Fast Volumetric Display of Natural Gaseous Phenomena

Stefan Roettger and Thomas Ertl

Visualization and Interactive Systems Group
University of Stuttgart

Abstract

Mesh simplification algorithms play an important role in computer graphics. In particular, view-dependent simplification methods are utilized widely to reduce the geometric complexity of large outdoor scenes. The so called continuous level of detail technique, for instance, is prevalent in the area of terrain rendering. In this paper we apply the latter technique to the more general volumetric case. We propose a volume rendering algorithm, which constructs a hierarchical tetrahedral mesh from regular volume data in a view-dependent fashion. The popping effect which generally arises from view-dependent algorithms is addressed by a new fast method for the smooth interpolation of the mesh hierarchy. We demonstrate the performance of our algorithm by displaying clouds in real-time. Here, the application of the pre-integration technique allows for a wide range of cloud appearance and guarantees accurate compositing. Additionally, we show how to utilize our algorithm for the accelerated display of ground fog in terrain rendering scenarios.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling, I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism.

Keywords: Volume Rendering, Continuous Level of Detail, Cell Projection, Pre-Integration, Gaseous Phenonema.

1 Introduction and Related Work

In general, the strategy to simplify a mesh in a view-dependent fashion is suited well for the real-time display of large scenes. This has been exemplified in the area of terrain rendering, where the continuous level of detail technique (C-LOD) [25, 9, 34] is well established. This technique achieves high frame rates by generating an approximate view-dependent triangulation of the terrain. In order to minimize the total screen space error of the approximation, small distant details are represented with fewer triangles than those which are nearby.

Despite the widespread use and the maturity of the C-LOD technique it has not been applied to the more general case of volume rendering [8] yet: Multi-resolution analysis for the display of polygonal meshes has been introduced by Rossignac et al. [36], and has been the subject of intense studies later on (see Xia et al. [49] as a starting point). General multi-resolution analysis of volumetric meshes has been given by Eck et al. [12] and more recently by Cignoni et al. [4]. Variants for the hierarchical visualization of regular volume data have been discussed by Laur et al. [23], Zhou et al. [50], and Schussman et al. [38]. A view-dependent simplification method for irregular grids has been proposed by Meredith et al. [29]. But still the efficient view-dependent simplification of regular volume data is an active research field.

In this paper we present a general purpose volume rendering algorithm which is based on the continuous level of detail idea. It maintains an octree to construct a view-dependent representation of regular volume data. After decomposing each leave node of the octree into tetrahedra these can be rendered efficiently by using the projected tetrahedra (PT) algorithm of Shirley and Tuchman [40].

A common property of view-dependent algorithms is the occurrence of the so called popping artifacts: Small distant details will suddenly pop up when approaching nearby. In the case of a C-LOD terrain renderer the total screen space error of the approximation can be pushed easily below the one pixel boundary, so that the popping effect becomes invisible. In the volumetric case, however, this approach is infeasible. As a solution to this problem, the mesh hierarchy has to be interpolated smoothly. In consideration of this fact, we present a new fast mesh interpolation method, which we refer to as volumetric morphing throughout the paper.

One of the classic volume rendering scenarios is the medical visualization of computer tomography data [44]. While this scenario is suitable for hardware-accelerated multi-resolution techniques [22, 43, 3], it has little potential with respect to view-dependent rendering. Thus, we demonstrate our octree based simplification algorithm by displaying gaseous phenomena in real-time. In particular we give an application example in the area of non-physically based weather visualization and we show how to enrich outdoor scenes with volumetric ground fog.

2 Generating Continuous Levels of Detail

In this section we describe how to adapt the C-LOD technique previously known from terrain rendering [25, 9, 34] to the volumetric case.

2.1 Hierarchical Volume Representation

Given a three-dimensional scalar field, which is defined by an array with 2^n+1 ($n>0$) grid points in each dimension, a hierarchical volumetric mesh is constructed by building an octree in a bottom-up fashion. Grids with a size other than 2^n+1 have to be padded or resampled. Each leave node of the octree

is decomposed into five tetrahedra. Since there exist two topologically different decompositions, adjacent nodes of the same level of detail have to be decomposed in an alternating fashion to ensure a conforming mesh. In Figure 1 the orientation of the tetrahedra is depicted for a coarse example hierarchy.

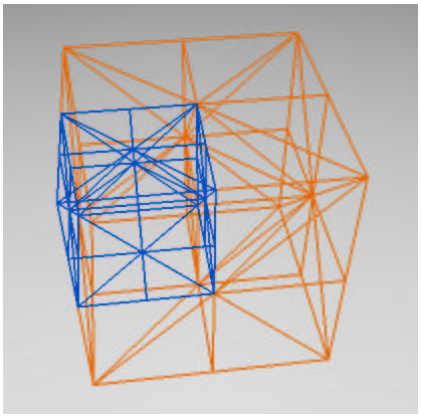

Figure 1: Hierarchical volume representation using an octree: The example hierarchy consists of the root node with 8 children (bright/orange), one of which has been refined into another 8 children (dark/blue). Each leave node of the octree has been decomposed into five tetrahedra in an alternating fashion.

2.2 View-Dependent Mesh Simplification

The key idea of a volumetric C-LOD algorithm can be described as follows: In order to perform a view-dependent simplification the octree has to be updated for each frame. During a top-down traversal of the octree our approach calculates an upper limit on the local screen space error of each node. If the local error exceeds a predefined threshold the corresponding node is split into eight children.

The error metric used to estimate the local screen space error is designed to meet the following criteria: A node should be refined if the local simplification error is large. Also, small distant nodes should be refined less likely than those which are nearby. Let s be the edge length of each node, let d denote the euclidean distance of the eye to the center of the node, and let Δ be the local simplification error of the node in object space. With the previous definitions, we introduce the error metric e as follows:

$$e = \frac{sC\max(c\Delta, 1)}{d} \tag{1}$$

If the error metric e is greater than one, the node is refined, else the refinement of the octree is stopped. The global accuracy of the mesh can be controlled via the user definable constant c. Higher values result in a finer mesh. Additionally, the constant C defines a lower bound on the global accuracy.

In a preprocessing step the local error Δ is computed. It is defined to be the average of the scalar deviations Δ_i at the center of the node and the midpoints of the edges and faces. The scalar deviations Δ_i are equal to the difference of the scalar value of each vertex and the interpolated scalar value derived from the next coarser level of detail. For instance, the deviation of the midpoint of an edge is equal to the absolute scalar difference of that vertex and the average of the two adjacent corner vertices (also compare Figure 2 where $\Delta_{midleft} = |\frac{1}{2}(S_{topleft} + S_{bottomleft}) - S_{midleft}|$).

2.3 Building a Conforming Mesh

For adjacent nodes, which do not belong to the same level of detail (depicted by the orange and blue colors in Figure 1), the interpolated scalar values at a T-vertex of the boundary face do not match. One solution to ensure a conforming mesh is to insert irregular tetrahedra into the coarser node. This technique is known as the red-green or regular-irregular refinement method [1, 17]. But if we want to morph between two of the large number of irregular configurations, the situation is getting inscrutable complex. Furthermore, adjacent nodes must not differ by more than one level for this method to work. In order to circumvent these problems, we employ a different approach: Rather than inserting irregular tetrahedra into the coarser node, we manipulate the scalar values of the refined node. To build a conforming mesh the scalar value at a T-vertex is simply substituted by the interpolated value from the coarser mesh of the adjacent neighbour node. A detailed example is given in Section 3.

2.4 Hierarchical Error Propagation

Since we use a top-down simplification approach, at each node only the local simplification error is known. However, in order to minimize the total screen space error of the generated mesh, we also need to know the local simplification error of all children in advance. This can be accomplished by propagating the local error from the children up to the parents of the octree. In principle, the error propagation has to ensure that a node is refined, if at least one child already fulfills the refinement condition. In mathematical terms this can be written as:

$$e_{child} > 1 \rightarrow e > 1 \quad \text{or} \quad e > e_{child} \tag{2}$$

Substituting Equation 1 into Equation 2 yields

$$\Delta > K\Delta_{child} \quad \text{with} \quad K = \frac{d}{2d_{child}}. \tag{3}$$

Now we determine an upper bound for K. Since we introduced a minimum accuracy C which always guarantees refinement for $d < sC$ we just have to consider the case $d \geq sC$. On the one hand, the minimum possible value of K is $\frac{1}{2}$ for an infinite distant viewer. On the other hand, the maximum possible value of K occurs for the minimum distance $d = sC$. Then the minimum distance to the center of one of its children is $d_{child} = sC - \frac{1}{4}\sqrt{3}s$. Resubstituting these distances into Equation 3 yields the following upper bound for K:

$$K_{max} = \frac{C}{2C - \frac{1}{2}\sqrt{3}} \quad (C > \sqrt{3}) \tag{4}$$

As a consequence, Formula 5 can be used to propagate the local error Δ from all eight children up to the parent nodes.

Starting with the leave nodes, all nodes which belong to the same level of detail are processed in a row. For each node the final propagated Δ-values are stored at the center vertex of each node using a linear mapping with 16 bits of accuracy.

$$\Delta := \max(\Delta, K_{max} \cdot \Delta_{child}) \tag{5}$$

In summary, the update of the view-dependent hierarchy is performed by refining the octree, if and only if Equation 1 is fulfilled. The simplicity of this approach is the basis for the real-time performance of our algorithm. Another important advantage is that volumetric morphing can be implemented very efficiently as shown in Section 3.

3 Volumetric Morphing

In this section we describe a new fast method to morph the view-dependent hierarchy. Volumetric morphing is mandatory, because otherwise the transition from one level of detail to another could be observed easily.

For each frame, first the hierarchy is updated using the view-dependent approach described in Section 2. During the update the error metric e is mapped to the range $[0,1]$ according to Equation 6 and stored at the center vertex of each node with 8 bits of accuracy.

$$e' = \min(\max(e-1,0),1) \tag{6}$$

In a second octree traversal, the normalized error metric e' is interpreted in the following way: A value of zero ($e \leq 1$) means that the corresponding node has not yet been refined, thus it can be decomposed into 5 tetrahedra and rendered as described in Section 4. A value greater than zero and less than one ($e \in (1,2)$) means that the node has been refined but still none of its children. A value of one ($e \geq 2$) means that the node and at least one of its children have been refined. As a consequence, the time between the two subsequent refinement events for $e' = 0$ and $e' = 1$ can be used to blend the scalar values of the corresponding node as smooth as possible. Thus, the parameter e' just serves as an interpolation factor to morph recursively between the actual node and its children.

In contrast to a fixed blending time interval [20], the speed of the interpolation is coupled to the error metric. This is a much better strategy for volumetric morphing, since distant details can be morphed much slower than those which are nearby. In practice, we have found that the maximum instead of the average of the deviations Δ_i suppresses the popping effect more reliably. This is due to the fact that the subjective observability of the interpolation is determined by the maximum and not by the average change of all vertices.

In the context of the described interpolation scheme a conforming mesh can be guaranteed simply by using the minimum interpolation factor of all adjacent nodes which share the interpolated vertex. If one of the adjacent nodes has not been refined, the corresponding interpolation factor is assumed to be zero.

In the following we illustrate the described interpolation scheme using a two-dimensional example, which is depicted in Figure 2. In general, only the scalar values of the non-corner vertices of a node have to be interpolated using the normalized error metric e' as the interpolation factor. In the two-dimensional case, the non-corner vertices of a node are the midpoints of the four edges (black dots) and the center vertex (white dot). For each of those vertices the interpolation is performed between the average scalar value of the two adjacent corner vertices (small black crosses) and the actual scalar value of the vertex. For the midpoint of the left edge of the grey node in Figure 2, for example, the interpolated scalar value $S'_{midleft}$ is calculated as follows:

$$w = \min(e', e'_n) \tag{7}$$

$$S'_{midleft} = w\frac{1}{2}(S_{topleft} + S_{bottomleft}) + (1-w)S_{midleft} \tag{8}$$

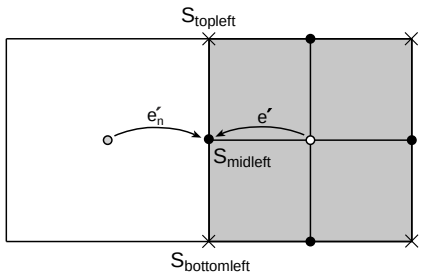

Figure 2: Two-dimensional morphing example.

In the three-dimensional case, the non-corner vertices of a node are the centroid, the midpoints of the six faces, and the midpoints of the eight edges. The scalar values of these vertices are interpolated in analogy to the two-dimensional example. The midpoints of the edges are shared among the actual node and a maximum of three adjacent nodes of the same level of detail. Thus, the minimum interpolation factor of these vertices has to be calculated from the interpolation factors of the actual and the three adjacent nodes. The midpoints of the faces are shared among two nodes. Here, additional care must be given to the calculation of the average scalar value of the corner vertices, since adjacent nodes are decomposed in an alternating fashion. The average scalar value is therefore computed from the appropriate two corner vertices of the tetrahedral decomposition of the adjacent neighbour node. The centroid of a node is located inside the center tetrahedron of the decomposition. In this case the average scalar value is computed from the four vertices of the center tetrahedron. Again, special care must be given to the calculation of the average scalar value, since the orientation of the center tetrahedron alters.

4 Cell Projection

Now that we have performed a view-dependent simplification of a regular volume, the generated tetrahedra have to be composed in a back to front fashion. We apply the cell-projection technique, that is the PT algorithm of Shirley and Tuchman [40, 42, 45, 47]. The original PT algorithm only supports linear transfer functions which are not appropriate

for the display of gaseous phenomena as demonstrated in Section 5. Therefore we also apply the pre-integration method of Roettger et al. [35, 26, 14] which allows the use of arbitrary transfer functions by storing the ray integral in a three-dimensional pre-integration table. Visibility sorting [48, 5] is performed by reordering the traversal of the octree in a back to front fashion.

4.1 Zero Opacity Test

In order to speed up the PT algorithm we discard transparent tetrahedra by applying the so called *Zero Opacity Test* (ZOT). While this test is obvious for linear transfer functions, it is not as obvious for arbitrary transfer functions. Fortunately, the three-dimensional pre-integration table contains all necessary information to apply this test. First, the minimum and maximum scalar values (denoted by S_{min} and S_{max}) of the tested tetrahedra are computed. If the entry at position $(\lfloor (n-1)S_{min} \rfloor, \lceil (n-1)S_{max} \rceil, m-1)$ of the pre-integration table is zero (using the notation of [35], n is the resolution along the S_f and S_b coordinates and m is the resolution along the l coordinate), then we can discard the tested tetrahedra. By applying the ZOT to each visited node of the octree, we can discard all transparent tetrahedra with virtually no computational overhead.

4.2 Hierarchical View Frustum Culling

Another common way to speed up rendering is view frustum culling. During the rendering traversal of the octree each node is tested against intersection with the view frustum. If a node does not overlap with the view frustum, it is invisible and can be discarded.

4.3 Volumetric Clipping

Since we want to allow the viewer to navigate freely inside the volume, we face the following problem: If a tetrahedron intersects the near clipping plane, the clipped two-dimensional projection is not identical with the clipped volume of the tetrahedron. In order to display the tetrahedron correctly, it has to be be clipped in a truly volumetric fashion. We distinguish two different cases: Either the tetrahedron is cut into one tetrahedron and one prism or it is cut into two prisms. Therefore, the visible part of the clipped tetrahedron is either a tetrahedron or a prism. In the latter case the total number of rendered tetrahedra is increased, since the prism has to be decomposed into three tetrahedra. In comparison to the total number of rendered tetrahedra, the number of clipped tetrahedra can be considered to be fairly low. In our test scenes it turned out that the fraction of additional tetrahedra was well below 10%.

5 Non-Physically Based Display of Clouds

In the previous sections we have described a general purpose volume rendering algorithm which is based on a view-dependent simplification. In this section we demonstrate the abilities of this approach by rendering gaseous phenomena, in particular clouds and ground fog.

In general, we can think of a cloud as a three-dimensional scalar function $f = f(x,y,z)$. The scalar values correspond to the optical density of the medium. Due to the complex anisotropic light scattering [2, 16, 11, 27, 41, 32, 21] inside a cloud the photorealistic display is a time consuming task. Impostors [37, 39] are currently the dominating technique here [7, 13, 18].

But if we restrict ourselves to isotropic light scattering the cloud intensities can be precomputed and we can apply the described view-dependent simplification algorithm. As a result, the clouds are modeled by two scalar fields, the scalar density $f(x,y,z)$ and the scalar isotropic light intensity $\gamma(x,y,z)$. The mesh simplification is driven by the maximum deviation of both scalar fields.

This approach has the following advantages: Since we use a truly volumetric representation there are no restrictions with respect to cloud shape and appearance. As opposed to the impostor method, the clouds are displayed without temporal aliasing or perspective artifacts, even for view points inside the clouds. This is guaranteed by the application of volumetric morphing and clipping.

5.1 Modified PT Algorithm

In principal, the volume density optical model [46] used for pre-integrated volume rendering presumes the transfer functions κ (the chromaticity vector) and ρ (the scalar optical density) to depend both on the scalar density function $f(x,y,z)$. But, since we want the optical density to depend on the density function $f(x,y,z)$ and the chromaticity vector to depend on the precomputed light intensities $\gamma(x,y,z)$, we circumvent this restriction of the optical model by slightly modifying the PT algorithm. For this purpose, we assume that $\rho = Id$. Then we apply the pre-integration to the chromaticity vector $\kappa = \kappa(\gamma(x,y,z))$ and the maximum optical density $\rho_{max} = f_{max}$. To introduce the dependency on $f(x,y,z)$ we modulate the effective length l of each tetrahedral ray segment by the scalar optical density $\rho = f(x,y,z))$ according to the following equation:

$$l' = l\frac{f(x,y,z)}{f_{max}} \tag{9}$$

5.2 Non-Physically Based Lighting

The previous approach requires a light scattering simulation [27, 32, 21] to calculate the light intensity function $\gamma(x,y,z)$. Instead of changing physical simulation parameters we propose a non-physical approach which achieves the desired look and feel of the clouds by a direct manipulation of the transfer functions.

For this purpose, the chromaticity vector $\kappa = \kappa(f(x,y,z))$ is defined to be an inverse color ramp and the optical density $\rho = \rho(f(x,y,z))$ is defined to be a linear function except for very small densities where it is set to zero. This allows to speed up rasterization by discarding nearly transparent areas with the ZOT. The light intensities γ are calculated by standard ambient and diffuse lighting and are used to modulate the effective ray segment length as described before. With respect

to the direction of the diffuse light this leads to high opacities at the front and to low opacities at the back of the clouds. As a consequence, the dark inside of each cloud shines through the translucent back, but at the front bright colors still dominate the appearance of the clouds. This approach effectively mimics the natural look and feel of clouds without requiring a physical lighting simulation.

6 Real-Time Display of Ground Fog

In this section we describe a variant of the view-dependent simplification technique which is specifically suitable for the real-time display of ground fog.

Each terrain renderer that is based on the C-LOD technique creates a view-dependent triangulation of a height field. In order to display volumetric ground fog, we introduce a second height field (the ground fog map) which defines the height of the fog layer above the ground. Each triangle that is generated by the C-LOD algorithm is treated as a base triangle onto which a vertically aligned prism is stacked. The heights of the three vertical edges of each prism are derived from the ground fog map. To account for the additional map, the simplification of the triangle mesh is driven by the deviations of both the height field and the ground fog map. In order to suppress the popping effect the upper boundary of the ground fog layer is geomorphed in the same fashion as the surface of the terrain.

Each prism is decomposed into three tetrahedra in an alternating fashion to ensure a conforming mesh. Prisms with zero height can be discarded to speed up rasterization. The generated tetrahedra are rendered using the methods described in Section 4 and 5. At the upper boundary of the fog layer the optical density is set to zero which leads to a more realistic appearance than using a constant fog density.

In the case that a specific C-LOD algorithm does not allow straight-forward back to front sorting of the tetrahedra, we propose an emissive optical model [28] with exponential saturation, which allows to omit the sorting stage: With the average optical density τ and the length l of a ray segment the exponentially saturated emission $\varepsilon = 1 - e^{\tau l}$ is stored in a 2D texture similar to the approach of Stein et al [42]. In our case, however, the blended intensity I' of a pixel is calculated from the previous intensity I in the frame buffer using the following blend function:

$$I' \quad := \quad \varepsilon + (1-\varepsilon) \times I \tag{10}$$

7 Results

Using the described non-physically based cloud and ground fog rendering algorithms, Figures 3 and 4 show the city center of Stuttgart with some cumulus clouds and ground fog in the valleys. The scene was rendered in real-time with approximately 26 frames per second on a PC equipped with a 1.2 GHz AMD Athlon and a NVIDIA GeForce3 graphics adaptor. About 25% of the total rendering time was spent on terrain rendering [34], 20% was spent for the display of the ground fog and the remainder of 55% for the display of the clouds. The latter were generated with 3D Perlin noise [33, 10], whereas the ground fog map was painted by hand. The applied transfer functions κ and ρ are depicted on the left side of Figure 3.

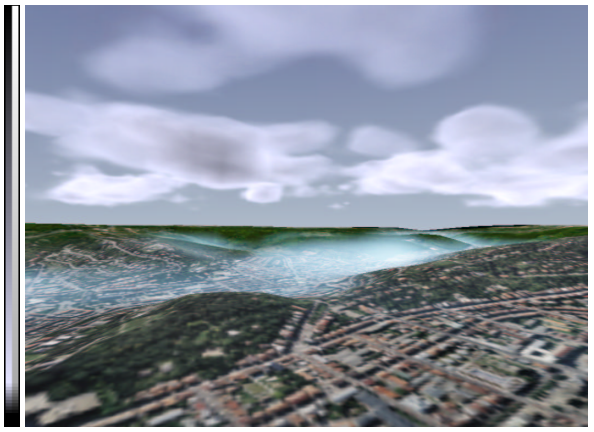

Figure 3: The city center of Stuttgart with clouds and ground fog in the valleys. The applied transfer functions κ and ρ are depicted on the left.

The size of the height field and the ground fog map is 2049×2049, whereas the cumulus clouds are represented by an 8 bit density field with a base size of 513×513 and a height of 65 grid points. For the density field one byte is consumed per grid point plus 16 bits for the deviations Δ and one byte for the interpolation parameter e' summing up to a total of 48 MB in our example.

The size of the pre-integrated 3D texture is $64 \times 64 \times 128$ which corresponds to 2 MB of graphics memory. Since only the 3D texture has to be kept in graphics memory, the maximum cloud size is limited by main memory only.

For the display of the clouds the number of rendered tetrahedra was reduced from a total of 83 million to less than 10 thousand tetrahedra on the average. This corresponds to four orders of magnitude reduction.

An analysis of the experimental results reveals two bottlenecks. The main bottleneck is the projection of the tetrahedra. This is due to the fact that for each single node of the octree 5 tetrahedra have to be decomposed into an average number of 17.5 triangles. If the view point is entirely inside a cloud, the algorithm is mostly fill-rate bound and the performance drops to approximately 15 frames per second for a window size of 512×384 pixels.

8 Discussion

In comparison to the impostor technique, our approach offers the following advantages: Most important, a flight through the clouds does not introduce temporal aliasing or perspective artifacts, since we use volumetric morphing and clipping. Furthermore, our general purpose volume rendering algorithm is able to render arbitrary weather conditions including overcast sky and storm clouds. Besides the shown 3D Perlin noise example more sophisticated cloud simulation algorithms [31, 30, 15]

are compatible with our approach, which makes the algorithm well suited for the purpose of weather visualization.

The use of a non-physically based rendering model somewhat limits the area of application, but in many cases the real-time performance outweighs this restriction such as in interactive entertainment. Here the layered fog technique [24, 19] is used commonly (i.e. in the DX8 game AquaNox [6]), but has the disadvantage that the vertical fog boundaries are fixed. With the described ground fog rendering algorithm we overcome this restriction by explicitely defining the height of the fog layer.

9 Conclusion

We have developed a general purpose volume rendering algorithm. It is based on the continuous level of detail technique, which approximates a three-dimensional scalar field in a view-dependent fashion. The necessity to suppress the popping effect has been addressed by a new fast algorithm for volumetric morphing. We have demonstrated the performance of our algorithm by displaying clouds and ground fog in real-time. Because of the truly volumetric representation of the clouds, the algorithm is suited for real-time weather visualization and the display of fast volumetric effects in games or VR environments.

References

[1] R. Bank, A. Sherman, and A. Weiser. Refinement Algorithm and Data Structures for Regular Local Mesh Refinement. *Scientific Computing*, 44:3–17, 1983.

[2] Jim. F. Blinn. Light Reflection Functions for Simulation of Clouds and Dusty Surfaces. *Computer Graphics*, 16(3):21–29, 1982.

[3] I. Boada, I. Navazo, and R. Scopigno. Multiresolution Volume Visualization with a Texture-Based Octree. *The Visual Computer*, pages 185–197, 2001.

[4] P. Cignoni, C. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of Tetrahedral Meshes with Accurate Error Evaluation. In *Proc. Visualization '00*, pages 85–92. IEEE, 2000.

[5] J. Comba, J. T. Klosowski, N. L. Max, J. S. B. Mitchell, C. T. Silva, and P. L. Williams. Fast Polyhedral Cell Sorting for Interactive Rendering of Unstructured Grids. *Computer Graphics Forum (Proc. Eurographics '99)*, 18(3):369–376, 1999.

[6] Massive Development. AquaNox Home Page, `www.aquanox.de`, 2001.

[7] Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, and T. Nishita. A Simple, Efficient Method for Realistic Animation of Clouds. In *Proc. SIGGRAPH '00*, pages 19–28. ACM, 2000.

[8] R. A. Drebin, L. Carpenter, and P. Hanrahan. Volume Rendering. *Computer Graphics*, 22(4):65–74, 1988.

[9] M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, Ch. Aldrich, and M. B. Mineev-Weinstein. ROAMing Terrain: Real-Time Optimally Adapting Meshes. In *Proc. Visualization '97*, pages 81–88. IEEE, 1997.

[10] D. Ebert, K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & Modeling, A Procedural Approach*. AP Professional, second edition, isbn 0-12-228730-4 edition, 1998.

[11] D. Ebert and R. Parent. Rendering and Animation of Gaseous Phenomena by Combining Fast Volume and Scanline A-Buffer Techniques. *Computer Graphics (Proc. SIGGRAPH '90)*, 24(4):357–366, 1990.

[12] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution Analysis of Arbitrary Meshes. In *Proc. of SIGGRAPH '95*, pages 173–182. ACM, 1995.

[13] P. Elinas and W. Stuerzlinger. Real-time Rendering of 3D Clouds. *Journal of Graphics Tools*, 5(4):33–45, 2000.

[14] K. Engel, M. Kraus, and Th. Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *Eurographics Workshop on Graphics Hardware '01*, pages 9–16. ACM SIGGRAPH, 2001.

[15] R. Fedkiw, J. Stam, and H. W. Jensen. Visual Simulation of Smoke. In *Proc. SIGGRAPH '01*, pages 15–22. ACM, 2001.

[16] Geoffrey Y. Gardner. Visual Simulation of Clouds. In *Proc. SIGGRAPH '85*, pages 297–303. ACM, 1985.

[17] R. Grosso, Ch. Lürig, and Th. Ertl. The Multilevel Finite Element Method for Adaptive Mesh Optimization and Visualization of Volume Data. In *Proc. Visualization '97*, pages 387–394. IEEE, 1997.

[18] M. J. Harris and A. Lastra. Real-Time Cloud Rendering. *Computer Graphics Forum (Proc. Eurographics '01)*, 20(3):76–84, 2001.

[19] W. Heidrich, R. Westermann, H.-P. Seidel, and Th. Ertl. Applications of Pixel Textures in Visualization and Realistic Image Synthesis. In *Proc. ACM Symposium on Interactive 3D Graphics*, pages 127–134, 1999.

[20] Hugues Hoppe. Smooth View-Dependant Level-of-Detail Control and its Application to Terrain Rendering. In *Proc. Visualization '98*, pages 35–42. IEEE, 1998.

[21] H. W. Jensen and P. H. Christensen. Efficient Simulation of Light Transport in Scenes with Participating Media Using Photon Maps. In *Proc. SIGGRAPH '98*, pages 311–320. ACM, 1998.

[22] E. C. LaMar, B. Hamann, and K. I. Joy. Multiresolution Techniques for Interactive Texture-Based Volume Visualization. In *Proc. Visualization '99*, pages 355–362. IEEE, 1999.

[23] D. Laur and P. Hanrahan. Hierarchical Splatting: A Progressive Refinement Algorithm for Volume Rendering. In *Proc. SIGGRAPH '91*, pages 285–288, 1991.

[24] Justin Legakis. Fast Multi-Layer Fog. In *ACM SIGGRAPH '98 Conference Abstracts and Applications*, page 266, 1998.

[25] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust, and G. Turner. Real-Time, Continuous Level of Detail Rendering of Height Fields. In *Proc. SIGGRAPH '96*, pages 109–118. ACM, 1996.

[26] N. L. Max, P. Hanrahan, and R. Crawfis. Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions. *Computer Graphics (San Diego Workshop on Volume Visualization)*, 24(5):27–33, 1990.

[27] Nelson L. Max. Efficient Light Propagation for Multiple Anisotropic Volume Scattering. In *5th Workshop on Rendering*, pages 87–104. Eurographics, 1994.

[28] Nelson L. Max. Optical Models for Direct Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):99–108, 1995.

[29] J. Meredith and K. Ma. Multi-Resolution View-Dependent Splat Based Volume Rendering of Large Irregular Data. In *Symposium on Large-Data Visualization and Graphics '01*, pages 93–99, 2001.

[30] R. Miyazaki, S. Yoshida, Y. Dobashi, and T. Nishita. A Method for Modeling Clouds Based on Atmospheric Fluid Dynamics. In *Proc. Pacific Graphics '01*, pages 363–372, 2001.

[31] Fabrice Neyret. Qualitative Simulation of Cloud Formation and Evolution. In *8th Workshop on Computer Animation and Simulation (EGCAS '97)*, pages 113–124, Wien, 1997. Eurographics, Springer.

[32] T. Nishita, Y. Dobashi, and E. Nakamae. Display of Clouds Taking into Account Multiple Anisotropic Scattering and Sky Light. In *Proc. SIGGRAPH '96*, pages 379–386. ACM, 1996.

[33] Ken Perlin. An Image Synthesizer. *Computer Graphics (Proc. SIGGRAPH '85)*, 19(3):287–296, 1985.

[34] S. Roettger, W. Heidrich, Ph. Slusallek, and H.-P. Seidel. Real-Time Generation of Continuous Levels of Detail for Height Fields. In *Proc. WSCG '98*, pages 315–322. EG/IFIP, 1998.

[35] S. Roettger, M. Kraus, and Th. Ertl. Hardware-Accelerated Volume and Isosurface Rendering Based on Cell-Projection. In *Proc. Visualization '00*, pages 109–116. IEEE, 2000.

[36] J. Rossignac and P. Borrel. *Multi-Resolution 3D Approximations for Rendering*, pages 455–465. Springer Verlag, 1993.

[37] G. Schaufler. Per-Object Image Warping with Layered Impostors. In *Proc. 9th Workshop on Rendering '98*, pages 145–156. Eurographics, 1998.

[38] G. Schussman and N. L. Max. Hierarchichal Perspective Volume Rendering Using Triangle Fans. In *Proc. TCVG Eurographics Workshop (VolumeGraphics '01)*, pages 309–320. IEEE/EG, 2001.

[39] J. Shade, S. Gortler, L. He, and R. Szeliski. Layered Depth Images. In *Proc. SIGGRAPH '98*, pages 231–242. ACM, 1998.

[40] P. Shirley and A. Tuchman. A Polygonal Approximation to Direct Scalar Volume Rendering. *ACM Computer Graphics (San Diego Workshop on Volume Visualization)*, 24(5):63–70, 1990.

[41] J. Stam and E. Fiume. Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes. In *Proc. SIGGRAPH '95*, pages 129–136. ACM, 1995.

[42] C. M. Stein, B. G. Becker, and N. L. Max. Sorting and Hardware Assisted Rendering for Volume Visualization. In *Symposium on Volume Visualization '94*, pages 83–89. IEEE, 1994.

[43] M. Weiler, R. Westermann, C. Hansen, K. Zimmerman, and Th. Ertl. Level-Of-Detail Volume Rendering via 3D Textures. In *Volume Visualization and Graphics Symposium '00*, pages 7–13. IEEE, 2000.

[44] R. Westermann and Th. Ertl. Efficiently Using Graphics Hardware in Volume Rendering Applications. In *Computer Graphics*, Annual Conference Series, pages 169–177. ACM, 1998.

[45] J. Wilhelms and A. van Gelder. A Coherent Projection Approach for Direct Volume Rendering. *Computer Graphics*, 25(4):275–284, 1991.

[46] P. L. Williams and N. L. Max. A Volume Density Optical Model. In *Computer Graphics (Workshop on Volume Visualization '92)*, pages 61–68. ACM, 1992.

[47] P. L. Williams, N. L. Max, and C. M. Stein. A High Accuracy Volume Renderer for Unstructured Data. *Transactions on Visualization and Computer Graphics*, 4(1):37–54, 1998.

[48] Peter L. Williams. Visibility Ordering Meshed Polyhedra. *ACM Transactions on Graphics*, 11(2):103–126, 1992.

[49] J. C. Xia, J. El-Sana, and A. Varshney. Adaptive Real-Time Level-of-Detail Based Rendering for Polygonal Models. *Trans. on Visualization and Computer Graphics*, 3(2):171–183, 1997.

[50] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution Tetrahedral Framework for Visualizing Regular Volume Data. In *Proc. Visualization '97*, pages 135–142. IEEE, 1997.

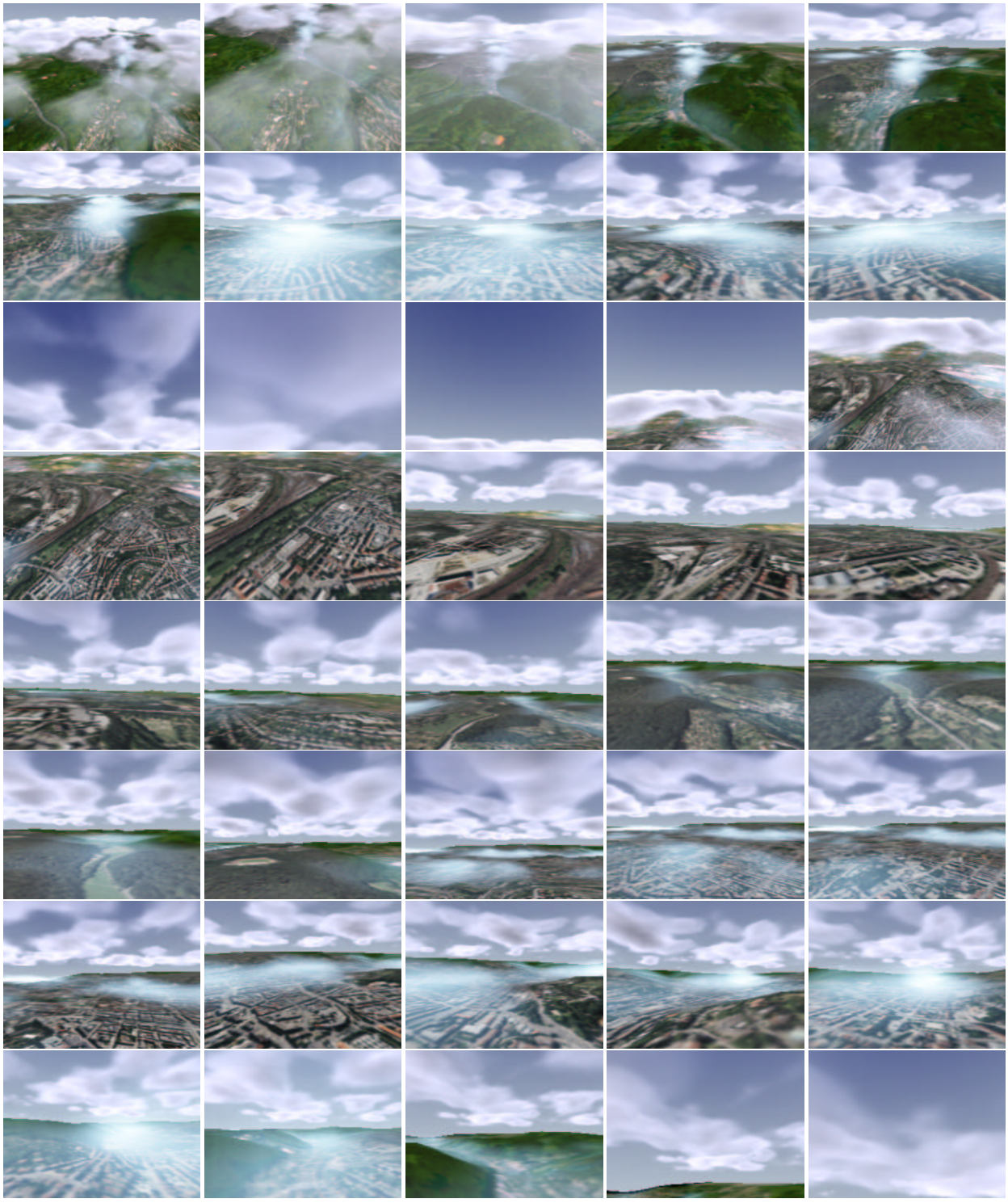

Figure 4: Scenes from a flight above Stuttgart showing volumetric clouds and ground fog in real-time.