PROCEEDINGS, Fire and Evacuation Modeling Technical Conference 2011
Baltimore, Maryland, August 15-16, 2011

A NOTE ON VISUALIZING SMOKE AND FIRE

Glenn Forney
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
e-mail: glenn.forney@nist.gov

ABSTRACT

This note documents how the radiation transport equation (RTE) and associated numerical algorithms are used by Smokeview to visualize smoke and fire. This equation models the interaction between light and smoke due to absorption, emission, and scattering. By noting that typical smoke due to fire has low albedo, the RTE may be approximated resulting in simpler solution procedures. Two methods are discussed for visualizing smoke, both using a form of the RTE. The first method, slice rendering, uses a series of partially transparent slices to represent smoke and fire. The second method, volume rendering, solves the RTE over the entire domain. Both methods use the RTE to account for extinction (absorption plus out-scattering) by the smoke and emission from the fire. Several areas where the visualization methods may be improved are given.

1 Introduction

This note documents the physics and associated numerical algorithms used by Smokeview [1] to visualize smoke and fire. Smokeview is a companion to the Fire Dynamics Simulator (FDS) [2] and CFAST [3], fire models which generate simulation data Smokeview uses to perform these visualizations. FDS is a computational fluid dynamics model of fire-driven fluid flow. FDS numerically solves a form of the Navier-Stokes equations appropriate for low-speed, thermally-driven flow with an emphasis on smoke and heat transport from fires. CFAST is a zone fire model used to calculate the evolving distribution of smoke, fire gases and temperature throughout compartments of a constructed facility due to a fire.

Realistic visualization of fire calculation methods are important for applications where one wishes to observe qualitative effects of fire and smoke rather than determine quantitative characteristics of data such as temperature or velocity. This would be the case for a fire fighter using a computer based fire fighting simulator. Realistic visualization methods, however, complement but do not replace other more traditional visualization methods such as 2D contouring or 3D iso-surfacing which are better suited for quantitatively analyzing data.

Complete methods for visualizing smoke and fire data taking into account interactions between light and smoke require the solution of the radiation transport equation (RTE) [4] also called the volume rendering equation in visualization literature. [5] This equation models how light is affected after interacting with smoke, a participating media. In particular, Smokeview uses the RTE to account for extinction (absorption plus out-scattering) by the smoke and emission from the fire. The RTE used by Smokeview to model smoke and fire appearance is identical to that used by FDS to model radiative heat transfer. Smokeview and FDS use this equation in different ways, however. FDS solves it at infrared wavelengths of light. Generally, Smokeview solves the RTE for visible wavelengths of light. With the proper extinction coefficient, however, Smokeview can also view smoke at other wavelengths, simulating a thermal imager, for example. Smokeview solves the RTE assuming a gray gas environment. This is the default solution method for FDS. One other important difference is that Smokeview requires a solution at only one point at a time (any arbitrary point though), the observer's viewpoint, while FDS requires a radiation field, a solution at all points within the solution

domain. Even so, the solution of this integro-differential equation for visualization requires significant computation and memory resources. Approximations are required in order to display smoke and fire at interactive frame rates. The primary approximation is to take advantage of the low albedo character of smoke allowing one to either simplify or eliminate scattering terms in the RTE.

Two techniques discussed for visualizing smoke are slice rendered and volume rendered methods [5, 6]. These methods both solve a form of the RTE equation. The integrated quantity in both cases is radiance, the intensity of light seen by the observer. They differ in how the integration path is partitioned.

The first approach, a slice rendering method, splits the integration path at grid planes within a 3D mesh. There is one partially transparent slice for each plane of simulated data. Planes are drawn through the data along the coordinate planes or diagonally to these planes. The particular plane drawn is chosen to be the one most perpendicular to the viewer's line of sight. The resulting partially transparent slice planes are drawn individually and combined by the video hardware to form one image.

The separation distance between slice planes becomes smaller as more planes are used to simulate a case. As a result, the computed opacity values are subject to increased round off error due to finite precision arithmetic. In fact, if these planes are sufficiently close, the computed opacities truncate to zero. In this situation, volume rendering methods are required.

The second approach, a volume rendering method, also integrates a simplified form of the radiation transport equation, but over the entire data mesh. The integration occurs from the front of the solution domain (relative to the observer) to the back, rather than across just one slice plane. By performing the entire integral at once instead of in pieces, the volume rendered method does not suffer from the round off errors of the slice rendered method. The intermediate computational terms in the volume rendered method are stored using full precision arithmetic rather than 8 bits. A volume rendered method then computes opacity across multiple grid planes. As a result, there is only one plane of data displayed. The video hardware is again exploited, but this time to compute a line integral for each pixel in the observer's view. Opacities and color for both methods are computed using transfer functions using soot density and temperature data obtained from a fire simulation.

This note is organized as follows. A model for visualizing smoke and fire is discussed. This model, the radiative transport equation (RTE), is simplified in several ways, one of which gives the Beer-Lambert law. Two methods are then discussed for visualizing smoke both using a form of the RTE. The first method, slice rendering, uses a series of partially transparent slices to represent smoke and fire. The second method, volume rendering, solves the RTE over the entire domain. Finally, several areas where the visualization methods may be improved are given.

2 Radiation Transport Equation

The model used here to visualize smoke is the radiation transport equation (RTE) [4]. This equation uses radiance to represent smoke appearance. Radiance has units of Watts per square meter per unit solid angle ($W/(sr \cdot m^2)$). The solid angle accounts for the fact that a light source appears brighter if it emits a given amount of light through a smaller cross-sectional area. This has the interesting implication, ignoring atmospheric effects and as pointed out in Ref. [7], that the sun's radiance appears the same observed from Earth as from Mars. The diminished heat flux (W/m^2) on Mars due to the increased distance from the sun is exactly offset by the reduced solid angle (sr) that the sun's disk subtends. From a visualization perspective, this implies that image radiance does not depend on distance from the observer unless participating media is present to absorb or scatter light. The radiation transport equation discussed in this section models the change in radiance due to these factors.

The radiation transport equation is used to calculate radiance due to one or more light sources within a region possibly containing a participating media such as smoke [4]. The change in radiance along a ray with direction ω at any one instant and wavelength may be expressed using

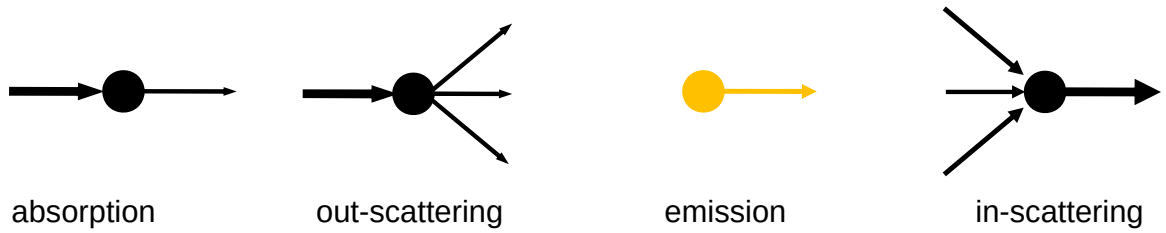


Figure 1: Components of the radiation transport equation decreasing radiance along a ray are absorption and out-scattering while components increasing radiance are emission and in-scattering.

$$(\omega \cdot \nabla) C(x,\omega) = -\underbrace{\sigma_a(x)C(x,\omega)}_{\text{absorption}} - \underbrace{\sigma_s(x)C(x,\omega)}_{\text{out-scattering}} + \underbrace{\sigma_a(x)C_e(x,\omega)}_{\text{emission}} + \underbrace{\sigma_s(x)\int_{4\pi} p(x,\omega,\omega')C_i(x,\omega')d\omega'}_{\text{in-scattering}} \tag{1}$$

where $C(x,\omega)$ represents the radiance at x along a direction ω. As illustrated in Figure 1, the right hand side of (1) is split into four components accounting for absorption, in and out scattering and emission where $\sigma_a(x)$ is the absorption coefficient, $\sigma_s(x)$ is the scattering coefficient, $C_e(x,\omega)$ is the radiance emitted at x along a direction ω and $p(x,\omega,\omega')$ is the fraction of light moving along direction ω' scattered along direction ω. Absorption and out-scattering cause radiance to decrease while emission and in-scattering cause radiance to increase. The radiance terms C, C_e and C_i have units of W/m^{2}sr^{-1}. The coefficients σ_a and σ_s have units of m^{-1} and specify the time and location dependent change per unit length to the radiance term to which they are applied.

2.1 Approximating the Radiation Transport Equation

The RTE may be simplified in several ways depending on which terms are included or ignored. This section derives a solution to one approximation where the in-scattering integral term in (1) is neglected and the absorption and out-scattering coefficients are combined (both are loss terms) using $\sigma_t(x) = \sigma_a(x) + \sigma_s(x)$. This simplification then only includes interactions between light and smoke due to absorption, out-scattering and emission. Note that the Beer-Lambert law results if the emission term is also dropped.

Equation (1) is then approximated by neglecting the integral term and using $\sigma_t(x) = \sigma_a(x) + \sigma_s(x)$ to obtain

$$\begin{aligned} \frac{dC}{dx}(x) &= -\sigma_t(x)C(x) + \sigma_a(x)C_e(x) \\ C(x_0) &= C_0 \end{aligned}$$

This equation may be solved by rearranging terms and applying the integrating factor $\exp(\int_{x_0}^{x} \sigma_t(s)ds)$ obtaining

$$\begin{aligned} \exp\left(\int_{x_0}^{x} \sigma_t(s)ds\right)\left(\frac{dC}{dx}(x) + \sigma_t(x)C(x)\right) &= \exp\left(\int_{x_0}^{x} \sigma_t(s)ds\right)\sigma_a(x)C_e(x) \\ \frac{d}{dx}\left(\exp\left(\int_{x_0}^{x} \sigma_t(s)ds\right)C(x)\right) &= \exp\left(\int_{x_0}^{x} \sigma_t(s)ds\right)\sigma_a(x)C_e(x) \end{aligned}$$

Integrating both sides and substituting the integration limits results in

$$\begin{aligned}
\exp\left(\int_{x_0}^{x}\sigma_t(s)ds\right)C(x)\Big|_{x_0}^{x_N} &= \int_{x_0}^{x_N}\exp\left(\int_{x_0}^{x}\sigma_t(s)ds\right)\sigma_a(x)C_e(x)dx \\
\exp\left(\int_{x_0}^{x_N}\sigma_t(s)ds\right)C(x_N)-C_0 &= \int_{x_0}^{x_N}\exp\left(\int_{x_0}^{x}\sigma_t(s)ds\right)\sigma_a(x)C_e(x)dx
\end{aligned}$$

Solving for $C(x_N)$ after noting that $\int_{x_0}^{x}\sigma_t(s)ds-\int_{x_0}^{x_N}\sigma_t(s)ds=-\int_{x}^{x_N}\sigma_t(s)ds$ results in

$$C(x_N) = \exp\left(-\int_{x_0}^{x_N}\sigma_t(s)ds\right)C_0+\int_{x_0}^{x_N}\exp\left(-\int_{x}^{x_N}\sigma_t(s)ds\right)\sigma_a(x)C_e(x)dx$$

which may be simplified to

$$C(x_N)=\tau(x_0,x_N)C_0+\int_{x_0}^{x_N}\tau(x,x_N)\sigma_a(x)C_e(x)dx \tag{2}$$

after defining $\tau(a,b)$ as

$$\tau(a,b)=\exp\left(-\int_{a}^{b}\sigma_t(s)ds\right) \tag{3}$$

which represents the optical depth between a and b. As noted earlier, if the emission term is neglected and $\sigma_t(x)=\sigma_t$ is constant over a path with length $L=x_N-x_0$, then (2) simplifies to

$$\frac{C(x_N)}{C_0}=\exp(-\sigma_t L)$$

which is the Beer-Lambert law.

2.2 Discretizing the Radiation Transport Equation

The approximate RTE solution given in (2) is discretized by converting integral terms into Riemann sums. Figure 2 illustrates the terms used to perform these discretizations. The path is split into N parts each with length $\Delta x=(x_N-x_0)/N$. The coordinate system is setup so that the initial radiance, C_0, is located at x_0, most distant from the observer and the final radiance, C_N, is located at x_N closest to the observer.

The optical depth, $\tau(a,b)$, defined in (3) is discretized using a Riemann sum after defining sample points $s_j=x_0+j\Delta s$ for $j=0$ to N with spacing $\Delta s=(x_N-x_0)/N$ to obtain

$$\begin{aligned}
\tau_i^{N-1}=\tau(x_i,x_N) &= \exp\left(-\int_{x_i}^{x_N}\sigma_t(s)ds\right)\approx\exp\left(-\sum_{j=i}^{N-1}\sigma_t(s_j)\Delta s\right) \\
&= \prod_{j=i}^{N-1}\exp\left(-\sigma_t(s_j)\Delta s\right)=\prod_{j=i}^{N-1}\tau_j
\end{aligned}$$

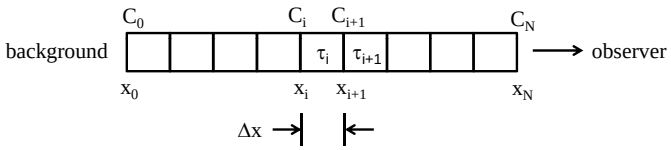


Figure 2: Setup for discretizing the equations used to model radiance within a column of 3D smoke data. The transparency across the interval from x_i to x_{i+1} is τ_i. The transparency across the intervals from x_i to the observer is the product of individual transparencies or $\tau_i\tau_{i+1}\cdots\tau_{N-1}$

where $\tau_j = \exp(-\sigma_t(s_j)\Delta s)$ represents the transparency over one discretization interval. For $i = N-1$ to 0, the optical depth τ_i^{N-1} may be computed recursively using

$$\tau_i^{N-1} \quad = \quad \tau_{i+1}^{N-1}\tau_i \tag{4}$$

where the recursion is initiated with $\tau_N^{N-1} = 1$. Substituting $1-\alpha_i^{N-1} = \tau_i^{N-1}$ and $1-\alpha_i = \tau_i$ into (4) gives

$$1-\alpha_i^{N-1} = (1-\alpha_{i+1}^{N-1})(1-\alpha_i) = 1-\alpha_{i+1}^{N-1}-\alpha_i+\alpha_{i+1}^{N-1}\alpha_i$$

which simplifies to

$$\alpha_i^{N-1} \quad = \quad \alpha_{i+1}^{N-1}+(1-\alpha_{i+1}^{N-1})\alpha_i \tag{5}$$

Similarly, the radiance given by the RTE solution $C(x_N)$ in (2) may be discretized to obtain

$$C_N = \tau_0^{N-1}C_0 + \sum_{i=0}^{N-1}\tau_i^{N-1}\sigma_{a,i}C_{e,i}\Delta x$$

where $\sigma_{a,i} = \sigma_a(x_i)$, $C_{e,i} = C_e(x_i)$, $x_i = x_0 + i\Delta x$ and $\Delta x = (x_N - x_0)/N$. This simplifies to

$$C_N = \tau_0^{N-1}C_0 + \sum_{i=0}^{N-1}\tau_i^{N-1}\hat{C}_{e,i} \tag{6}$$

where $\hat{C}_{e,i} = \sigma_{a,i}C_{e,i}\Delta x$. If $\hat{C}_{e,i}$ is interpreted as the emitted color of the fire or heated gas at location i and C_0 is interpreted as the color of the light source *behind* the smoke, then (6) restated in words gives color seen by the observer computed as a weighted average of source and emitted colors where each weight is the optical depth from the observer to the corresponding color location. These emitted colors can be determined from a black body temperature curve or from a colormap meant to show variations in temperature in terms of color.

The terms in (6) are summed from back to front meaning that the location of the $i = 0$ term is farthest from the observer, while the location of the $i = N-1$ term is closest. We wish to perform this sum in reverse order, from front to back so that the sum may be terminated early if additional contributions would not significantly change the result.

Therefore, to compute C_N, let $\hat{C}_j^N$ denote the partial sum using terms $i = j$ through $i = N-1$ in the summation term in (6). Using this notation $\hat{C}_N = \tau_0^{N-1}C_0 + \hat{C}_0^N$. Then

$$\hat{C}_j^N \quad = \quad \sum_{i=j}^{N-1}\tau_i^{N-1}\hat{C}_{e,i} \tag{7}$$

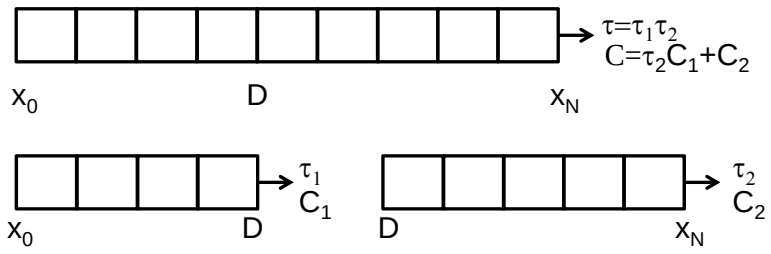


Figure 3: Solutions to the RTE on two sub-intervals are combined to form a solution to the RTE for the merger of these intervals.

$$\hat{C}^N_{j+1} = \sum_{i=j+1}^{N-1} \tau_i^{N-1} \hat{C}_{e,i} \tag{8}$$

Subtracting (8) from (7) and solving for $\hat{C}^N_j$ results in

$$\hat{C}^N_j = \hat{C}^N_{j+1} + \tau_j^{N-1} \hat{C}_{e,j} \tag{9}$$

The strategy then for volume rendering an image is for each pixel in the 2D projected image to

1. convert the background radiance C_0 to a color,
2. convert the emitted radiances along the integration path to colors and
3. form a weighted average of these these colors using either equation (6) or (9) where the weights are optical depths obtained using (5) .

The conversion from radiance to color may be based on a black body temperature curve if realistic flame colors are the goal or arbitrary if only a qualitative view of the fire is the goal. Equations (5) and (9) are equivalent to the recursions presented in Ref. [8, Chapter 39] for performing volume rendering.

2.3 Splitting the Radiation Transport Equation

This section discusses splitting the RTE, so that it may be solved separately on multiple meshes or multiple slice planes. The final solution is then obtained by appropriately combining solutions obtained on each mesh or slice plane.

It is more practical to draw smoke one mesh at a time when visualizing multiple mesh cases since required data may not be easily accessible from all meshes, especially if the GPU[1] is used for drawing. The RTE solution or equivalently the computed radiance and smoke opacity require properties for the entire line of sight which may encompass more than one data mesh. This section discusses how radiance and opacity may be computed by combining solutions from each individual mesh along the line of sight. For example, as illustrated in Figure 3, consider an interval $[x_0, x_N]$ that is split at D into two sub-intervals $[x_0, D]$ and $[D, x_N]$. The goal then is to compute a radiance and opacity for $[x_0, x_N]$ using radiances and opacities computed on the two sub-intervals.

Let C be a radiance computed on $[x_0, x_N]$ and C_1 and C_2 be radiances computed on two sub-intervals of $[x_0, x_N]$. Likewise, let τ be an optical depth computed on $[x_0, x_N]$ and τ_1 and τ_2 be optical depths computed on two sub-intervals of $[x_0, x_N]$. From (2) and (3), the radiance C and optical depth τ are given by

[1] graphics processing unit of the video card

$$
\begin{aligned}
C &= \int_{x_0}^{x_N} \tau(x,x_N)\sigma_a(x)C_e(x)dx \\
\tau &= \tau(x_0,x_N)
\end{aligned}
$$

Likewise, the radiances C_1 and C_2 and optical depths τ_1 and τ_2 for the intervals $[x_0,D]$ and $[D,x_N]$ are given by

$$
\begin{aligned}
C_1 &= \int_{x_0}^{D} \tau(x,D)\sigma_a(x)C_e(x)dx \\
C_2 &= \int_{D}^{x_N} \tau(x,x_N)\sigma_a(x)C_e(x)dx \\
\tau_1 &= \tau(x_0,D) \\
\tau_2 &= \tau(D,x_N)
\end{aligned}
$$

The optical depth τ can be split into two parts giving

$$
\tau = \tau(x_0,x_N) = \tau(x_0,D)\tau(D,x_N) = \tau_1\tau_2
$$

since from (3) it can be shown that for any x, $\tau(a,b) = \tau(a,x)\tau(x,b)$. The radiance C can also be split into two parts giving

$$
\begin{aligned}
C &= \int_{x_0}^{x_N} \tau(x,x_N)\sigma_a(x)C_e(x)dx \\
&= \int_{x_0}^{D} \tau(x,x_N)\sigma_a(x)C_e(x)dx + \int_{D}^{x_N} \tau(x,x_N)\sigma_a(x)C_e(x)dx \\
&= \tau(D,x_N)\int_{x_0}^{D} \tau(x,D)\sigma_a(x)C_e(x)dx + \int_{D}^{x_N} \tau(x,x_N)\sigma_a(x)C_e(x)dx \\
&= \tau_2 C_1 + C_2
\end{aligned}
$$

Summarizing, full-interval values C and α may be written in terms of sub-interval values C_1, C_2, α_1 and α_2 using

$$
\tau = \tau_1\tau_2 \tag{10}
$$

$$
C = \tau_2 C_1 + C_2 \tag{11}
$$

Equations (10) and (11) may then be used to draw smoke and fire one mesh at a time. These two equations are also the basis for combining RTE solutions obtained on multiple slice planes, which is discussed next.

3 Slice Rendering

A slice rendering algorithm for visualizing smoke consists of splitting the RTE across individual slice planes within a single mesh. The 3D computational domain is partitioned into a series of 2D slices. The RTE is then solved on each slice. Each slice solution only accounts for conditions between adjacent slices. The individual partially transparent slice solutions are then combined using video hardware to form the final image. Problems can occur with numerical

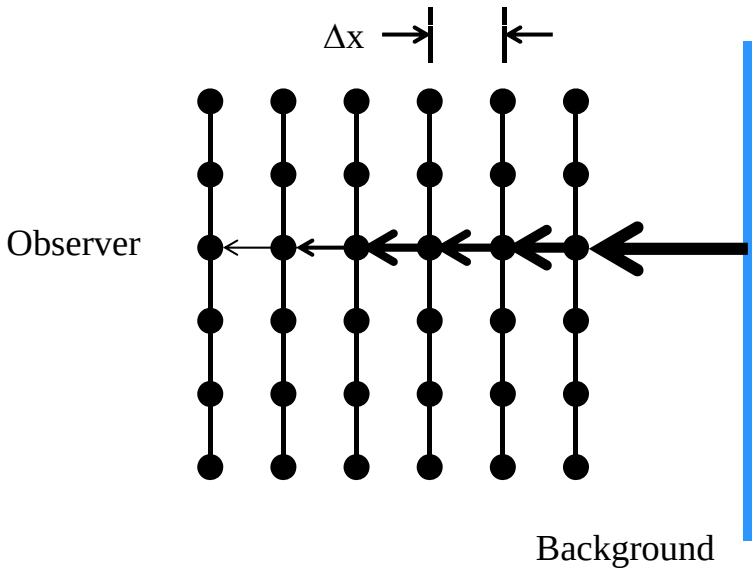


Figure 4: Light emitted from the background is obscured by intervening smoke.

round off error if two slices are too close together which require solution methods involving data volumes rather than data slices. These methods are discussed in the next section.

There are many ways to slice a 3D data set. The slice orientation is chosen to be the one most perpendicular to the viewer's line of sight where possible choices are slice planes parallel to the three cartesian coordinate planes (XY, XZ, YZ) or planes diagonal to the data. The opacity at each grid node is computed using the distance Δx between adjacent YZ planes and soot density data computed by the fire model. If slice orientations other than YZ are displayed, then opacities are adjusted if the distance between planes is different than Δx. Opacity data is computed and compressed using run length encoding as a preprocessing step and decompressed one frame at a time as data is displayed.

3.1 Computing Opacity

Computing opacity at slice plane nodes is illustrated in Figure 4. A ray travels from the background to the observer through intervening smoke. Light is absorbed or scattered by the smoke as the ray passes each slice plane . Emission effects are accounted for by the coloring the smoke. Scattering effects presently are only accounted for in the value of the total mass extinction coefficient. Light losses are assumed to be from both absorption and scattering. The obscuration is computed along each ray one grid plane at a time, using the Beer-Lambert law as follows. The $\alpha = 1 - \tau$ values are pre-computed by FDS using the Beer-Lambert law [4]

$$\alpha = 1 - \exp(-\sigma_t \Delta x) \tag{12}$$

for a particular view direction (down the x axis) where Δx is this distance between two grid planes and as before $\sigma_t = \sigma_a + \sigma_s$ is the total mass extinction coefficient. The Beer-Lambert law is an empirical relationship relating light absorption to the material properties of the media the light is travelling through, in this case soot or smoke.

The α parameter in (12) is used by OpenGL [9] to blend smoke planes with the current background. The α parameter used here also represents an opacity, 0.0, for completely transparent, and 1.0 for completely opaque.

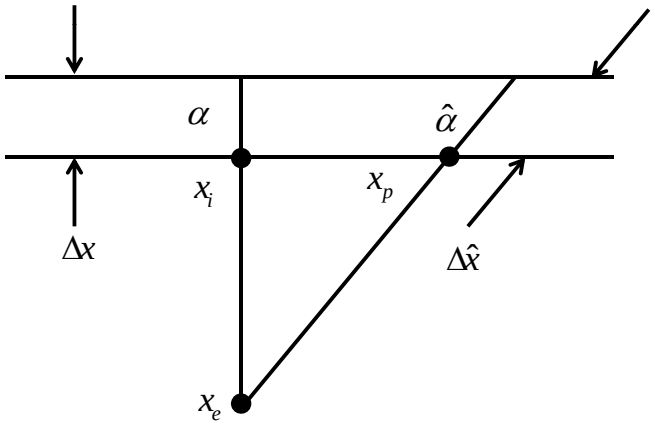


Figure 5: Diagram illustrating the adjustment required to the opaqueness parameter, α, for non axis aligned views. The adjusted opacity $\hat{\alpha}$ along the $\Delta\hat{x}$ segment is related to α value along the Δx segment using $(1-\hat{\alpha}) = (1-\alpha)^{\Delta\hat{x}/\Delta x}$

3.2 Adjusting Opacity

The absorption parameter, α, needs to be adjusted for view directions not aligned with the axis orthogonal to the viewing planes (see Figure 5). The absorption coefficient also needs to be adjusted when the distance between adjacent smoke planes changes, or viewing planes are skipped.

Ten million exponential operations per second are required to display smoke with corrected α's at 10 frames per second if the simulation has grid dimensions of $100 \times 100 \times 100$. Recent advances in CPU and video hardware makes these types of visualizations possible. These corrections may also be performed in the video card (GPU), resulting in increased display rates because the GPU performs the corrections simultaneously at all or many of the grid nodes rather than one at a time as the CPU would.

The α obscurations are pre-computed using the distance Δx between adjacent planes along the x-axis. The adjusted $\hat{\alpha}$ expressed in terms of $\Delta\hat{x}$ is given by

$$\hat{\alpha} = 1 - \exp(-\sigma_t \Delta\hat{x}) \tag{13}$$

where $\Delta\hat{x}$ is the distance between planes along the line of site. Equations (12) and (13) may be used to solve for $\hat{\alpha}$ in terms of α to obtain

$$\hat{\alpha} = 1 - (1-\alpha)^{\Delta\hat{x}/\Delta x} \tag{14}$$

after noting that

$$1 - \hat{\alpha} = \exp(-\sigma_t \Delta\hat{x}) = \exp(-\sigma_t \Delta x)^{\Delta\hat{x}/\Delta x} = (1-\alpha)^{\Delta\hat{x}/\Delta x}$$

The computation of (14) is expensive because the exponential is computed at each grid node for every time step. In addition, numerical cancellation may occur for small α leading to loss of significant digits. Both problems may be solved by expanding (14) in a Taylor series and keeping only the first few terms:

$$\hat{\alpha} \approx \alpha r - \frac{\alpha^2}{2} r(r-1) + \frac{\alpha^3}{6} r(r-1)(r-2)$$

where $r = \sec(\theta) = \Delta\hat{x}/\Delta x = ||x_p - x_e||/n \cdot (x_p - x_e)$, n is the unit vector normal to the current plane being drawn, θ is the angle between the view direction and n, x_e is the observers position and x_p is the vertex being drawn (along the view direction). These terms are illustrated in Figure 5.

When planes are skipped, (14) may be simplified. In particular, when every 2nd plane is skipped, $\Delta\hat{x}/\Delta x = 2$, so that (14) simplifies to

$$\hat{\alpha} = 1 - (1-\alpha)^2 = 2\alpha - \alpha^2$$

The video hardware uses α values contained in the smoke planes to obscure the background much like a camera uses a neutral density filter to darken a scene. Extending the analogy, Smokeview uses one spatial/time varying *numerical* neutral density filter for each plane of smoke data. On a node by node basis then, each smoke plane obscures the current image stored in the OpenGL back buffer by the amount $(1-\alpha)$ to form a new back buffer image. Figure 6 illustrates this process showing several snapshots of a fire plume. The final image in the lower right is the most realistic. A simplistic description of one step of this process is given by

$$\text{new buffer image} = (1-\alpha) \times \text{old buffer image}$$

This process is repeated for each smoke plane.

Figure 7 illustrates this process showing smoke and fire in a townhouse kitchen fire. The visualization is performed by displaying a series of partially transparent planes. For illustration, these planes are made more conspicuous (in Figure 7a) by skipping smoke planes (displaying every third plane) and orienting them along the 'x' axis. Figure 7b shows the visualization as it normally appears with all slice planes shown and oriented along a plane most perpendicular to the view direction.

3.3 Orienting Slice Planes

Smoke opacity data computed as described in the previous sections is stored in a 3D array. This array corresponds to the solution domain as set up in an FDS input file (or some other model). Smoke planes are drawn in Smokeview through this data. The orientation is chosen to be most perpendicular to the viewer's line of sight. A plane orientation exactly perpendicular to the view direction could be drawn if one is willing to pay the added CPU cost of interpolating opacity values between grid nodes.

Figure 8 illustrates this process showing three view directions and the corresponding smoke plane orientations that would be used. Off-axis viewing is minimized by selecting the view planes orientation that minimizes the angle between the planes normal direction and the view direction. This angle, θ, is illustrated in Figure 9, and is given by

$$\cos(\theta) = \frac{n \cdot v_e}{||n||\ ||v_e||}$$

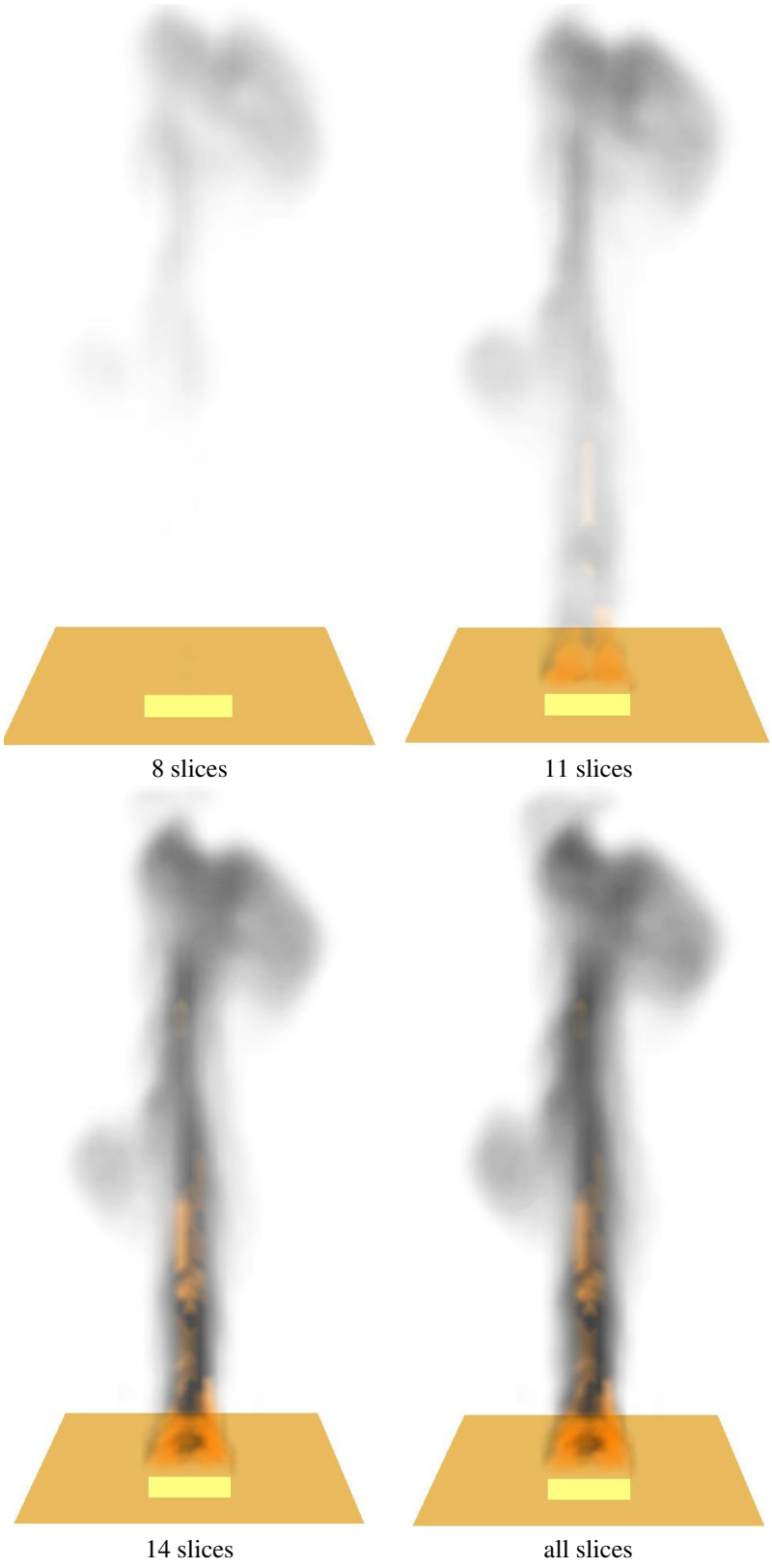
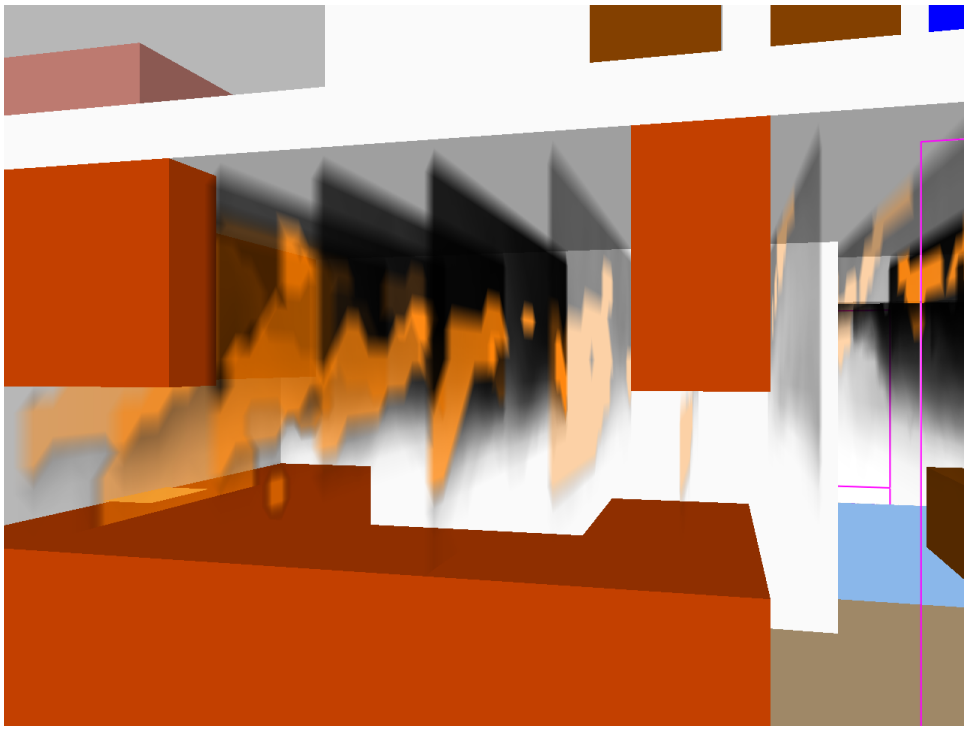
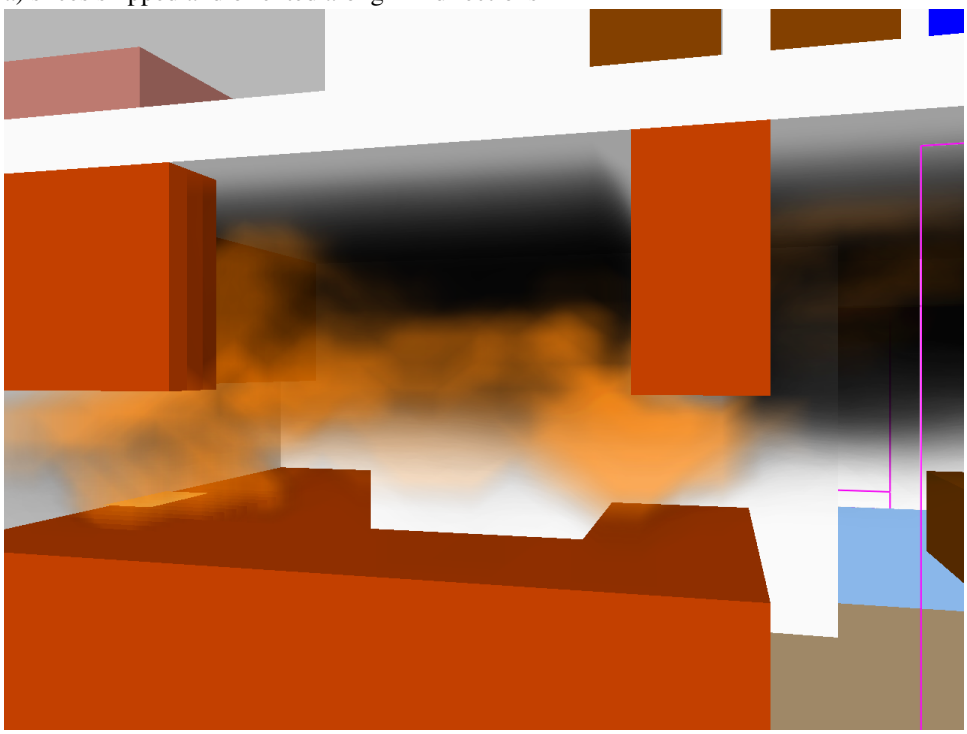


Figure 6: Smoke plume visualized using several vertical parallel partially transparent planes. The smoke plume looks more realistic as more slice planes are included to form the image.

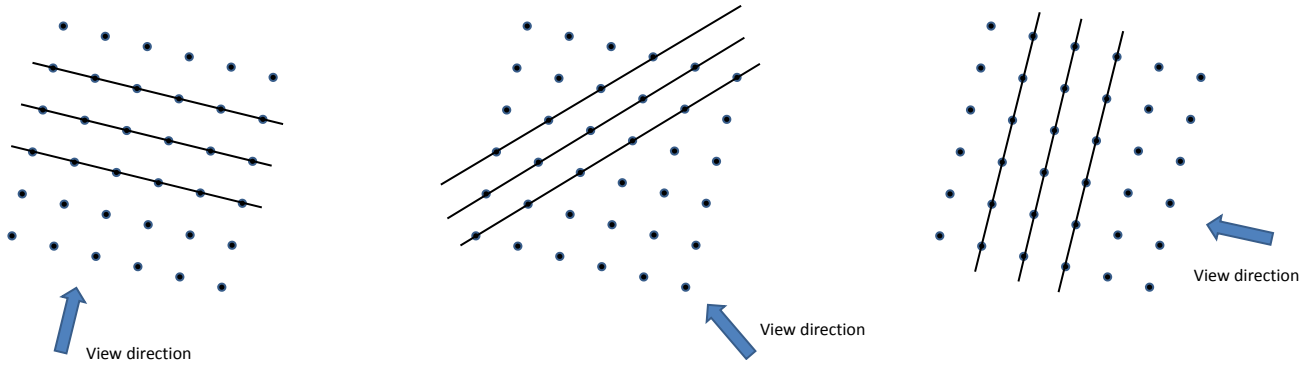


a) slices skipped and oriented along ‘X’ directions



b) all slices shown and oriented towards viewer

Figure 7: Realistic visualization of a townhouse kitchen fire simulated using FDS. For illustrative purposes, planes in the top image are oriented along the X axis. Planes in the bottom image are aligned along the Y axis, the axis most perpendicular to the line of sight.



a) slice planes parallel to the y axis b) slice planes parallel to the $y = x$ axis c) slice planes parallel to the x axis

Figure 8: Slice plane orientation chosen to be *most perpendicular* to the line of sight

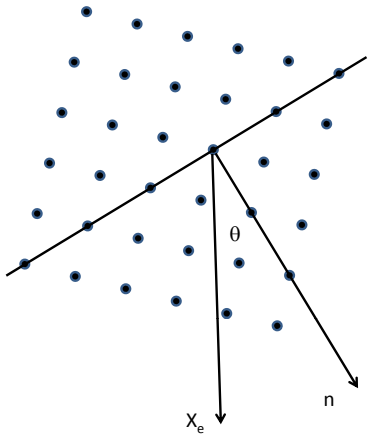


Figure 9: Diagram illustrating the angle between the line of sight and the vector normal to the slice planes. Slice plane orientation is chosen to minimize this angle.

where n is normal vector for the candidate smoke plane, and v_e is the view direction vector. In OpenGL, the view direction vector, v_e, is computed by simply obtaining the OpenGL modelview matrix, M and multiplying it by the vector, $(0,0,1)^T$ or equivalently the third row of M.

3.4 Compressing Slice Plane Data

The opacity parameters are computed at each slice plane node for all time steps. The space required to store these values can easily become quite large. Compression techniques are required to reduce storage requirements.

Storage reduction occurs in two steps. First, four byte floating point soot densities are converted to one byte smoke opacities using the Beer-Lambert law. Video cards presently use only one byte to represent opacity. Next, the sequence of opacity values are compressed using run-length encoding, a compression scheme where repeated "runs" of data are replaced with a number (number of repeats), and the value repeated. In more detail,

1. Represent four or more consecutive identical characters as #*nc* where # is a special character denoting the beginning of a repeated sequence, *n* is the number of repeats and *c* is the character repeated. *n* can be up to 254 (255 is used to represent the *special* character).

2. Represent characters not repeated four or more times as is.

The character string `aaaaaabbbbcc` would then be encoded as `#6a#4bcc`.

Run length encoding provides a reasonably good compression ratio, is simple to implement and more importantly can be decompressed quickly. This last property is important for any compression scheme chosen because it is a rate limiting step in the process that Smokeview uses to display smoke data. The CPU time required to compute the smoke flow can easily exceed one minute of CPU time per output time step, so extra time used to produce a more compact file is affordable. However, each data frame is decompressed *on the fly* so a compression format that can be rapidly decompressed is critical.

A second compression scheme is used by Smokezip, companion software to FDS and Smokeview, to more compactly compress FDS files. Smokezip uses the ZLIB compression library [10].

3.5 A Limitation of the Slice Solution Method

As noted earlier, the slice rendering method for visualizing smoke records opacities on grid planes using

$$\alpha = 1 - \exp(-\sigma_t \Delta x) \tag{15}$$

where again Δx is the distance between adjacent grid planes. Problems with round off error can occur because α's are stored using only 8 bits, the size typical video hardware uses to represent color and alpha channels. As a result, opacity values are quantized. Only values of 0, 1/255, $\cdots$, 254/255 and 1 can be represented. In particular, any α value computed to be smaller than $1/255$ will truncate to 0. This can occur when more grid cells are used to resolve a solution domain, *i.e.* when Δx is sufficiently small so that $\alpha < 1/255$.

For example, suppose that a 1 m column of smoke has opacity of 0.5. Substituting $\Delta x = 1$ and $\alpha = 1/2$ into (15) gives $\sigma_t = -\ln(1/2)$. Solving for Δx using this value of σ_t and $\alpha = 1/255$ gives

$$\Delta x = \frac{\ln(254/255)}{\ln(1/2)}$$

Therefore, the opacity, α, will truncate to zero (for this example) whenever $\Delta x < \ln(254/255)/\ln(1/2) \approx 0.00567$. Equivalently, the opacity truncates to zero when $N > \ln(1/2)/\ln(254/255) \approx 177$, where N is the number of grid planes in the 1 m column of smoke. Smoke drawn in this situation will be *invisible*.

To overcome this problem, the RTE needs to be solved across the full 3D data mesh where intermediate smoke opacities are stored using full precision arithmetic. One such algorithm is discussed in the next section.

4 Volume Rendering

Volume rendering is the process of visualizing 3D data by projecting partially transparent colors derived from 3D data onto a 2D plane forming an image. An entire volume of data is used to generate an image rather than just a slice as in the previous section. Transfer functions or colormaps are used to map data to color and optical density (opacity). In this application temperature is mapped to color and soot density is mapped to opacity. These colors and opacities are then combined to form an image. The transfer functions may be arbitrary designed to highlight certain portions of the data or based on physics designed to produce realistic appearing images. Figure 10 illustrates this process. Soot density is mapped to opacity. The colors and opacities are then combined to form an image. The strategy then is to perform color mixing in the same way that light would behave by solving an approximate form of the radiation

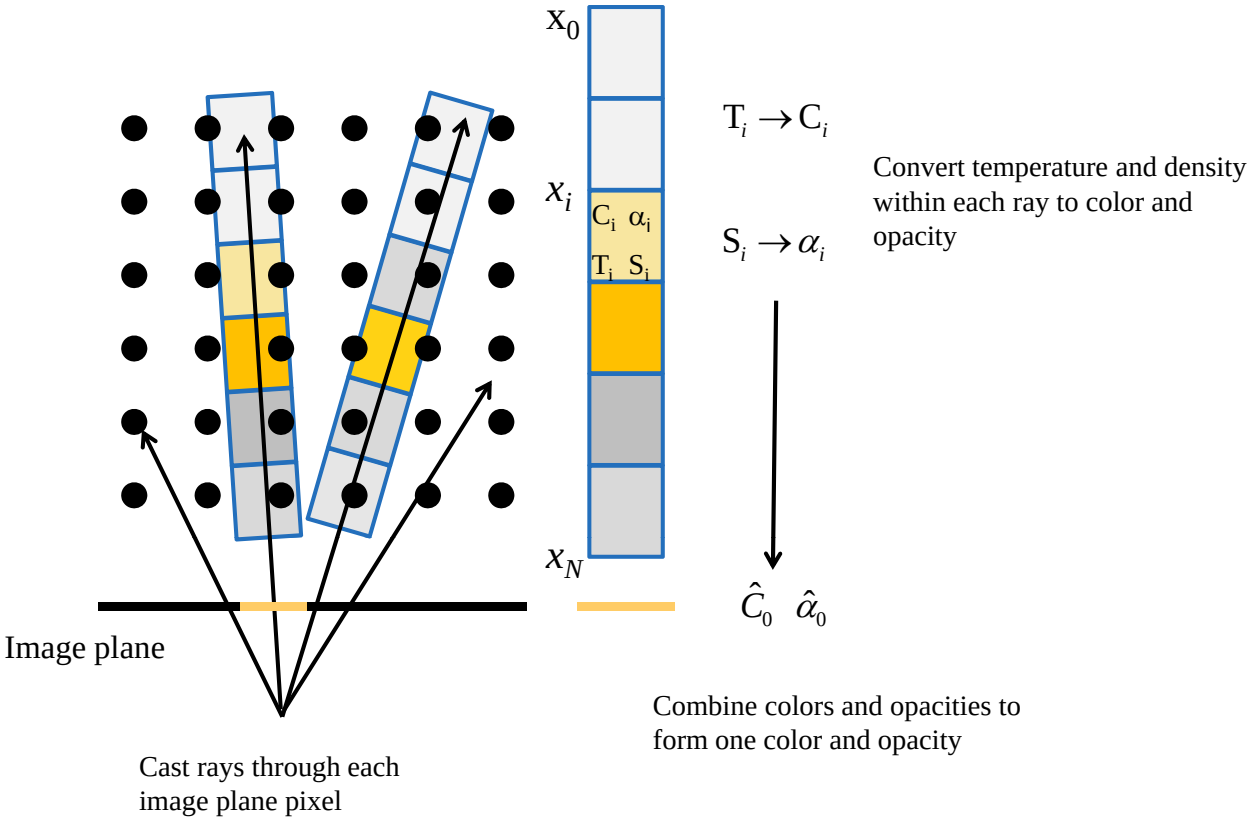


Figure 10: Opacity and color is computed for each pixel on an image plane by solving a line integral representation of a simplified form of the radiation transport equation. Rays are cast from the observer through the image plane into the 3D data set converting temperatures to color and soot densities to opacity. The line integral is computed for each pixel where the ray intersects the data set.

transport equation. The approximate RTE solution given in (2) is used to collapse 3D data into 2D images projected onto the sides of the mesh visible to the observer.

Figure 11 illustrates the effect of projecting an image onto a plane. The top center image is presented from the point of view of the observer. The perspective is correct. The two images below show the same image but from a different viewpoint. The scene is rotated left and right to show more clearly the images projected on the front facing sides of the data mesh. As the scene is moved through rotations and translations the projected surface images are constantly recomputed and redrawn presenting the illusion that the image is 3D and drawn within (rather than on the surface) of the data mesh.

Three example colormaps are illustrated in Figure 12. The black/orange colormap in Figure 12a is used to produce an image. If the temperature is below a user specified value, the color is black, representing soot (where emission is not significant). If the temperature is above this value then it is a shade of orange, representing emission from a fire. The particular shade chosen is based on the color emmitted by a black body source. The black to white colormap in Figure 12b along with an extinction coefficient appropriate for infrared could be used to simulate the effect of a thermal imager by using black for cool temperatures and white for warm temperatures. The rainbow colormap in Figure 12c produces non-realistic images which can still be of value for highlighting data of interest.

4.1 Implementation

Data required to volume render smoke is made available to Smokeview by adding lines to an FDS input file specifying 3D slice files for temperature and soot density such as

```
&SLCF XB=xmin,xmax,ymin,ymax,zmin,zmax, QUANTITY='TEMPERATURE' /
&SLCF XB=xmin,xmax,ymin,ymax,zmin,zmax, QUANTITY='DENSITY',SPEC_ID='SOOT' /
```

where xmin, $\cdots$, zmax represent the domain boundary.

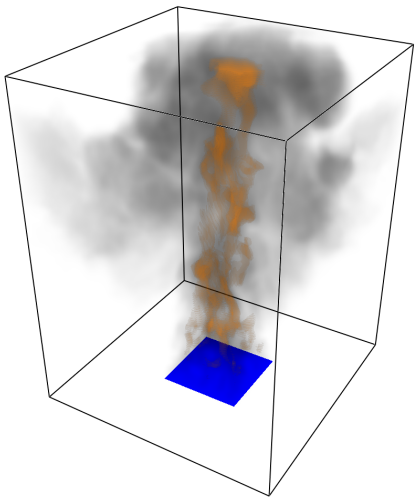
An algorithm for determining image opacities and colors using volume rendering is detailed below.

1. For each pixel in the image plane, cast a ray from the observer's viewpoint through that pixel into the 3D data set
2. Step along the ray from the front (relative to the observer) to the back of the 3D data set converting data values along the way to color and opacity. Choose a step size (possibly varying) to capture changes occurring in the data set.
3. Combine the colors and opacities found in step 2 using recursions equations (5) and (9). Initiate the recursion with $\hat{\alpha}_N = \hat{C}_N = 0$. Continue the recursion for $i = N-1$ to $i = 0$ by computing $\hat{\alpha}_i$ and $\hat{C}_i$ using:

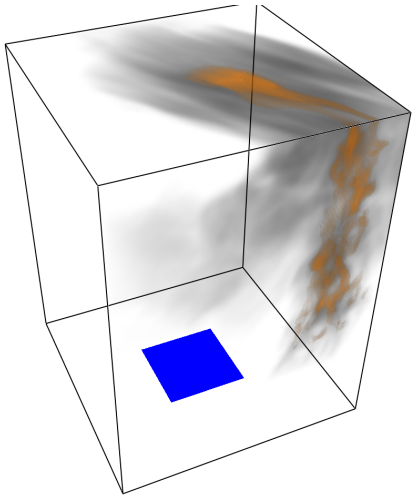
$$\begin{aligned} \hat{\alpha}_i &= \hat{\alpha}_{i+1} + (1-\hat{\alpha}_{i+1})\,\alpha_i \\ \hat{C}_i &= \hat{C}_{i+1} + (1-\hat{\alpha}_{i+1})\,C_i \end{aligned}$$

   where $\hat{C}_i$ and $\hat{\alpha}_i$ are color and opacity accumulated from steps N to i while stepping through the 3D data set from front to back. When data is mapped appropriately to color and opacity, this recursion is simply a numerical integration of the radiation transport equation. The values C_i and α_i are the color and opacity at the i'th step.
4. Mix the volume rendered color, $\hat{C}_N$, with the colors already rendered using equation (11) re-written as

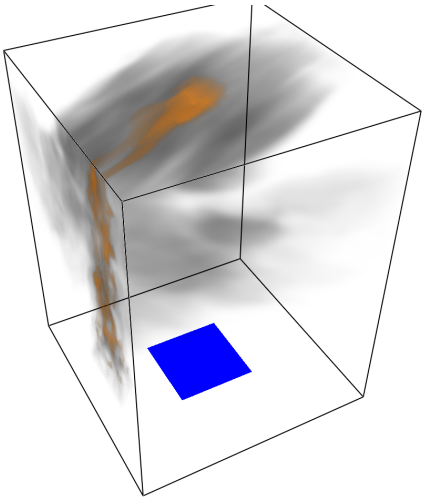
$$\text{updated background color} = (1-\hat{\alpha}_N) \times \text{original background color} + 1 \times \hat{C}_N$$



a) normal view as seen by the observer



b) same image as in a) but viewed from the left



c) same image as in a) but viewed from the right

Figure 11: Volume rendered smoke plume projected onto the outside surfaces of the data mesh and shown from several points of view. The image, in a), is as viewed by the observer. The other two images, b) and c), are rotated versions of the image as in a).

a) black stepped to orange

b) black to white

c) rainbow

Figure 12: Example colormaps used for converting temperature to color.

The OpenGL call, `glBlendFunc(GL_ONE,GL_ONE_MINUS_SRC_ALPHA);` , is used to implement this mixing mode in Smokeview.

A limitation of the volume rendering procedure is the large file sizes required to store the full precision being used in computations. For example, smoke flow computed on a $128 \times 128 \times 128$ mesh for 1024 time steps requires 16 gigabytes to store data for visualization. As with the slice rendering method, compression procedures are implemented using the ZLIB library [10] to reduce this size.

5 Future Work

This note describes the algorithms Smokeview uses to display smoke and fire using physics based algorithms. These algorithms may be improved in several ways. Presently, only radiation from soot is used to visualize smoke. Radiation emissions from heated gas components such as carbon dioxide and water vapor are neglected. The gray gas assumption may be relaxed by solving the RTE for several wavelength bands (as may be done by FDS) and combining the results. The RTE line integration needs to terminate at the first solid object encountered (FDS OBST) rather than the far side of the data mesh. The computational efficiency may be improved by implementing algorithms to skip over regions with little or no smoke. Research on unstructured geometries for future incorporation into FDS may also lead to better visualization. Finally, the transfer function relating temperature to color may be improved by using a physics based approach rather than an assumed color map.

References

[1] G.P. Forney. Smokeview (Version 5), A Tool for Visualizing Fire Dynamics Simulation Data, Volume I: User's Guide. NIST Special Publication 1017-1, National Institute of Standards and Technology, Gaithersburg, Maryland, August 2007.

[2] K.B. McGrattan, S. Hostikka, J.E. Floyd, W.E. Mell, and R. McDermott. Fire Dynamics Simulator, Technical Reference Guide, Volume 1: Mathematical Model. NIST Special Publication 1018, National Institute of Standards and Technology, Gaithersburg, Maryland, October 2007.

[3] W. W. Jones, R. D. Peacock, G. P. Forney, and P. A. Reneke. CFAST, Consolidated Model of Fire Growth and Smoke Transport (Version 6. technical reference guide. NIST Special Publication 1026, National Institute of Standards and Technology, Gaithersburg, Maryland, April 2009.

[4] Robert Siegel and John R. Howell. *Thermal Radiation Heat Transfer*. Taylor & Francis, Inc., New York, NY, 4th edition, 2001.

[5] Marc Levoy. Display of surfaces from volume data. *IEEE Comput. Graph. Appl.*, 8(3):29–37, 1988.

[6] Klaus Engel, Markus Hadwiger, Joe M. Kniss, Christof Rexk-Salama, and Daniel Weiskopf. *Real-Time Volume Graphics*. A. K. Peters, Wellesley, Massachussets, 2006.

[7] Philip Dutre, Kavita Bala, and Philippe Bekaert. *Advanced Global Illumination*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

[8] Randima Fernando. *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*. Pearson Higher Education, 2004.

[9] The Khronos OpenGL ARB Working Group Dave Shreiner. *OpenGL Programming Guide - The Official Guide to Learning OpenGL, Version 3.0 and 3.1*. Addison-Wesley, Upper Saddle River, NJ, 7 edition, 2009.

[10] Jean loup Gailly and Mark Adler. zlib version 1.1.4, http://zlib.net/, November 2002.