Volume rendering of smoke propagation CFD data

Oliver Staubli[1] Christian Sigg[1] Ronald Peikert[1] Daniel Gubler[2] Markus Gross[1]

[1]ETH Zürich* [2]Air Flow Consulting†

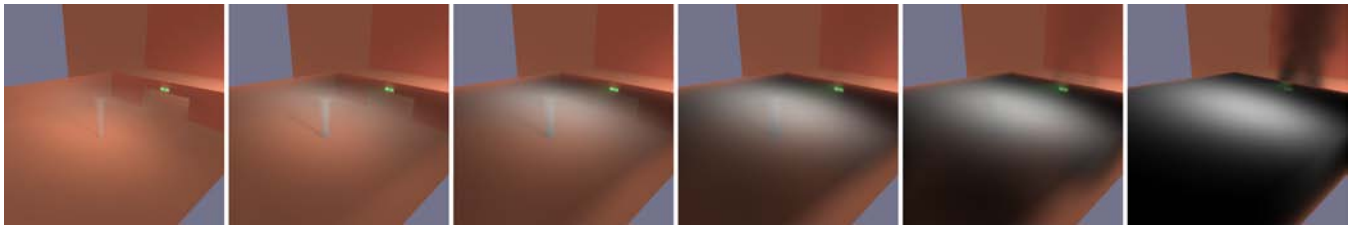

Figure 1: Six timesteps of smoke emission from a fire inside a hall. The smoke propagation is simulated by computational fluid dynamics and lit with an absorbtion and limited diffuse scattering-based lighting model. Illuminated exit signs shine through dense smoke, which is modeled by adjusting the transfer function in front of exit signs.

Abstract

The evacuation of buildings in the event of a fire requires careful planning of ventilation and evacuation routes during early architectural design stages. Different designs are evaluated by simulating smoke propagation using computational fluid dynamics (CFD). Visibility plays a decisive role in finding the nearest fire exit. This paper presents real-time volume rendering of transient smoke propagation conforming to standardized visibility distances. We visualize time dependent smoke particle concentration on unstructured tetrahedral meshes using a direct volume rendering approach. Due to the linear transfer function of the optical model commonly used in fire protection engineering, accurate pre-integration of diffuse color across tetrahedra can be carried out with a single 2D texture lookup. We reduce rounding errors during framebuffer blending by applying randomized dithering if high accuracy frame buffers are unavailable on the target platform. A simple absorption-based lighting model is evaluated in a preprocessing step using the same rendering approach. Back-illuminated exit signs are commonly used to indicate the escape route. As light emitting objects are visible further than reflective objects, the transfer function in front of illuminated exit signs must be adjusted with a deferred rendering pass.

CR Categories: I.3.8 [Computer Graphics]: Applications; J.2 [Applications]: Physical Sciences and Engineering—Engineering.

Keywords: volume rendering, flow visualization.

1 Introduction

Every year, 4 million fires occur worldwide. As a result, 15,000 people are killed and damages totalling 70 billion USD are incurred.

Contrary to what one might think, the deaths are rarely caused by the flames. In most cases, the victims are poisoned by the smoke. The smoke not only poisons people, it also causes the majority of the property damage, estimated at 70%. Therefore, fire safety has become more and more important during the architectural planning of a building. In the event of a fire the first priority is to remove the smoke from the building, thereby enabling the people to quickly and safely find the fire exits. In addition, the rescue of people trapped within the building and effective fire fighting is dependant on the removal of smoke.

To assess smoke removal concepts and to prove the efficiency of smoke removal measures to guarantee a safe evacuation and a safe operation of the fire rescue staff, the visibility in a building is a key parameter. In response to market demands for improved fire safety, a new modeling technique for fire and smoke simulation has been developed. The technique is based on Computational Fluid Dynamics (CFD). For smoke propagation inside closed buildings, buoyancy of hot air and heat dissipation are important aspects of the simulation. Based on several simulated fire scenarios, the ventilation can be optimized to guarantee effective removal of smoke. The output of a CFD fire-simulation is a distribution of smoke concentrations within the room. Using today's standard post-processing software (e.g. Fluent, Ansys CFX, Ensight), it is possible to visualize the distribution of smoke concentration as color coded slices through the volume data or as an iso-surface. Two examples are shown in Figure 2.

The distance of visibility in uniformly distributed smoke is directly coupled to the smoke concentration. The iso-surface rendering clips the smoke at a certain level (e.g. visibility of more than 10m), which only gives a very crude approximation of real visibility for two reasons. First, visibility will still be limited outside the isosurface and second, the visibility is evaluated only locally assuming uniform smoke distribution. An assessment of the visibility in various directions, considering different lighting situations or illuminated signs etc. along an escape route is not possible.

Also, the interpretation of iso-surface illustration or contour plots is very difficult for non-experts, such as architects. The goal of this project was to apply well-known methods to visualize complex CFD data from a commercial CFD code with volume rendering and to create a simple and productive software tool.

We use an adoption of unstructured direct volume rendering, which incrementally composes transmittance and colors in visibility order to the final image using the computational power of modern graphics cards. Direct volume rendering has proven to be an effective rendering technique for volumetric data in many different fields of application. One of the few areas where direct volume rendering does not play a major role yet is flow visualization. Reasons for this can be found in the lack of clearly separated structures, in

*e-mail:staubli@itdesign.ch, sigg,peikert,grossm@inf.ethz.ch

†e-mail:gubler@afc.ch

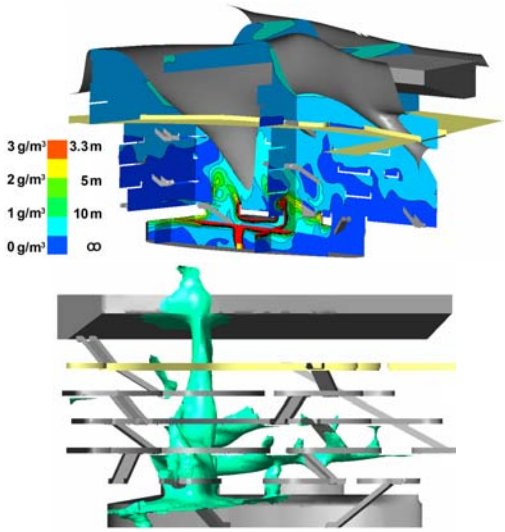

Figure 2: Current visualization techniques of smoke concentration inside a mall. Top: contour plot of slices through smoke concentration and corresponding local visibility distance. Bottom: iso-surface of smoke concentration $1g/m^3$. Inside this surface the visibility is less than $10m$, outside it is more than $10m$ but not unlimited.

the predominance of vector-valued data and in the more abstract meaning of the transfer functions. The visualization of smoke concentration data, however, is a case where direct volume rendering is an immediate and natural approach.

The input data from the numerical simulation consists of time dependent smoke densities on unstructured grids of tetrahedra. While the optical model defines a mapping from smoke concentration data to opacities, the color of smoke does not effect visibility distances and can be used for improving appearance. Our focus is not realism, but efficient processing of large time-dependent datasets with a simple lighting model. A fraction of light traversing smoke is absorbed and scattered as diffuse color, depending on the particle concentration. Note that we do not consider indirect lighting, which would lead to disproportionate complex computations for scientific visualization. The light model is evaluated in a preprocessing step for each mesh vertex at every time step.

2 Related Work

Flow volumes [9] provide a smoke generator for flow fields using direct volume rendering to visualize results of a CFD simulation. Our flow field is coupled to the placement of the fire, which is also the source of smoke. Therefore, smoke concentration is computed during numerical simulation instead of during a post processing step.

In contrast to previous work on realistic modeling and rendering of smoke and clouds [3, 4, 16], the objective of our work is not realism, but a technical visualization of given smoke concentration data under the constraints of a standardized optical model commonly used in fire protection engineering [12]. The optical model gives a fixed translation from smoke concentration data to opacities. Assuming linearly varying smoke concentration within each mesh cell, we get a opacity transfer function completely specified by the optical model, which can be pre-integrated and stored in a two dimensional lookup table [2]. In contrast, more general pre-integration methods of piecewise linear color transfer functions require a three dimensional lookup table, which consume considerable larger amount of memory even at low sampling rates [10, 13, 19].

One common approach of volume rendering on unstructured grids is to store the mesh as a texture on the graphics card and run a fragment program per visible pixel which accumulates the colors along the corresponding viewing ray [20]. However, fragment program limitations usually require the use of temporary rendering buffers storing intermediate results and frequent context switching can slow down performance if not implemented carefully. A more natural approach for graphics hardware is the algorithm of projected tetrahedra [17]. The basic idea is to project the tetrahedra onto the image plane in visibility order and accumulate the final color using framebuffer blending. In order to interpolate the parameters of the pre-integration lookup specified at the vertices, the tetrahedron has to be split into three or four sub-tetrahedrons. This can also be done in the vertex shader [22], but then the computation is discarded after every frame even for static cameras, which produces unnecessary overhead for animation of time dependent data. Interpolation, pre-integration sampling and framebuffer blending are all prone to artifacts. Our renderer employs the techniques presented in [6] to keep artifacts at a minimum level.

3 Optical Model used for Smoke

For evacuating in smoke, the visibility of way guidance markings and signage is critical. It has been shown [5] that the crucial factor is viewing distance, much more than power or luminance. The actual maximum distance for visibility of an object through smoke depends on properties of the smoke, but also of the object itself and the lighting conditions. For practical purposes such as architectural specifications, a simpler empirical model is used [12] where visibility distance is defined as the reciprocal value of the optical density of the smoke, implying a simple emission-absorption optical model [8]. The visibility distance is the distance at which the light is reduced to 10% by absorption and scattering, or at which an opacity of 90% is attained.

The optical density of smoke can be described [12] by a linear function of the particle concentration c_p:

$$D = \frac{K_m}{3} y_s c_p \tag{1}$$

The optical density D is proportional to the more common *extinction coefficient* $\tau = D \ln 10$. The quantity K_m is the light absorption of soot. Typical values are $K_m = 7.6 \frac{m^2}{g}$ for flaming burn and $K_m = 4.4 \frac{m^2}{g}$ for pyrolysis. The soot yield y_s is the mass ratio of soot particles per total particles, which is material dependent value. The combined factor $(K_m/3)y_s$ is called the smoke potential. Example values are shown in Table 1.

Material	flaming burn	non-flaming burn
Fibre insulation board	0.6	1.8
Chipboard	0.37	1.9
Hardboard	0.35	1.7
Birch plywood	0.17	1.7
External plywood	0.18	1.5
α-Cellulose	0.22	2.4
Rigid PVC	1.7	1.8
Extruded ABS	3.3	4.2
Rigid PU foam	4.2	1.7
Flexible PU foam	0.96	5.1
Plasterboard	0.042	0.39

Table 1: Smoke potentials for various materials. The unit used here is $\frac{dBm^2}{g}$. Table reprinted from [1].

The *opacity* α is the fraction of light lost by absorption and scattering over a path length L along a viewing ray. *Transmittance* $T = 1 - \alpha$ is the remaining fraction and can be expressed as:

$$T(s) = 10^{-\int_0^s D(t)dt} = e^{-\int_0^s \tau(t)dt} \tag{2}$$

For given constants K_m and y_s, Eq. 1 defines an opacity transfer function, mapping smoke concentration to optical densities or extinction coefficients. In principle, the constants are given by the conditions of the CFD simulation, so the opacity transfer function is completely determined by our application of visualizing smoke concentration data. However, in practice, the smoke potential is not known exactly. Also, CFD simulations [14] are valid for a certain range of smoke potentials. Therefore, the smoke potential can be used as a parameter for explorative visualization. The color of the smoke is evaluated with a simple light model. A fraction of the light E traversing smoke is scattered as diffuse color $C = \tau E$. The volume rendering integral describes the color of one pixel, which is the diffuse color accumulated along one viewing ray:

$$c = \int C(s)T(s)ds = \int E(s)\tau(s)T(s)ds \tag{3}$$

Assuming linear interpolation within each tetrahedron, the volume rendering equation can be split into piecewise linear segments. For one such segment of length L, extinction and light color within the segment can be written as

$$E(s) = (1 - \frac{s}{L})E_0 + \frac{s}{L}E_1 \tag{4}$$

$$\tau(s) = (1 - \frac{s}{L})\tau_0 + \frac{s}{L}\tau_1 \tag{5}$$

where E_0 and E_1 (τ_0 and τ_1) are the light colors (extinctions) at the segment borders. For one segment, Eq. 3 can now be converted to a weighted sum of the light colors.

$$c = \int_0^L E(s)\tau(s)T(s)ds = w_0E_0 + w_1E_1 \tag{6}$$

with weights $w_{0,1}$ depending on the segment length and the extinctions at the segment borders:

$$\begin{aligned} w_0 &= L\int_0^1((1-s)\tau_0 + s\tau_1)e^{-L\int_0^s(1-t)\tau_0+t\tau_1 dt}(1-s)ds \\ w_1 &= L\int_0^1((1-s)\tau_0 + s\tau_1)e^{-L\int_0^s(1-t)\tau_0+t\tau_1 dt}sds \end{aligned} \tag{7}$$

Analytical integration of Eq. 7 is not possible because they contain Gaussian integrals. However, they can be pre-computed numerically and stored in a lookup table. Two parameters are sufficient to reference weights for any triple $\{\tau_0, \tau_1, L\}$ (for example $\{\tau_0 L, \tau_1 L\}$).

4 Rendering Approach

For each frame, we first render the opaque walls of the building. The geometry is lit by a simple approach as described in Section 4.3, which reuses the lighting of smoke. Next, the smoke is rendered with the algorithm of projected tetrahedra. For a given viewpoint, the tetrahedra are first sorted in visibility order, described in the next section. Each tetrahedron is then rendered as a set of triangles, as described in Section 4.2. Section 4.3 covers the approach used to light smoke by a set of spotlights.

4.1 Visibility sorting

For correct blending, the tetrahedra are sorted back-to-front by their distance to the camera position, which produces less precision artifacts than front-to-back ordering, see end of Section 4.2 for details. Front-to-back rendering allows early ray termination, which locks pixels once they exceed a certain opacity level and thus avoids computations with neglectible impact on the final image [7]. Back-to-front rendering only allows culling of cells against opaque walls. However, smoke configurations that completely occlude a large fraction of cells are rare in our application.

We are using an adoption of the XMPVO [18] algorithm to sort tetrahedra according to the distance to the camera position. First, we build a set of visibility relations between neighboring tetrahedra. The orientation of the common face of two neighboring tetrahedra with respect to the camera position determines the visibility relation. Next, we run a topological sort, which is essentially a breadth first search. If the mesh contains disconnected components or is generally non-convex, the weak ordering of face orientation make a complete sorting impossible. To obtain a complete sorting, ambiguities are resolved by comparing the distances between the camera position and the centers of the tetrahedra. Note that this approach does not guarantee exact results if visibility cycles are present in the data. Visibility cycles could be resolved by subdividing tetrahedrons, but are ignored in our implementation because we never noticed any spurious artifacts.

Because only tetrahedra intersecting the view frustum need to be sorted to achieve a correct rendering, an implementation could use a visibility acceleration structure (like BSP trees) to determine the subset of tetrahedra which need to be sorted. However, we chose to sort all tetrahedra because often only a small fraction lies outside of the view frustum. Thus, querying an acceleration structure does not pay off. Moreover, if all tetrahedra are sorted, the user can rotate and zoom the camera at a fixed position without performing additional sorting.

The depth of the thick vertex can be chosen arbitrarily. If set to a convex combination of the corner positions, the depth test can be used to cull cells that lie behind opaque walls. For each thick vertex, the extinction $\tau_{0,1}$ and light color $E_{0,1}$ at the front and back face has to be computed. The coefficients of the corresponding linear interpolation as well as the projection class can be computed from four determinants defined by the three vectors from the eye point to the vertices of each face of the tetrahedron. The class depends only on the signs of these determinants: class 1 has an odd number of positive signs, while class 2 has an even number. To animate time dependent data, projection class and interpolation weights are reused as long as the camera position stays fixed. Figure 3 shows the vertex data which is stored in a vertex buffer object on the graphics card.

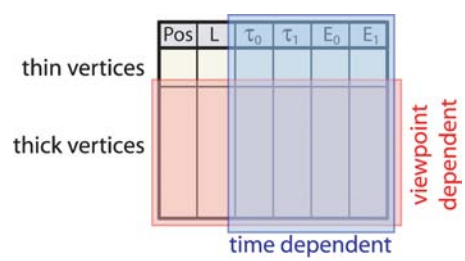

Figure 3: Vertex data of the tetrahedral mesh. Thick vertices are generated by the view dependent splitting of each mesh cell. Each vertex stores its position, ray segment length, extinction and light color at the front and back face. For thin vertices, $L = 0$, $\tau_0 = \tau_1$ and $e_0 = E_1$ holds. The linear array is stored in a vertex buffer object on the graphics card.

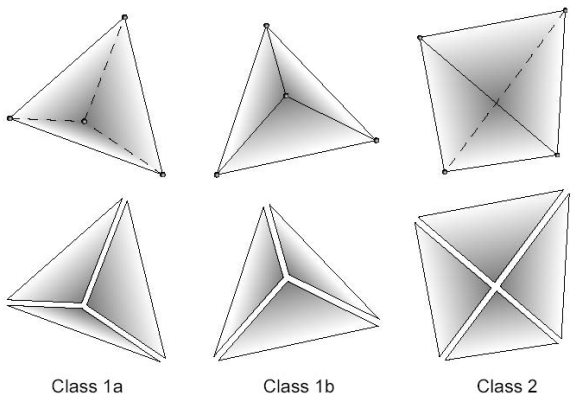

Figure 4: Classification of the projected cell outlines.

4.2 Cell projection

The following section explains the rendering of the sorted tetraedra. The first stage splits each tetrahedron in three or four sub-tetrahedra so that the screen space projection can be broken up into triangles with no triangular region crossing an edge of the tetrahedron. Then, special perspective interpolation has to be used to concurrently interpolate extinction and light color at the front and back plane of each sub-tetrahedron during rasterization. A fragment program performs the color and transmittance integration along the viewing ray segment inside the tetrahedron using a pre-integration approach [2]. Fragment blending incrementally composes the final color for each pixel, employing the depth sorting of the tetrahedra.

Cell Splitting.

Graphics hardware is optimized for triangle rasterization and supports correct perspective interpolation of linear functions within triangles. However, the parameters of Eq. 3 have to be interpolated on four front or back planes. Thus, each tetrahedron is split into sub-tetrahedra so that each of them projects to a triangle in image space. The screen space projection of one tetrahedron can be classified into three cases, depending on the orientation with respect to the image plane. Figure 4 shows the (non-degenerate) cases that can occur, resulting in fans of either three or four triangles. The center vertex is called the thick vertex, the outer vertices the thin vertices. All parameters of Eq. 7 vary linearly in perspective coordinates of the triangles and can be interpolated from the vertex values by the rasterization process.

Perspective Interpolation.

The rasterization process performs perspective linear interpolation of vertex values in screen space. For any linear function f in object space, $\frac{f}{w}$ is a linear function in screen space, where w is the homogeneous clipping coordinate. Internally, the rasterization process linearly interpolates $\frac{f}{w}$ and $\frac{1}{w}$, from which the function f is reconstructed for every pixel and passed to the fragment shader. For the algorithm of projected tetrahedra, perspective interpolation has to be carried out concurrently for the front and the back face of one tetrahedron, while the triangle actually drawn lies in the middle between those two faces. If we denote values of front, middle and back face with x_0, x and x_1 respectively, the vertex program computes $f = \frac{f_i w}{w_i}$ and $f' \frac{w}{w_i}$. The value of f_i can then be reconstructed in the fragment program by computing $f/f' = \frac{f_i w}{w_i} \frac{w_i}{w}$. The interpolation scheme is illustrated in Figure 5. Alternatively, we could also carry out perspective division in the vertex program to obtain $w = 1$ for all vertices. However, perspective division is not a reversible operation and thus correct clipping cannot be performed if two vertices of a primitive have differently signed w-coordinates. The ray segment length is not a linear function but can be computed from the (linear) positions on the front and back face.

Cell Integration.

A fragment program is responsible for computing the integral of the diffuse smoke color c and transmittance T along the viewing ray segment, according to Eq. 2 and 3. For direct volume rendering applications, the integral is pre-computed for a set of linear density functions and stored as a 3D texture. The densities at the segment borders and the segment length are used to lookup the precomputed color values in the fragment program. Because the transfer function used in our application is linear, two parameters $(\tau_0 L, \tau_1 L)$ are sufficient to determine the weights of the light color (see Eq. 7). We use the inverse of the exponential function to map the parameter range $[0,\infty]^2$ to the texture coordinate range $[0,1]^2$, producing denser sampling for small values, where the weighting functions change more rapidly. In comparison to the approach used in [6], the exponential mapping does not require a bound parameter range and thus the lookup table can be computed once for all datasets at high precision.

First, the fragment program reconstructs the positions and smoke densities at the front and back plane by inverting the perspective division carried out in the vertex program. The integration length is computed from the positions on the front and back plane as $L = \|p_1 - p_0\|$, and the color weights are read from the lookup texture. The fragment program also needs to compute the transmittance for the frame buffer blending described in the next paragraph. The transmittance as defined in Eq. 2 could be computed from the color weights, because $T = 1 - w_0 + w_1$ holds. However, it is more precise to compute the transmittance analytically. We can combine the computation of texture coordinates and transmittance:

$$u = e^{-\tau_0 L/2}, \qquad v = e^{-\tau_1 L/2} \tag{8}$$

$$T = uv, \qquad \{w_0, w_1\} = tex(u^2, v^2) \tag{9}$$

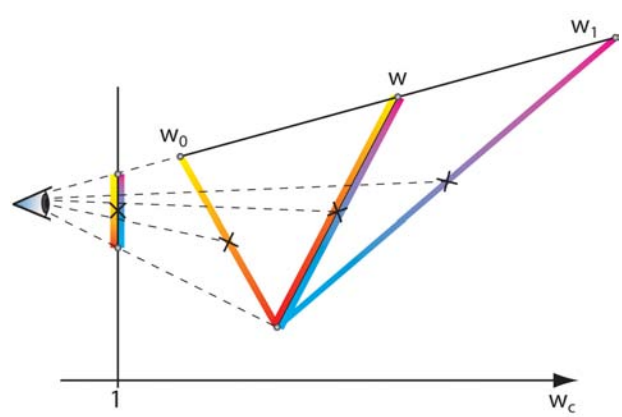

Figure 5: Perspective interpolation of values on the front and back face of a tetrahedron. Each tetrahedron (with one view-aligned edge) is rendered as a triangle (middle line). To achieve correct interpolation of vertex values on the front and back face (left and right line), perspective correction has to be employed, because linear interpolations on the triangle or the image plane do not match the linear interpolations on the front and back faces. This is illustrated by the dashed lines going through the midpoints x.

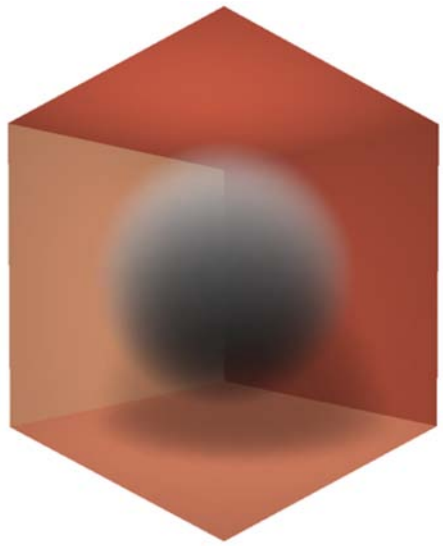

Figure 6: Synthetic sphere of smoke to illustrate the improved appearance with our simple lighting model.

Fragment Blending.

The contribution of each cell is accumulated to the final pixel color using alpha blending. For back-to-front compositing, the framebuffer contains the color of all cells behind the one currently being rendered. The transmittance is the fraction of the color that is absorbed by this cell. The blending equation in OpenGL terminology becomes

$$c_{dst} = c_{src} + \alpha_{src} c_{dst} \tag{10}$$

where the transmittance T is written to the alpha component by the fragment program. Although not generally needed for back-to-front compositing, we also store the accumulated transmittance in the alpha channel of the framebuffer to later adjust the transfer function in front of exit signs as explained in Section 4.4. The separate blending function for the alpha channel is

$$\alpha_{dst} = \alpha_{src} \alpha_{dst} \tag{11}$$

While the fragment program does all computations in 24 or 32bit floating point precision, the color and alpha output is rounded to 8bit fixed point numbers. Alpha dithering is a sufficient way to overcome quantization artifacts, as suggested in [21]. The idea is to round transmittance depending on some random number q:

$$T = (\lfloor 255 \cdot T \rfloor + frac(255 \cdot T) > q ? 1 : 0)/255 \tag{12}$$

The approach described in [6] uses a dependent texture lookup to retrieve the random number. Instead, we use the lower bits of the transmittance to compute a pseudo random number.

$$q = frac(256 \cdot 255 \cdot T) \tag{13}$$

Because quantization also occurs after alpha blending, artifacts can only be avoided completely if a high precision framebuffer is available. The current generation of graphics hardware supports 16bit floating point framebuffers and do not show quantization artifacts. See Figure 7 for a comparison.

Front-to-back compositing uses a different blending function for the color channel ($c_{dst} = \alpha_{dst} c_{src} + c_{dst}$), which produces strong rounding artifacts at 8bit precision: For $\alpha_{dst} c_{src} < 0.5/255$, the blending does not change the value of the destination color. For example, infinite compositing of white smoke cells with 25% opacity leads to a grey value of only 33% instead of 100%.

4.3 Lighting

The lighting of the smoke must be calculated every vertex and time step of the simulated CFD data. We used a simple lighting model which considers the absorption of the light by smoke and opaque surfaces. By restricting the light sources to spotlights, lighting can be performed by the same cell projection procedure as rendering. The scene is rendered from the position of each light source, with a view-frustum bounding the cone of light. The cells are now rendered in front-to-back order. Note that α-compositing is independent of the ordering (in contrast to RGB-compositing, see Eq. 10 and 11), and thus the alpha buffer contains the amount of light transmitted through all cells already rendered. The transmittance value of the alpha buffer can be multiplied with the standard spotlight model to gain the light intensity at the vertex. However, the pixels of the framebuffer are generally not aligned with the vertex positions. To minimize artifacts, we perform bilinear interpolation of the alpha buffer at the projected vertex position. Furthermore, we read back the alpha channel before the cell in front of the vertex is rendered and compute the last factor of the transmittance compositing at the exact vertex position on the CPU. Occlusion is handled by sampling the depth buffer of the opaque walls rendered beforehand and comparing it with the vertex depth. The accumulation of all light sources in the scene equals to the light intensity that reaches a mesh vertex.

All lighting calculations are done only once for every time step in a preprocessing step and the results are saved to disk for later use during visualization.

Because the computational grid of the smoke propagation is generally aligned with the building geometry, we can use boundary faces of the grid to render shadows on walls of the building. First, all opaque geometry is rendered using standard OpenGL lighting. Then, semi-transparent boundary faces are rendered on top to darken the walls according to absorption from smoke, occlusion and attenuation of the spotlight model. The depth range is shifted slightly to avoid depth buffer collision. The result of the lighting is visualized for a synthetic ball of smoke in Figure 6.

4.4 Rendering of exit signs

An important part of our application is the visibility of exit signs under various smoke conditions. Compared to reflecting signs, self-emitting signs have a larger visibility distance, which is modeled by a reduced smoke potential of $(K_m/8)y_s$ instead of $(K_m/3)y_s$, see [11]. We can exploit the special relationship of the two transfer functions to draw illuminated exit signs after the smoke has been rendered. According to Eq. 2, the transmittance of the reduced smoke potentional is

$$T' = T^{3/8} \tag{14}$$

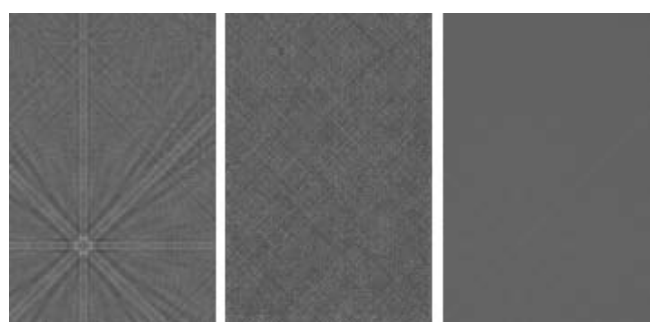

Figure 7: Precision artifacts for a block of uniform smoke density. Standard 8bit framebuffer blending (left), 8bit framebuffer with alpha dithering (middle), 16bit framebuffer (right).

As described in Section 4.2, the transmittance T of the smoke is rendered to the alpha component of the frame buffer. Illuminated exit signs can then be rendered in an additional pass using front-to-back compositing:

$$c = T' c_e + c_s \tag{15}$$

where c_s is the smoke color and c_e is the color of the exit sign. To access the per-pixel value of uncorrected transmittance T in the fragment program, a bounding box per exit sign is copied from the frame buffer to a texture. Texture coordinate interpolation needs special attention when redrawing the exit signs: while the decal texture of the exit sign uses perspective texture mapping, the α-map is parallel to the image plane and uses linear texture mapping. Figure 8 shows the effect of adjusting the transfer function.

5 Results

We had access to two datasets of different size to measure the rendering performance of our algorithm. The smaller dataset consists of two connected rooms with no ventilation. The seat of fire is placed in the middle of the smaller room. During the animation, a dense column of smoke forms above the source, accumulates below the ceiling and eventually fills out the whole room as shown in Figure 1. The larger dataset (shown in Figure 9 models a shopping mall with seven stories and ventilation. The simulated source of fire is placed on the first floor and during the animation, the smoke quickly spreads in the building. The details of the two datasets are listed in Table 2 and the frame rates achieved by our approach are listed in Table 3.

6 Discussion

Simulations of smoke propagation inside buildings has been previously analyzed using iso-surfaces and contour plot slices, but not with volume visualization techniques. Visibility plays a major role for safe evacuation and safe operation of the fire rescue staff during a fire. With previous methods, only a crude approximation of the visibility is visualized, corresponding to a local visibility distance assuming uniform smoke concentration. Our volume rendering approach allows analysis of accurate visibility for any point of view described by a standardized optical model.

Dataset	vertices	cells	triangles
Hall	26'392	120'925	12'458
Mall	345'536	1'724'811	219'584

Table 2: Geometric complexity of two test datasets. The first two columns state the number of vertices and number of tetrahedrons of the volumetric mesh. The last column states the number of triangles of the building geometry.

Smoke	steady	animated	steady	animated
Camera	fixed	fixed	animated	animated
Hall	4.6 / 2.8*	4.6 / 2.7*	2.4 / 1.9*	1.9 / 1.5*
Mall	0.61	0.37	0.42	0.25

Table 3: Frame rates (in s^{-1}) of two datasets at different levels of interactivity. A camera that is zoomed and rotated is still considered fixed because tetrahedra sorting and projection does not need to be performed. Frame rates denoted with * contain a back-illuminated exit sign, which requires an extra transfer function adjustment path. The frame rates are averages of the configurations shown in the screen shots in Figures 1 and 9.

Figure 8: Illuminated exit signs (left column) are visible further than reflective exit signs (right column), which is achieved by adjustment of the transfer function.

Realistic simulation of disorientation in a major fire is an important aspect of evacuation planning. By adapting common volume rendering techniques, time dependent smoke data can be lit with a simple absorption and scattering-based lighting model and rendered in real-time. Adapting the transfer function in front of exit signs accounts for the increased distance of visibility for illuminated objects.

7 Conclusion

We have shown that direct volume rendering can successfully be applied to the application of technical smoke visualization. Although photo realism was not the aim of the approach, the visual quality of our renderer is superior to previously used iso-surface rendering and allows realistic visualization of the view of a person inside a building filled with smoke. In comparison to more general volume rendering approaches, the pre-defined linear mapping from smoke density to opacity allows pre-integration with a 2D table lookup. Employing perspective interpolation and 16bit framebuffers, no visible artifacts are present in the renderings. Our simple light model can be evaluated using the same rendering approach and significantly improves appearance of the smoke. The transfer function in front of illuminated exit signs can be adjusted in a final rendering pass of image space complexity.

References

[1] Dougal Drysdale. *An Introduction into Fire Dynamics*. J. Wiley Sons, Chichester (UK), 1998.

[2] Klaus Engel, Martin Kraus, and Thomas Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *HWWS '01: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 9–16, New York, NY, USA, 2001. ACM Press.

[3] Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22, 2001.

[4] Mark J. Harris and Anselmo Lastra. Real-time cloud rendering. In *Computer Graphics Forum*, volume 20, pages 76–84, 2001.

[5] Geir Jensen. *Evacuating in Smoke - Decisive Factors*. IGP AS, Trondheim, Norway, 1994.

[6] Martin Kraus, Wei Qiao, and David S. Ebert. Projecting tetrahedra without rendering artifacts. In *Proceedings of IEEE Volume Visualization Symposium '04*, pages 27–34, 2004.

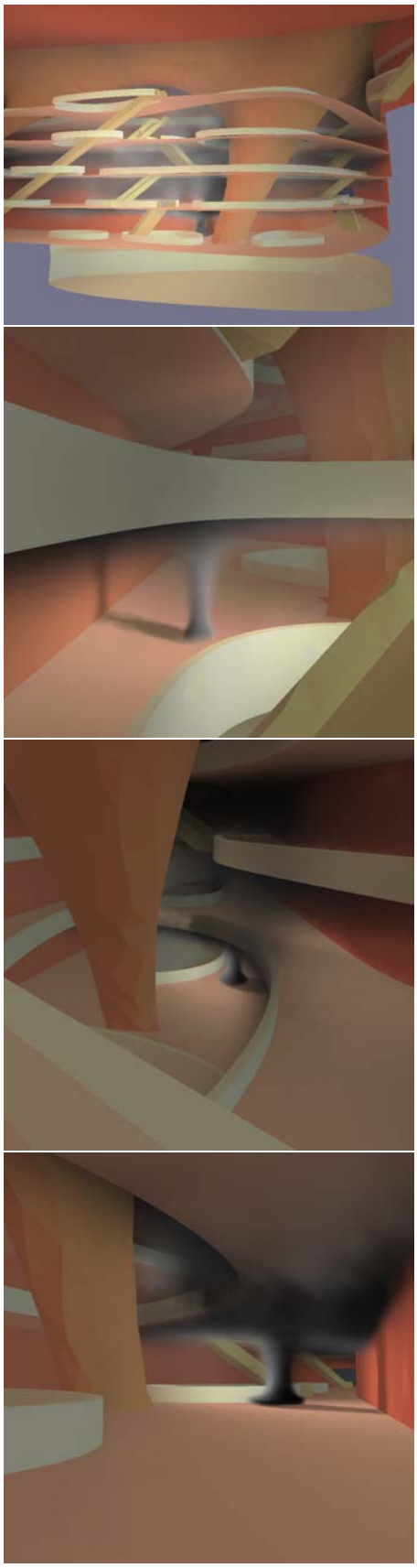

Figure 9: Four different viewports and time steps of smoke propagating inside a mall with seven stories. The average performance is stated in Table 3.

[7] Jens Krueger and Ruediger Westermann. Acceleration techniques for gpu-based volume rendering. In *Proceedings IEEE Visualization 2003*, 2003.

[8] Nelson Max. Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[9] Nelson Max, Barry Becker, and Roger Crawfis. Flow volumes for interactive vector field visualization. In *VIS '93: Proceedings of the 4th conference on Visualization '93*, pages 19–24, 1993.

[10] Kenneth Moreland. *Fast High Accuracy Volume Rendering*. PhD thesis, University of New Mexico, July 2004.

[11] George W. Mulholland. *Smoke production and properties, SFPE Handbook of Fire Protection Engineering*. Society for Fire Protection Engineering, Boston, 2nd edition, 1995.

[12] D.J. Rasbash and R.P. Phillips. *Quantification of smoke produced at fires*, volume 2(3). 1978.

[13] Stefan Roettger, Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Proceedings of IEEE Visualization 2000*, pages 109–116, 2000.

[14] P. Rosemann and A. Moser. Smoke movement in buildings. In *Indoor Air 99, 8th International Conference on Indoor Air Quality Climate, Edinburgh, Scotland*, August 1999.

[15] Jens Schneider and Ruediger Westermann. Compression domain volume rendering. In *Proceedings IEEE Visualization 2003*, 2003.

[16] Joshua Schpok, Joseph Simons, David S. Ebert, and Charles Hansen. A real-time cloud modeling, rendering, and animation system. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 160–166, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[17] Peter Shirley and Allan Tuchman. A polygonal approximation to direct scalar volume rendering. In *VolViz '90: Proceedings of the 1990 workshop on Volume visualization*, pages 63–70, 1990.

[18] Cludio T. Silva, Joseph S. B. Mitchell, and Peter L. Williams. An exact interactive time visibility ordering algorithm for polyhedral cell complexes. In *IEEE Symposium on Volume Visualization*, pages 87–94. IEEE, ACM Siggraph, 1998.

[19] Manfred Weiler, Martin Kraus, and Thomas Ertl. Hardware-based view-independent cell projection. In *VolViz '02: Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*, pages 13–22, 2002.

[20] Manfred Weiler, Martin Kraus, Markus Merz, and Thomas Ertl. Hardware-based ray casting for tetrahedral meshes. In *Proceedings of IEEE Visualization '03*, pages 333–340, 2003.

[21] Peter L. Williams, Randall J. Frank, and Eric C. LaMar. Alpha dithering. Technical Report UCRL-ID-153185, Lawrence Livermore National Laboratory, March 2003.

[22] Brian Wylie, Kenneth Moreland, Lee Ann Fisk, and Patricia Crossno. Tetrahedral projection using vertex shaders. In *VolViz '02: Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*, pages 7–12, 2002.