum. Illumination is based on the Phong illumination model with one light source. Samples are shaded on-the-fly and all samples along a ray are incrementally composited. The ray stops when a user defined opacity threshold is reached. Transfer function for color assignment to raw voxel value is a simple linear palette of grey (i.e., $\hat{\rho}_\lambda = \rho$). Transfer function for opacity assignment is also a simple linear palette assigning $\alpha_\lambda = \rho/6$ so that the rays would accumulate less opacity and will not stop in one or two steps. Normal is calculated by central difference in a 6-neighborhood. Shading includes ambient, diffuse, and specular shading.

4.2 Results for the template-based rendering

The gas images shown in Figure 4 were generated by an adaptive variation of the ray template algorithm. This approach renders the empty space using one template (discrete) and another template at the region of the data values. In this area, samples are taken at $\Delta t = 0.7$. The generation of this 140^2 resolution image, took 3.01 seconds for one ray per-pixel, which is approximately 3 times faster than the previous algorithm. Supersampling increases rendering time linearly (e.g., 4-rays per pixel takes 12.1 seconds). At $\Delta t = 0.3$, the image is rendered in 4.11 seconds.

5 Future extensions

There are several extensions that can be made to the current system. First of all, further optimizations of the rendering algorithms are being explored by the authors to increase the performance of the system.

Second, we are developing techniques to support the visualization of unstructured, multi-level, and adaptive grid data [16]. The current system will only allow the display of structured (rectilinear) grid data. Note that illumination models and transfer function design are the same for all types of grids, only stepping along the ray and sampling methods change. Techniques have been developed to resample structured and unstructured grid data to a rectilinear grid for visualization; however, this step may introduce aliasing artifacts into the resulting images. Resampling unstructured or multi-resolution data is also too time consuming to be done in an interactive system. For instance, resampling the curvilinear grid data used in Figures 1 and 2 took several minutes. Direct support for unstructured grids and multi-level solution of adaptive grids is needed. The use of adaptive, multi-dimensional, and unstructured grids has increased greatly in the recent past [17, 18]. These grid structures allow for better utilization of computational power during the CFD simulations and need to be supported in an interactive visualization system. We plan to explore incorporating and enhancing these recent algorithms for unstructured grids into our visualization system. We will also develop algorithms for the direct support of multi-level grids and multi-level solution of adaptive grids.

Finally, a graphical user-interface for the visualization system needs to be developed to increase the usability and interactivity of the system. A significant part of this task involves the development of the interface between the computational model and the visualization system to allow direct interaction with the computations to aid debugging.

6 Conclusion

We have described and compared three new efficient volume rendering techniques for flow visualization and demonstrated their value for visualizing the flow between two blades of a turbo-jet compressor. As can be seen from the resulting images in this paper, these techniques provide higher quality images than most CFD visualization systems. Combining these techniques in a single system can provide CFD researchers with a powerful visualization system. With these tools, researchers can see photo-realistic images of their flows for detailed analysis and also produce high-quality images at interactive rates for testing and debugging of the computational model.

While the renderings we generated of the static pressure provide a good representation of flow features caused by pressure gradients, numerous other flow quantities will provide even better representations of other significant flow features. Thus, these renderings represent a significant step in the development of an important, enabling technology for the analysis of complex flow phenomena encountered in advanced aerospace propulsion systems as well as a wide range of other CFD visualization applications.

References

[1] R. Yagel and A. Kaufman, ''Template-Based Volume Viewing,'' *Computer Graphics Forum*, vol. 11, pp. 153--157, September 1992.

[2] K. Perlin and E. Hoffert, ''Hypertexture. Proceedings of SIGGRAPH'89,'' in *Computer Graphics 23,3*, pp. 253--262, July 1989.

[3] J. J. van Wijk, ''Flow visualization with surface particles,'' *IEEE Computer Graphics and Applications*, vol. 13, pp. 18--24, July 1993.

[4] N. Max, R. Crawfis, and D. Williams, ''Visualization for climate modeling,'' *IEEE Computer Graphics and Applications*, vol. 13, pp. 18--24, July 1993.

[5] D. Ebert and R. Parent, ''Rendering and animation of gaseous phenomena by combining fast volume and scanline A-buffer techniques. Proceedings of SIGGRAPH'90,'' in *Computer Graphics 24,4*, pp. 357--366, August 1990.

[6] D. Ebert, W. Carlson, and R. Parent, ''Solid spaces and inverse particle systems for controlling the animation of gases and fluids,'' *The Visual Computer*, vol. 10, no. 4, pp. 179--190, 1994.

[7] M. Kass and G. Miller, ''Rapid, stable fluid dynamics for computer graphics. Proceedings of SIGGRAPH'90,'' in *Computer Graphics 24,4*, pp. 49--58, August 1990.

[8] G. Gardner, ''Forest fire simulation,'' in *Computer Graphics (SIGGRAPH '90 Proceedings)* (F. Baskett, ed.), vol. 24, p. 430, Aug. 1990.

[9] H. Rushmeier and K. Torrance, ''The zonal method for calculating light intensities in the presence of a participating medium. Proceedings of SIGGRAPH'87,'' in *Computer Graphics 21,4*, pp. 293--302, July 1987.

[10] J. Scott and W. L. Hankey, ''Navier-stokes solutions of unsteady flow in a compressor rotor,'' *ASME Journal of Turbo Machinery*, vol. 108, pp. 206--215, October 1986.

[11] D. S. Ebert, *Solid Spaces: A Unified Approach to Describing Object Attributes*. PhD thesis, The Ohio State University, 1991.

[12] J. Kajiya and B. Von Herzen, ''Ray tracing volume densities. Proceedings of SIGGRAPH'84,'' in *Computer Graphics 18,3*, pp. 165--174, July 1984.

[13] J. Kajiya and T. Kay, ''Rendering fur with three dimensional textures. Proceedings of SIGGRAPH'89,'' in *Computer Graphics 23,3*, pp. 271--280, July 1989.

[14] J. Blinn, ''Light reflection functions for simulation of clouds and dusty surfaces. Proceedings of SIGGRAPH'82,'' in *Computer Graphics 16,3*, pp. 21--29, July 1982.

[15] R. Yagel, ''High quality template-based volume viewing,'' Tech. Rep. OSU-CISRC-10/92-TR28, Department of Computer and Information Science, The Ohio State University, 2036 Neil Ave, Columbus, Ohio 43210-1277, October 1992.

[16] R. Yagel, ''Volume rendering polyhedral grids by incremental slicing,'' Tech. Rep. OSU-CISRC-10/93-TR35, Department of Computer and Information Science, The Ohio State University, 2036 Neil Ave, Columbus, Ohio 43210-1277, October 1993.

[17] H. Neeman, ''A decomposition algorithm for visualizing irregular grids,'' in *Computer Graphics (San Diego Workshop on Volume Visualization)*, vol. 24, pp. 49--56, Nov. 1990.

[18] M. P. Garrity, ''Raytracing irregular volume data,'' in *Computer Graphics (San Diego Workshop on Volume Visualization)*, vol. 24, pp. 35--40, Nov. 1990.


Figure 1: Gas Rendering of static pressure.

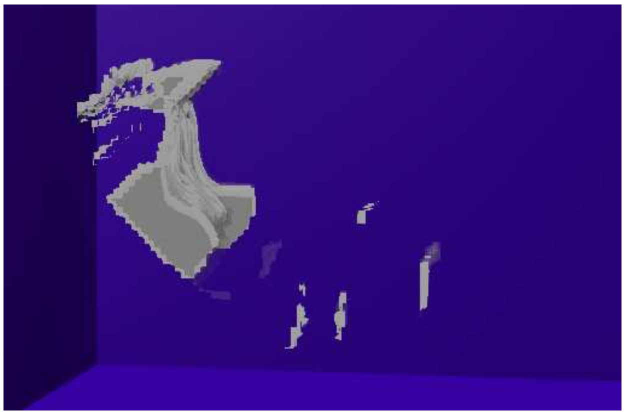
Figure 2: Traditional isosurface volume rendering of static pressure.

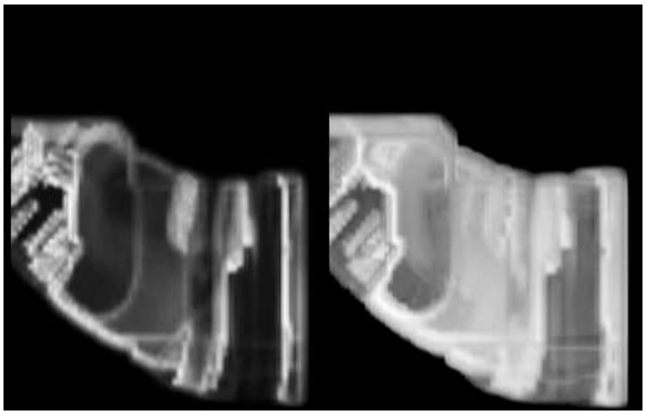
Figure 3: Changing opacity transfer function to emphasize gradient

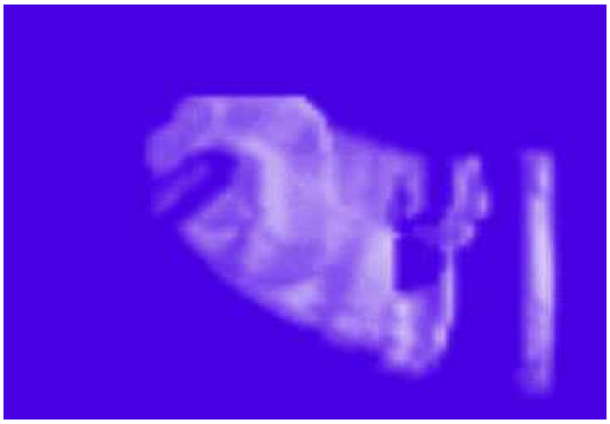
Figure 4: Adaptive template–based rendering

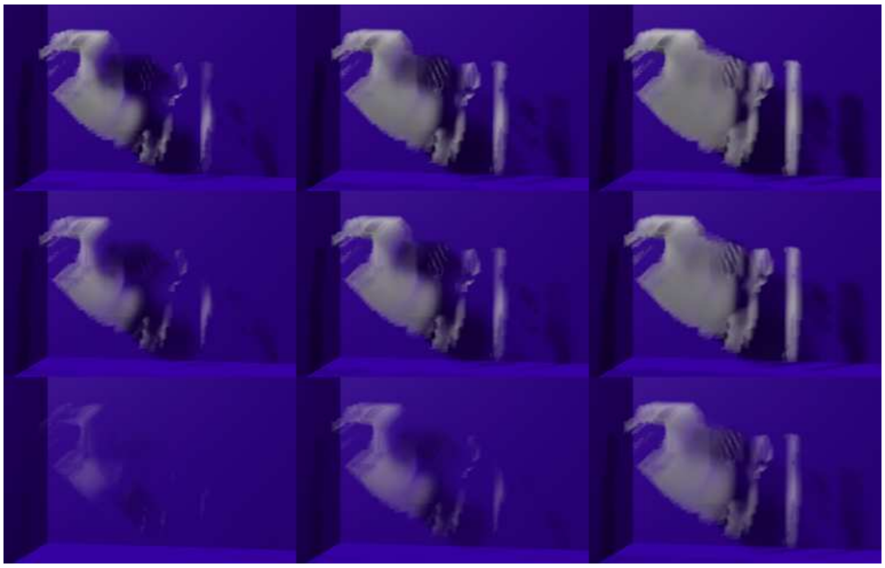
Figure 5: Varying density scaling and power exponent values.