

Pre-integrated Flow Illustration for Tetrahedral Meshes

N. A. Svakhine¹, D. Ebert¹, E. Tejada², T. Ertl² and K. Gaither³

¹Purdue University

²University of Stuttgart

³Texas Advanced Computing Center, University of Texas, Austin

Abstract

Previous work has demonstrated the clarity and usefulness of illustrative techniques for visualizing flow data. However, previous systems were limited to applying these techniques to uniform grids. Since unstructured grids have emerged as a common basis for computing flow simulations, we present a method to apply and extend the flow illustration approach to tetrahedral meshes using pre-integrated GPU-accelerated raycasting. Our illustrative rendering techniques can also be applied for other pre-integrated volume rendering systems. Additionally, we explore new feature illustration techniques for flow visualization.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Interaction techniques; I.3.7 [Computer Graphics]: Color, shading;

1 Introduction

In recent works [SJEG05], we have seen the use of illustrative techniques [ER00] applied to three-dimensional flow datasets. The usage of these illustrative effects has been very effective in removing the usual clutter associated with three-dimensional flow visualizations. However, the approach has been limited to regular rectangular grids. In this work, we are applying several approaches from tetrahedral rendering research to extend and apply illustrative effects to unstructured volumetric data.

2 Background

Many flow visualization techniques have been developed recently (e.g., [Wij02]) and are effective for 2D flows, however the density of the representations often limits their effectiveness for 3D flows, particularly in viscous flows. As an alternative, illustration has been considered as a viable approach; in [SLM02] various NPR techniques are applied to flow volumes, showing the potential of volume illustration [ER00].

Svakhine et al. [SJEG05] fused flow visualization methods with hardware-accelerated volume illustration algorithms [SE03] and NPR techniques. This method, however, is limited to regular grids, while flow simulations are very often performed on non-regular grids and visualization tech-

niques that operate on these grids do not map directly to GPU textures.

The first approach proposed to interactively render tetrahedra was the Projected Tetrahedra algorithm [ST90]. GPU-accelerated tetrahedral rendering (e.g., [RKE00, WKME03b]) addresses the rendering performance, while recent works in pre-integrated volume rendering provide several improvements [EKE01, WKME03a, KQE04].

3 Exposition

In this section, we describe the modifications introduced in the GPU-based raycasting algorithm for tetrahedral meshes to generate illustrative effects.

3.1 Pre-integrated Volume Rendering

Hardware-accelerated pre-integrated volume rendering pre-computes the ray integral for each rendering primitive using the ray entry and exit point values ($s^{(f)}$ and $s^{(b)}$), the thickness of the primitive l , and stores these values in the table texture. During the rendering step, for each ray, the ($s^{(f)}$, $s^{(b)}$, l) values (see Figure 1) are extracted (in the fragment shader) and used as indices for pre-integration table lookup. To estimate the gradients, we use barycentric interpolation [WKME03a], which is sufficient for datasets with

good resolution where regions of interest are highly sampled.

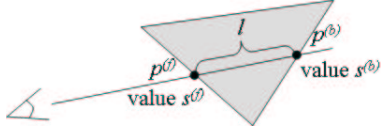


Figure 1: Ray traversal for a tetrahedral raycaster. The information gathered from the current tetrahedron is depicted.

3.2 Adding Illustrative Effects

The illustrative effects we present are accomplished by modifying either the pre-integrated table, as done in [ME04] for regular volumes, or the fragment shader. We introduce illustrative effects based on gradient enhancement, silhouetting, curvature enhancement and banding as described below. For that, a scaling mask m , which depends on the illustrative effect, is computed and used to scale the extinction coefficient linearly to modify the associated color $\tilde{C}(s^{(f)}, s^{(b)}, l)$ and opacity $\alpha(s^{(f)}, s^{(b)}, l)$ stored at each entry $(s^{(f)}, s^{(b)}, l)$ of the pre-integrated table. The masked opacity $\alpha^{(m)}$ and masked associated color $\tilde{C}^{(m)}$ are defined as:

$$\alpha^{(m)}(s^{(f)}, s^{(b)}, l) = 1 - e^{(m \log(1 - \alpha(s^{(f)}, s^{(b)}, l)))}, \quad (1)$$

$$\tilde{C}^{(m)}(s^{(f)}, s^{(b)}, l) = \frac{\alpha^{(m)}(s^{(f)}, s^{(b)}, l)}{\alpha(s^{(f)}, s^{(b)}, l)} \tilde{C}(s^{(f)}, s^{(b)}, l). \quad (2)$$

View-dependent gradient-based effects. Illustrative effects can be achieved by masking each entry $(s^{(f)}, s^{(b)}, l)$ of the pre-integrated table using the approximated view-dependent gradient magnitude given by $(s^{(b)} - s^{(f)})/l$. While not as accurate as the full gradient magnitude enhancement [ER00], it tends to highlight parts of essential high-gradient dataset features. The view-dependant gradient magnitude can be used to generate different effects (e.g. gradient magnitude isolines, low/negative gradient magnitude sample cull). For instance, the effect shown in Figure 2 was obtained by defining the mask m as

$$\omega \cdot \left(\left| \frac{s^{(f)} - s^{(b)}}{l} \right| \right)^p, \quad (3)$$

where ω is an artificial used-defined opacity scaling factor and p is a scalar that controls how much the gradient magnitude affects the opacity of the pre-integrated table entry.

Curvature-based contour effects. Curvature-based illustration (Figure 2) is obtained by estimating the curvature for the segment between $s^{(f)}$ and $s^{(b)}$ as $\|(\vec{n}^{(b)} - \vec{n}^{(f)})/l\|$. The mask m used in this case is given by

$$\omega \cdot \left(\frac{\|\vec{n}^{(f)} - \vec{n}^{(b)}\|}{l} \right)^p. \quad (4)$$

Banding effects. Banding is achieved by scaling the entries

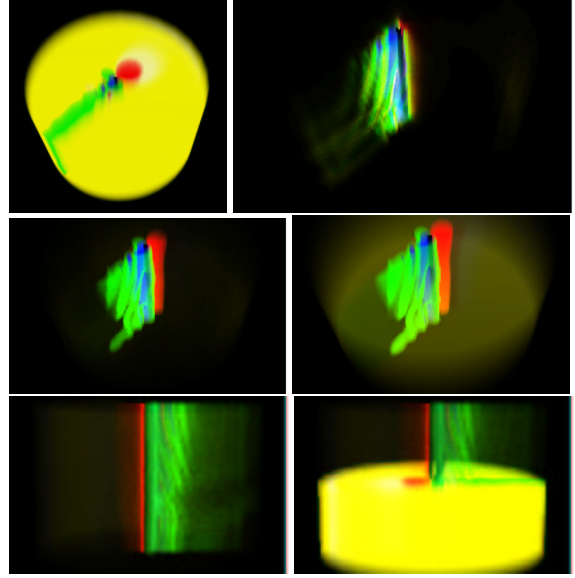


Figure 2: The Cylinder dataset. From top to bottom, left to right: volume rendering with no enhancement, view-dependent gradient enhancement; curvature-based illustration with two different p values; banding (and mixed banding-volume rendering) of the scalar value.

of the pre-integrated table with the mask:

$$((1 + \sin(o_o \pi k)) \cdot 0.5)^p. \quad (5)$$

where o_o is the value to mask (e.g. scalar value, gradient magnitude). Figure 2 shows banding of the gradient magnitude of the Cylinder dataset.

Gradient-based contour effects (silhouettes). As the work by Engel et al. [ME04] also describes, a similar approach can be used for contour enhancement. The contour/silhouette information can be extracted from the dataset by calculating $\phi = \langle \vec{n}, \vec{v} \rangle$, where \vec{n} is the normalized gradient vector, \vec{v} is the view vector [ER00]. A one-dimensional opacity transfer function $\alpha(\phi)$ is then used to highlight areas where ϕ is close to 0, showing only the samples where \vec{v} and \vec{n} are near orthogonal, which corresponds to a contour line of a surface. This illustrative effect is shown in Figure 5.

Gradient-based masked contour effects (masked silhouettes). Another way of obtaining gradient-based contours/silhouettes (Figure 3) is by masking the entries of the pre-integrated table similarly to the view-dependent gradient-enhancement effect. The mask m is obtained using the following formula

$$g(\|\nabla f\|) \cdot (1 - |\phi|)^p, \quad (6)$$

where ∇f is the gradient and $g(\cdot)$ is a windowing function (usually a linear function clamped to $[0, 1]$) used to restrict the detection of contours to the interfaces between different materials.

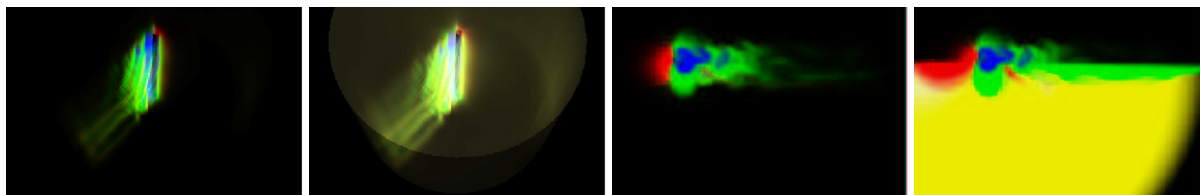


Figure 3: Masked silhouettes for the Cylinder dataset. From left to right: silhouettes, silhouettes and surface rendering, top view of the silhouettes, top view of silhouettes and volume rendering.

4 Implementation

We modified the implementation of our single-pass raycaster [TE05] to include the illustration techniques described above. As before, the data structure holding the tetrahedral mesh is stored in a set of textures used by the shader to calculate, for each iteration of the ray integral computation, the current tetrahedron and its entry and exit points. As in previous work [WKME03a] the values at these points are interpolated from the values stored at the vertices of the tetrahedron using barycentric coordinates. The values obtained and the distance between the points are used to fetch the contribution of the ray segment to the ray integral from the pre-integrated table modified by our techniques. We also implemented the logarithmic sampling proposed by Kraus et al. [KQE04] for the pre-integrated table.

Depending on the illustration technique applied to the data, modifications to the raycaster were introduced either in the pre-integrated table computation or in the shader. These modifications are described in the following paragraphs.

Gradient enhancement. View-dependent gradient enhancement is achieved by scaling each entry of the pre-integrated table using Equations 1 and 2 and the mask given by Equation 3. Therefore, the only modification to the raycaster is introduced in the computation of the pre-integrated table.

Silhouettes. Silhouetting is implemented as in the work by Meissner and Engel [ME04]. A pre-integrated table is calculated as usual but instead of using the interpolated values $s^{(f)}$ and $s^{(b)}$ at the entry and exit points respectively to perform the color and opacity fetch, we construct the pre-integrated table indexed by $(\phi^{(f)}, \phi^{(b)}, l)$.

Masked silhouettes. For masked silhouettes given by Equation 6, since ϕ can only be calculated in the fragment shader, the masking takes places during rendering using Equations 1 and 2.

Curvature-based. As for masked silhouettes, the mask in this case can be calculated only during rendering. Therefore, we modify the shader to apply the mask given by Equation 4.

Banding. Banding is achieved by masking the entries of the pre-integrated table using Equation 5. No modification is introduced into the rendering process.

5 Results

In this section we present performance results and a discussion of the visual results obtained. The tests were performed on a standard PC equipped with a NVidia GeForce 7800 GTX (512MB) graphics card. The target viewport size was 640×480 . Table 1 shows the performance results for the illustration techniques presented. We notice that the performance change is depreciable even for datasets with millions of tetrahedra.

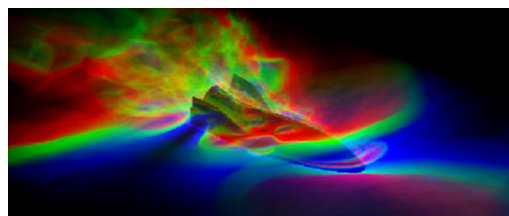


Figure 4: Shock dataset: silhouettes illustration.

	Combustion	Cylinder	Shock
Num. tetrahedra	215040	624960	1923048
Volume rendering	4.74	5.09	2.88
Gradient enhanced	4.68	5.12	2.99
Curvature-based	5.26	4.96	2.51
Banding	4.68	5.12	2.24
Silhouettes	5.02	4.98	2.94
Masked silhouettes	4.99	5.05	2.78

Table 1: Performance in frames per second for the test datasets with the illustration techniques presented.

From the set of figures shown above, we can see that flow structures can be extracted from the data using the illustrative effects described. For the case of the Cylinder (Figures 2-3), Shock (Figure 4), and Combustion Chamber (Figure 5) datasets, flow structures were extracted by applying illustrative effects without needing to spend much time in finding a proper transfer function, as is generally needed with normal volume rendering.

6 Conclusion and Future Work

Volume illustration applied to flow datasets helps gain insight into the structures contained in the flow data. In previous work [SJEG05], illustration techniques were used exclusively for flow datasets resampled on regular grids. However, a significant number of simulations are computed on

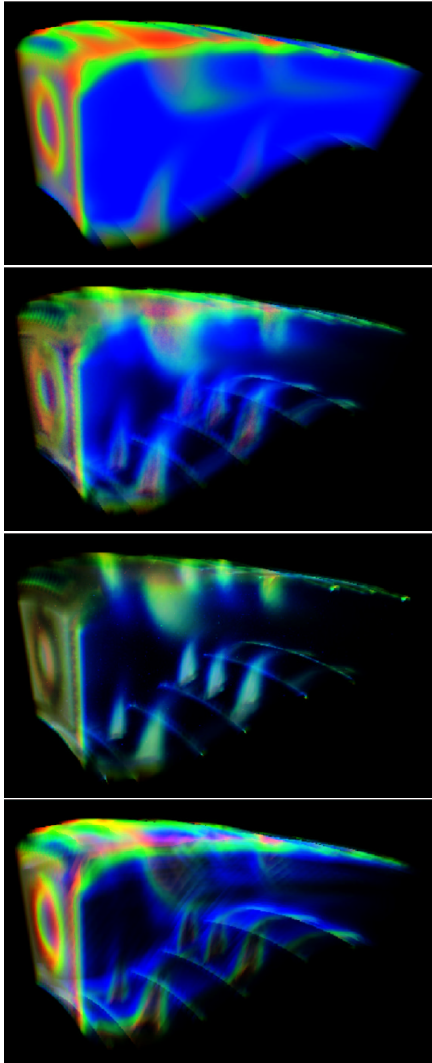


Figure 5: Illustrative effects for the Combustion Chamber dataset. From top to bottom: no illustration, banding, gradient enhancement, and silhouettes. Notice that the same transfer function is used for all cases except the silhouettes rendering where the transfer function is indexed by $(\phi^{(f)}, \phi^{(b)}, I)$.

unstructured grids. Applying illustrative effects directly to tetrahedral grids provides a direct rendering approach. Although we have shown how these effects can be applied to such grids, there are still issues to be solved, such as a semiautomatic setting of the parameters used for each technique, and the removing of artifacts that arise due to the constant gradient within each tetrahedron. Another promising research direction is employing the extra local flow properties (such as curl, vorticity, second directional derivatives

etc.) for calculating the illustrative enhancements highlighting flow features of interest.

Acknowledgments

This work was partially supported by the German Academic Exchange Service (DAAD) with grant A/04/08711, US National Science Foundation (grants NSF ACI-0081581, NSF ACI-0121288, NSF ACI-0328984, NSF IIS-0098443, NSF ACI-9978032, and NSF ACI-0222675) and Adobe Systems Incorporated. The authors would like to thank Martin Kraus, University of Stuttgart, for valuable discussions.

References

- [EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proc. of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware* (2001), pp. 9–16.
- [ER00] EBERT D., RHEINGANS P.: Volume illustration: non-photorealistic rendering of volume models. In *Proc. of IEEE Visualization* (2000), pp. 195–202.
- [KQE04] KRAUS M., QIAO W., EBERT D. S.: Projecting tetrahedra without rendering artifacts. In *Proc. of IEEE Visualization* (2004), pp. 27–34.
- [ME04] MEISSNER M., ENGEL K.: Pre-integrated non-photorealistic volume rendering. In *Proc. of VMV 2004* (2004), pp. 437–445.
- [RKE00] ROTTGER S., KRAUS M., ERTL T.: Hardware-accelerated volume and isosurface rendering based on cell-projection. In *Proc. of IEEE Visualization* (2000), pp. 109–116.
- [SE03] SVAKHINE N., EBERT D.: Interactive volume illustration and feature halos. *Pacific Graphics '03 Proceedings 15*, 3 (2003), 67–76.
- [SJEG05] SVAKHINE N. A., JANG Y., EBERT D. S., GAITHER K. P.: Illustration and photography inspired visualization of flows and volumes. In *Proc. of IEEE Visualization* (2005), p. 87.
- [SLM02] STOMPPEL A., LUM E. B., MA K.-L.: Feature-enhanced visualization of multidimensional, multivariate volume data using non-photorealistic rendering techniques. In *Proc. of Pacific Graphics* (2002).
- [ST90] SHIRLEY P., TUCHMAN A.: A polygonal approximation to direct scalar volume rendering. In *Proc. of Volume Visualization* (1990), pp. 63–70.
- [TE05] TEJADA E., ERTL T.: Large Steps in GPU-based Deformable Bodies Simulation. *Simulation Practice and Theory* 13, 9 (2005), 703–715.
- [Wij02] WIJK J. V.: Image based flow visualization. *ACM Transactions on Graphics* 21, 3 (2002), 745–754.
- [WKME03a] WEILER M., KRAUS M., MERZ M., ERTL T.: Hardware-based ray casting for tetrahedral meshes. In *Proc. of IEEE Visualization* (2003), p. 44.
- [WKME03b] WEILER M., KRAUS M., MERZ M., ERTL T.: Hardware-based view-independent cell projection. *IEEE Transactions on Visualization and Computer Graphics* 9, 2 (2003), 163–175.