

Bivariate Transfer Functions on Unstructured Grids

Yuyan Song^{†1} and Wei Chen^{‡2} and Ross Maciejewski^{†1} and Kelly P. Gaither^{§3} and David S. Ebert^{†1}

¹Purdue University Rendering & Perceptualization Lab, Purdue University, USA

²State Key Lab of CAD&CG, Zhejiang University, China

³Texas Advanced Computing Center, USA

Abstract

Multi-dimensional transfer functions are commonly used in rectilinear volume renderings to effectively portray materials, material boundaries and even subtle variations along boundaries. However, most unstructured grid rendering algorithms only employ one-dimensional transfer functions. This paper proposes a novel pre-integrated Projected Tetrahedra (PT) rendering technique that applies bivariate transfer functions on unstructured grids. For each type of bivariate transfer function, an analytical form that pre-integrates the contribution of a ray segment in one tetrahedron is derived, and can be precomputed as a lookup table to compute the color and opacity in a projected tetrahedron on-the-fly. Further, we show how to approximate the integral using the pre-integration method for faster unstructured grid rendering. We demonstrate the advantages of our approach with a variety of examples and comparisons with one-dimensional transfer functions.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction and Motivation

Multidimensional transfer functions are widely used for feature enhancement on rectilinear volume datasets [KKH02]. Concerning the efficiency, one common choice is a bivariate transfer function, which is built upon the two-dimensional histogram of the scalar quantity and the gradient magnitude within the underlying dataset [KD98]. This induces one challenging task, that is, how to effectively modulate a transfer function for visualizing complex structures within the volume data. In the past decades, many automatic or semi-automatic two-dimensional transfer function design approaches [PLB*01, KPI*03] have been proposed.

No matter how complicated a two-dimensional transfer function is, its color and opacity maps can be represented and manipulated with a collection of specific design toolkits, such as iso-region, Gaussian, triangle, and sine curve based classification widgets, as shown in Figure 1. Other geometric

shapes can also be used, such as trapezoids, tents, boxes and ramps presented in [KG01]. The Gaussian transfer function approach [KPI*03] employs a sum of Gaussians to approximate a transfer function, which can be analytically integrated over a line segment under the assumption that data values vary linearly between two sampled points.

While two-dimensional transfer functions have been popularized for rectilinear volume datasets, most unstructured grid rendering algorithms are limited to one-dimensional transfer functions, leaving materials, material boundaries or subtle variations along boundaries undiscovered.

Our work introduces a high-quality unstructured volume rendering algorithm, taking advantage of the power of two-dimensional transfer functions by means of a new pre-integrated bivariate transfer function scheme. For volume rendering, we utilize the Projected Tetrahedra (PT) due to its efficient parallelization, and we employ the use of Gaussian, triangle wave and sine wave transfer functions to enhance features in complex unstructured grids. Furthermore, we simplify the rendering computations by subdividing an integral range into small subintervals and accumulate their contributions through computationally inexpensive analyti-

[†] Email: {song7|rmacieje|ebertd}@purdue.edu

[‡] Email: chenwei@cad.zju.edu.cn

[§] Email: kelly@tacc.utexas.edu

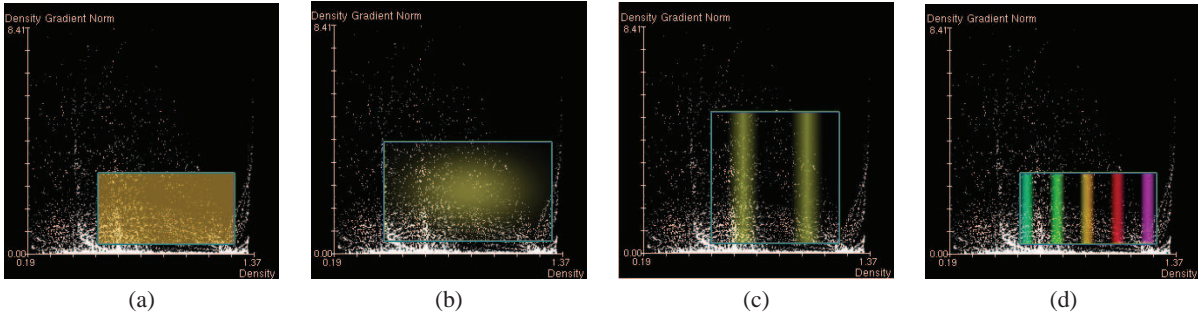


Figure 1: Four widgets used in two-dimensional transfer function design: (a) Uniform; (b) Gaussian; (c) Triangle wave; (d) Sine wave.

cal approximations. Figure 2 compares the results that are produced with conventional one-dimensional transfer functions and our new scheme.

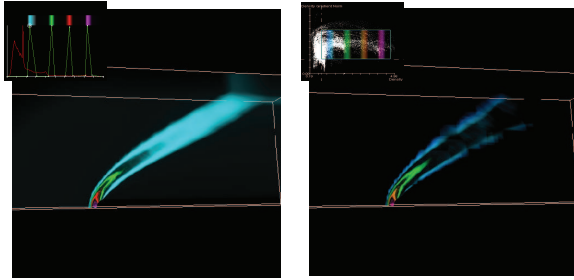


Figure 2: Rendering of the Bluntfin dataset with different pre-integrated rendering algorithms. (a) One-dimensional transfer function; (b) two-dimensional transfer function. Corresponding transfer functions are shown in the upper left-hand corners of the images. Homogeneous areas are made transparent by 2D transfer functions in (b) therefore the features can be better separated and visualized.

The remainder of this paper is organized as follows. Related work is discussed in Section 2. We present our solution for a hardware-accelerated PT rendering algorithm in Section 3. The analytical integrals of common bivariate transfer functions are derived in Section 4. We introduce another simplified method for interactive rendering in Section 5. Section 6 presents experimental results and detailed discussions. Finally, we draw the conclusions in Section 7.

2. Related Work

Previous work on visualizing unstructured grids can be roughly divided into two classes. The first category uses a backward projection scheme, ray casting, that resamples the data along the ray from the viewpoint and composites the contributions. Alternatively, the standard volume splatting algorithm [CM93] processes each element individually by computing and accumulating its contribution in image space.

The projected tetrahedra algorithm [ST90] is another object-space approach which decomposes the projected shape of a tetrahedron into multiple triangles and computes their contributions individually. It is quite suitable for hardware-accelerated volume rendering for tetrahedral grids or even more complex unstructured meshes, e.g., the work presented in [KQE04].

Pre-integration techniques [EKE01] can reduce sampling artifacts by pre-computing the integration in a small sampling interval. It is useful in hardware-accelerated regular volume rendering [HKRs*06] because it improves quality without the high computational overhead. This scheme has also been extended to unstructured grid rendering [MA04, SET*06]. However, only the use of one-dimensional transfer functions or partial pre-integration is employed, where as our work employs two-dimensional transfer functions.

3. GPU-based Projected Tetrahedra Rendering

Our PT rendering algorithm is built upon the hardware-accelerated approach introduced in [KQE04]. Every stage of the PT pipeline can be parallelized using programmable graphics hardware. Challenges in this include how to perform cell culling to preserve the sparsity of the underlying data, and how to perform the cell decomposition during runtime. In this section we describe our schemes that leverage the newest features of programmable graphics hardware to accomplish cell-based classification and decomposition. We also demonstrate how to incorporate two-dimensional transfer function designs within the framework of a pre-integrated projected tetrahedra rendering pipeline.

3.1. Conservative Cell-based Culling

To reduce the number of processed cells, without sacrificing quality, we design a conservative cell culling technique based on the assumption that the two-dimensional value pairs (s, g) are linearly distributed in a tetrahedron. At the beginning, all cells are loaded into the video memory. Once the transfer function is modified, the cells will be validated

against the transfer function and the identifications of the valid cells are updated into an index buffer for rendering. To validate a cell, we check the two-dimensional value pairs of its four vertices in the CPU. If any of the vertices is in the valid range of the employed transfer function, i.e., the opacity after applying the transfer function is not zero, this cell is valid and should be rendered. In this way, cell culling is conservative, and only requires several arithmetic comparisons between the value ranges of the four vertices and the bounding box of each individual transfer function widget.

3.2. Cell Decomposition

The primary obstacle of cell decomposition in the GPU has historically been a lack of vertex neighbor information within the graphics pipeline. Our approach classifies the projected silhouette of a tetrahedron on the image plane into two classes, as shown in Figure 3 (a-b). After projection, the silhouette of a tetrahedron is either a triangle (the inside case), or a quadrilateral (the outside case). In both cases, the image space position and scalar value of the thickest point in the tetrahedron can be determined by computing the barycentric coordinates from the image space coordinates of four vertices. The decomposition yields three or four triangles, where each vertex contains the image and object space coordinates and scalar values of the corresponding front and back points for later rasterization. The rendering of a tetrahedron is performed in three steps:

The positions of four vertices in both the screen space and the object space are computed in the vertex processing stage.

The cell decomposition is carried out using the geometry shader. Perspective corrections are applied to the image space coordinates before the classification of the projected silhouette. For the inside case, the barycentric coordinates are computed and used to interpolate the values of the thickest point. For the outside case, the intersection parameters and the thickest point are calculated.

The object-space coordinates, scalars of the front and back points, and other parameters are used for volume illumination in the fragment processing stage.

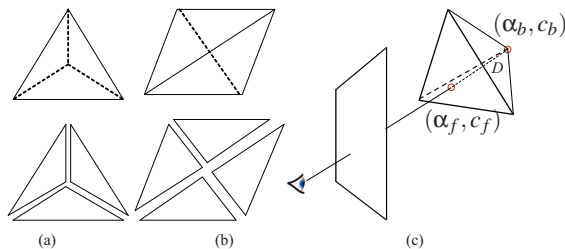


Figure 3: (a) The inside case; (b) The outside case; (c) The integral in a tetrahedron.

3.3. Cell Shading

Standard pre-integrated volume rendering [EKE01] precomputes a three-dimensional lookup table for each triple of (s_f, s_b, D) , where s_f and s_b are the scalar values of the entry point and exit point along an integral ray, and D is the thickness of the ray within the tetrahedron:

$$C = \int_0^D c\left(\left(1 - \frac{\lambda}{D}\right)s_f + \frac{\lambda}{D}s_b\right) \cdot \exp\left(-\int_0^\lambda \alpha\left(\left(1 - \frac{\rho}{D}\right)s_f + \frac{\rho}{D}s_b\right) d\rho\right) d\lambda$$

$$\bar{\alpha} = 1 - \exp\left(-\int_0^D \alpha\left(\left(1 - \frac{\lambda}{D}\right)s_f + \frac{\lambda}{D}s_b\right) d\lambda\right) \quad (1)$$

where $\alpha(\cdot)$ and $c(\cdot)$ denote the employed transfer functions for opacity and color respectively (see Figure 3 (c)). Please note that throughout this paper, we use the subscripts f and b to denote the properties of the entry and exit points.

One main drawback of this scheme is that the two-dimensional transfer function can not be used for unstructured grids [KQE04] because the integral presented in Equation 1 only relates to the input scalar values. If we incorporate the gradient magnitude into the transfer function (i.e., the $\alpha(\cdot)$ and $c(\cdot)$ items in the integral), a five dimensional pre-integration table is required.

In volume visualization, there are two data classification modes, namely, post-classification and pre-classification [HKRS*06]. Logically the standard PT rendering uses a post-classification shading scheme, which avoids densely sampling the input data by means of an one-dimensional pre-integrated kernel. To facilitate efficient multi-dimensional transfer function design, we propose two high-quality schemes based on the post-classification and pre-classification modes separately.

3.3.1. Post-classification with Analytical Pre-Integration

For each type of transfer function widget, we derive an analytic integration form in the post-classification mode, which will be explained in Section 4. The renderings for view rays involve the volume rendering integral evaluations with the pre-integrated tables for a certain type of transfer function. Theoretically, this mode yields better quality than that of the approximated mode, which is discussed below.

3.3.2. Pre-classification with Approximated Pre-Integration

We denote the color and opacity of the entry and exit points of a viewing ray as (c_f, α_f) and (c_b, α_b) , and assume that the color and opacity are linear between the two points in a

given tetrahedron (Figure 3 (c)):

$$C = \int_0^D \left((1 - \frac{\lambda}{D})c_f + \frac{\lambda}{D}c_b \right) \cdot \exp\left(-\int_0^\lambda \left((1 - \frac{\rho}{D})\alpha_f + \frac{\rho}{D}\alpha_b \right) d\rho\right) d\lambda$$

$$\bar{\alpha} = 1 - \exp\left(-\int_0^D \left((1 - \frac{\lambda}{D})\alpha_f + \frac{\lambda}{D}\alpha_b \right) d\lambda\right) \quad (2)$$

It yields:

$$C = c_f F(\alpha_f, \alpha_b, D) + c_b B(\alpha_f, \alpha_b, D)$$

$$\bar{\alpha} = 1 - \exp\left(-\frac{D}{2}(\alpha_f + \alpha_b)\right) \quad (3)$$

where

$$F(\alpha_f, \alpha_b, D) = \int_0^D \left(1 - \frac{\lambda}{D}\right) \exp\left(-\int_0^\lambda \left((1 - \frac{\rho}{D})\alpha_f + \frac{\rho}{D}\alpha_b \right) d\rho\right) d\lambda$$

$$B(\alpha_f, \alpha_b, D) = \int_0^D \frac{\lambda}{D} \exp\left(-\int_0^\lambda \left((1 - \frac{\rho}{D})\alpha_f + \frac{\rho}{D}\alpha_b \right) d\rho\right) d\lambda$$

We pre-compute $F(\alpha_f, \alpha_b, D)$ and $B(\alpha_f, \alpha_b, D)$ for all triples of (α_f, α_b, D) and store them in a 3D texture. In the fragment processing stage, (α_f, α_b, D) is used as an index to access the corresponding values $F(\alpha_f, \alpha_b, D)$ and $B(\alpha_f, \alpha_b, D)$ from the pre-computed texture. Thereafter, Equation 3 can be quickly evaluated with simple fragment instructions. Note that the determination of $(c_f, c_b, \alpha_f, \alpha_b)$ is independent of the pre-integration table, and thus can be computed with any two-dimensional transfer function. This is significantly different from the standard pre-integrated PT rendering algorithm with a one-dimensional transfer function [KQE04]. Our approach is similar to the partial pre-integration method that was discussed in [MA04]. We go beyond the work to derive the approximation method that is more suitable for 2D transfer function renderings.

This mode may lead to unpleasant results if we use front and back points directly after cell decomposition because the transfer functions of the underlying tetrahedra may vary significantly along the viewing rays. We propose an approximation method that subdivides a viewing ray into a fixed number of equal length segments and computes the total contribution of the segments, where a segment's contribution can be easily obtained from the above pre-computed texture. The method is discussed in Section 5.

4. Analytical Bivariate Transfer Functions

Given the density and gradient magnitude of two ending points of a ray segment, (s_f, g_f) and (s_b, g_b) the values in the segment are linearly distributed. Ideally, the post-classification mode computes and integrates the color and

opacity at each point of the segment:

$$C = \int_0^D c\left(\left(1 - \frac{\lambda}{D}\right)s_f + \frac{\lambda}{D}s_b, \left(1 - \frac{\lambda}{D}\right)g_f + \frac{\lambda}{D}g_b\right) \cdot \exp\left(-\int_0^\lambda \alpha\left(\left(1 - \frac{\rho}{D}\right)s_f + \frac{\rho}{D}s_b, \left(1 - \frac{\rho}{D}\right)g_f + \frac{\rho}{D}g_b\right) d\rho\right) d\lambda$$

$$\bar{\alpha} = 1 - \exp\left(-\int_0^D \alpha\left(\left(1 - \frac{\lambda}{D}\right)s_f + \frac{\lambda}{D}s_b, \left(1 - \frac{\lambda}{D}\right)g_f + \frac{\lambda}{D}g_b\right) d\lambda\right) \quad (4)$$

Straightforwardly implementing it would be quite time-consuming and be impractical for real applications. Our solution is based on a simple analysis to the domain space of the two-dimensional transfer functions. An arbitrary two-dimensional transfer function widget covers a region of the domain. Meanwhile, the range of the density and gradient magnitude along one ray segment in a tetrahedron forms a rectangle. The rectangle may be inside, outside or intersecting with the region covered by the transfer function widget in the two-dimensional histogram space (see Figure 4).

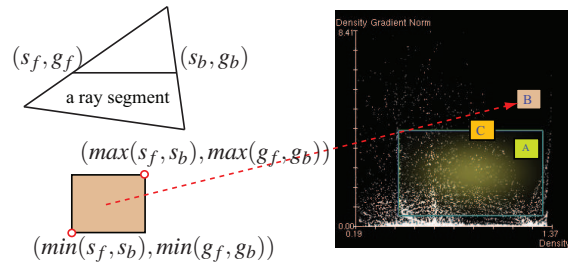


Figure 4: Three cases of the relationship between a value range in a ray segment and a two-dimensional transfer function widget.

For the first two cases, the ray segment is either visible or invisible. If it is visible (case A in Figure 4), the values of the ray segment can be fully modeled with the transfer function widget. If the transfer function widget itself has an analytical form, computing the color and opacity of the ray segment can be represented in a closed form. In the following sections, we show how to derive the forms for some representative transfer function widgets. In the third case, the ray segment can be adaptively divided into a list of smaller intervals, of which the value ranges are either inside or outside of the underlying transfer function widget. This process can be efficiently handled by means of the technique introduced in Section 5.

The analytic integrals along ray segments for common 2D transfer functions are derived below, including uniform transfer function, Gaussian transfer function, banded triangle waves and banded sinusoid waves. Uniform transfer functions are suitable for rendering homogeneous areas. Gaussian transfer functions are well suited for classification and visualization of local features in the transfer function do-

main. Triangle waves and sinusoid waves are used for illustrative rendering effects while sinusoid wave transfer functions tend to give smoother transitions between layers.

4.1. Uniform Transfer Functions

Uniform 2D transfer functions assign uniform color and opacities to all vertices of a cell. Therefore, one ray segment for this cell has the contribution:

$$C = \int_0^D c \times \exp\left(-\int_0^\lambda \alpha d\rho\right) d\lambda = \frac{c}{\alpha}(1 - e^{-\alpha D}) \quad (5)$$

4.2. Gaussian Transfer Functions

Often, a two-dimensional Gaussian transfer function widget is specified with a constant color and a two-dimensional Gaussian opacity function whose center and variation are (s_0, g_0) and (σ_s, σ_g) respectively. The two-dimensional Gaussian distribution function for the opacity is:

$$GTF(s, g) = \alpha_{max} \exp\left(-\left(\frac{(s-s_0)^2}{2\sigma_s^2} + \frac{(g-g_0)^2}{2\sigma_g^2}\right)\right) \quad (6)$$

where α_{max} is the maximum opacity of the Gaussian distribution.

For a Gaussian transfer function, the color is constant, meaning $color(s(\lambda)) = c$. Thus, we have:

$$s(\lambda) = s_f + \frac{s_b - s_f}{D} \times \lambda, \quad g(\lambda) = g_f + \frac{g_b - g_f}{D} \times \lambda \quad (7)$$

By specifying the $\alpha(\cdot)$ function to be the two-dimensional Gaussian distribution, and substituting the linear relationships Eq. 7 into Eq. 4, we have the contribution as:

$$\begin{aligned} C &= \int_0^D c \int_0^\lambda \exp\left(-\left(\frac{(s(\rho) - s_0)^2}{2\sigma_s^2} + \frac{(g(\rho) - g_0)^2}{2\sigma_g^2}\right)\right) d\rho d\lambda \\ &= \int_0^D c \int_0^\lambda \exp(-(A\rho + B)^2 - C) d\rho d\lambda \end{aligned}$$

Here, $k_s = \frac{s_b - s_a}{D}$, $k_g = \frac{g_b - g_a}{D}$, $d_s = s_a - s_0$, $d_g = g_a - g_0$,
 $A = \sqrt{\frac{k_s^2}{2\sigma_s^2} + \frac{k_g^2}{2\sigma_g^2}}$, $B = \frac{k_s \cdot d_s / 2\sigma_s^2 + k_g \cdot d_g / 2\sigma_g^2}{A}$, $C = \left(\frac{d_s^2}{2\sigma_s^2} + \frac{d_g^2}{2\sigma_g^2}\right) - B^2$.

The above integral can be further written as

$$\begin{aligned} C &= \int_0^D c \cdot \exp\left(-\alpha_{max} \frac{\sqrt{\pi}}{2} \frac{e^{-C}}{A} (erf(A\lambda + B) - erf(B))\right) d\lambda \\ &= c \cdot \exp(P \cdot erf(B)) \frac{1}{A} \int_B^{AD+B} \exp(-P \cdot erf(t)) dt \end{aligned}$$

here $erf(\cdot)$ is the error function and $P = \alpha_{max} \frac{\sqrt{\pi}}{2} \frac{e^{-C}}{A}$.

The two dimensional pre-integration table can be computed and looked up on-the-fly based on (P, t) . Thus,

the integral part of above equation can be evaluated as $table(P, AD + B) - table(P, B)$.

A special situation occurs when k_s and k_g are 0. In this case the ray contribution can be evaluated with the uniform transfer function in Section 4.1 with $\bar{\alpha} = \alpha_{max} \exp\left(-\left(\frac{(s_a - s_0)^2}{2\sigma_s^2} + \frac{(g_a - g_0)^2}{2\sigma_g^2}\right)\right)$. Figure 5 demonstrates the effects of a 2D uniform transfer function and a 2D Gaussian transfer function respectively. The 2D Gaussian transfer function clearly shows the vortex tubes and bow shock of the flow.

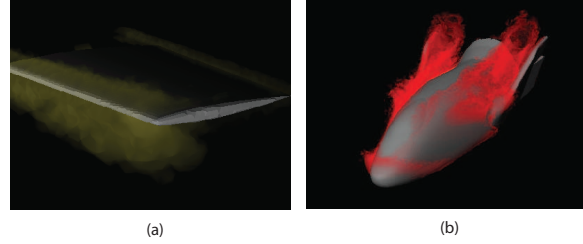


Figure 5: (a) Applying a uniform 2D transfer function to the M6 Wing dataset. (b) Two-dimensional Gaussian transfer function applied to the NASA X38 dataset and the vortex tubes and bow shock of the flow are clearly shown in this image.

4.3. Triangle Waves Transfer Functions

Triangle wave bivariate transfer functions are often used to create contour rendering effects. Figure 1 (c) is a typical transfer function with two triangle peaks. If we assume there is only one peak along the view ray in a tetrahedron cell, the contribution of the ray can be computed in an increasing and a decreasing part.

The opacity can be described as the following equations for the increasing and decreasing parts are:

$$\bar{\alpha}(\lambda) = \frac{\alpha_{max}}{D_t} \lambda, \quad \bar{\alpha}(\lambda) = \alpha_{max} \left(1 - \frac{\lambda}{D_t}\right) \quad (8)$$

where D_t is the segment length for interval $[0, \alpha_{max}]$.

To obtain the contribution of a ray segment from $\lambda = d_1$ to $\lambda = d_2$ from the increasing part, we can rewrite the integral as:

$$C = \int_{d_1}^{d_2} c \cdot \exp\left(-\int_{d_1}^\lambda k_\alpha \cdot \rho d\rho\right) d\lambda \quad (9)$$

for a uniform color triangle wave transfer function.

Equation 9 becomes:

$$\begin{aligned} C &= c \int_{d_1}^{d_2} \exp\left(-\frac{k_\alpha}{2} \rho^2 \Big|_{d_1}^\lambda\right) d\lambda \\ &= \sqrt{\frac{\pi}{k_\alpha}} c \cdot \exp\left(\frac{k_\alpha}{2} d_1^2\right) \left[erf\left(\sqrt{\frac{k_\alpha}{2}} d_2\right) - erf\left(\sqrt{\frac{k_\alpha}{2}} d_1\right)\right] \end{aligned}$$

where $\text{erf}(\cdot)$ is error function.

The contribution of a segment in the decreasing part from $\lambda = d_3$ to $\lambda = d_4$ can be evaluated as:

$$\begin{aligned} C &= \int_{d_3}^{d_4} c \cdot \exp\left(-\int_{d_3}^{\lambda} \alpha_{\max}\left(1 - \frac{\rho}{D_t}\right) d\rho\right) d\lambda \\ &= c \cdot \exp\left(-\alpha_{\max}\left(\frac{d_3 - D_t}{\sqrt{2D_t}}\right)^2\right) \int_{d_3}^{d_4} \exp\left(\alpha_{\max}\left(\frac{\lambda - D_t}{\sqrt{2D_t}}\right)^2\right) d\lambda \end{aligned}$$

The integral part of the above equation can be pre-integrated and stored as a two-dimensional look-up table with a look-up entry $(\frac{|\lambda - D_t|}{\sqrt{2D_t}}, \alpha_{\max})$. Note that the look-up entry is bounded with D_t as the maximum edge length in the grid. When there is no opacity variation between the front and back points, we have the same volume rendering integral as the uniform transfer function discussed in Section 4.1. Figure 6 (a) shows the effect for a banded triangle wave transfer function with no lighting applied on the Heatsink dataset.

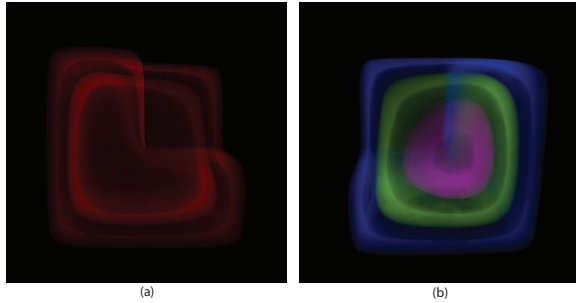


Figure 6: Volume rendering on the Heatsink dataset with (a) a banded triangle wave transfer function with no lighting; (b) a banded sinusoid wave transfer function with gradient-based diffuse lighting.

4.4. Banded Sinusoid Transfer Functions

Banded sinusoidal transfer functions describe the opacity with a section of a sinusoidal function (see Figure 1 (d)). If we assume there is at most one peak inside a tetrahedron cell and a ray segment's entry and exit points are within one cycle of the sinusoid, we will have the volume rendering integral for a uniform color segment as:

$$C = \int_{d_1}^{d_2} c \cdot \exp\left(-\int_{d_1}^{\lambda} \alpha_{\max} \sin\left(\pi \frac{\rho}{D_{\pi}}\right) d\rho\right) d\lambda \quad (10)$$

where D_{π} is the angular frequency.

Let $k_{\pi} = \pi/D_{\pi}$ and Equation 10 becomes:

$$\begin{aligned} C &= c \cdot \int_{d_1}^{d_2} \exp\left(\frac{\alpha_{\max}}{k_{\pi}} (\cos(k_{\pi}\lambda) - \cos(k_{\pi}d_1))\right) d\lambda \\ &= c \frac{D_{\pi}}{\pi} \cdot \exp\left(-\frac{\alpha_{\max}}{k_{\pi}} \cos(k_{\pi}d_1)\right) \int_{k_{\pi}d_1}^{k_{\pi}d_2} \exp\left(\frac{\alpha_{\max}}{k_{\pi}} \cos t\right) dt \end{aligned}$$

The above integral part can be pre-computed and stored

as a two-dimensional texture with $(t, \alpha_{\max}/k_{\pi})$ as the look-up entry of this integrated table. Figure 6 (b) demonstrate the effects of a 2D banded sinusoid wave transfer function with gradient-based diffuse lighting applied to the Heatsink dataset.

5. Approximated Bivariate Transfer Functions

Our pre-classification with approximated pre-integration scheme is valid given that the color and opacity vary linearly between the two ends of a viewing ray segment. Often, the color and opacity vary non-linearly or even a peak appears along the viewing ray between the entry and exit points of a tetrahedron. However, if the viewing ray is divided into a set of segments, each segment can be seen as having the color and opacity vary linearly between its two ends. The contribution of the ray can be approximated if each segment's contribution is known.

Our approximation method divides a viewing ray into a fixed number of equal length segments. For each segment, its two ends can have two-dimensional values linearly interpolated along the ray, and their corresponding colors and opacities can also be obtained. The segment's contribution can then be looked up from the pre-integration texture provided by our pre-classification with approximated pre-integration scheme. The total contribution of the viewing ray is computed by blending the segments.

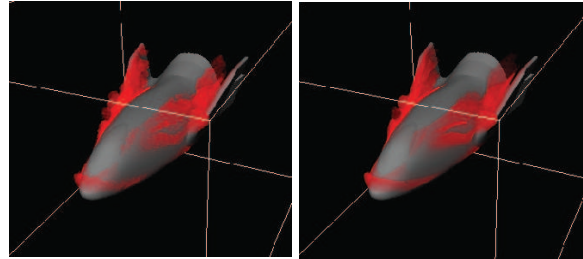


Figure 7: Rendering results of the NASA X38 dataset with the analytic form (left) and approximated pre-integration mode (right). A two-dimensional Gaussian transfer function is used.

Figure 7 compares the effects with the analytical and approximated pre-integration schemes. Both share a two-dimensional Gaussian transfer function, while the latter uses 10 samples along viewing rays for approximation. The results show that both schemes perform well on revealing the internal structures of the dataset.

6. Results and Discussions

We have implemented and tested our system on a PC with a Xeon 2.0 GHZ CPU, 16.0GB RAM and a NVIDIA GeForce 8800 GTS video card. The Bluntfin, Heatsink, Cylinder, ON-ERA M6 Wing, NASA X38, and Delta Wing datasets are

rendered using our system. Table 1 lists the data configuration and performances with conventional one-dimensional transfer functions, the analytical two-dimensional Gaussian transfer functions, and the approximated pre-integrated 2D Gaussian transfer functions.

Table 1: Datasets and performance comparison in FPS. In the last column, the performances with the analytical and approximated bivariate transfer functions are reported.

Data	#Tetrahedra	1D	2D Analy.	2D Approx.
Blunt Fin	224874	1.27	0.16	0.91
Heatsink	121668	1.91	0.23	1.00
Cylinder	762048	0.42	0.07	0.36
M6 Wing	287962	1.27	0.11	1.27
X38	1943483	0.18	0.04	0.18
Delta Wing	3853502	0.08	0.03	0.07

6.1. Volume Rendering

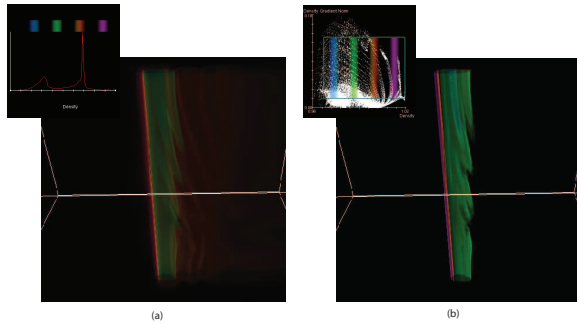


Figure 8: Rendering the Cylinder dataset using (a) an one-dimensional transfer function [SET*06]; (b) our approximated bivariate sinusoid transfer function.

Although unstructured grids can be visualized with one-dimensional transfer functions, our two-dimensional transfer function design scheme provides much more flexibility for achieving various feature enhancement effects. Figure 2 compares the results of a standard pre-integral PT rendering [KQE04] and our new scheme on visualizing Bluntfin dataset. The same banded patterns based on the scalar values can be seen in the transfer functions. It is apparent that our result (Figure 2 (b)) more clearly separates and displays the internal structures by making homogeneous areas transparent through 2D transfer functions.

Another comparison between 1D and 2D transfer functions is applied to the Cylinder dataset. Figure 8 (a) is generated following the method proposed in [SET*06], which employs gradient-based shading on top of a banding 1D transfer

function. Figure 8(b) is generated using our 2D banded sinusoid transfer function and it shows the same internal structures compared to (a) with much sharper material boundaries.

6.2. Illustrative Rendering Effects

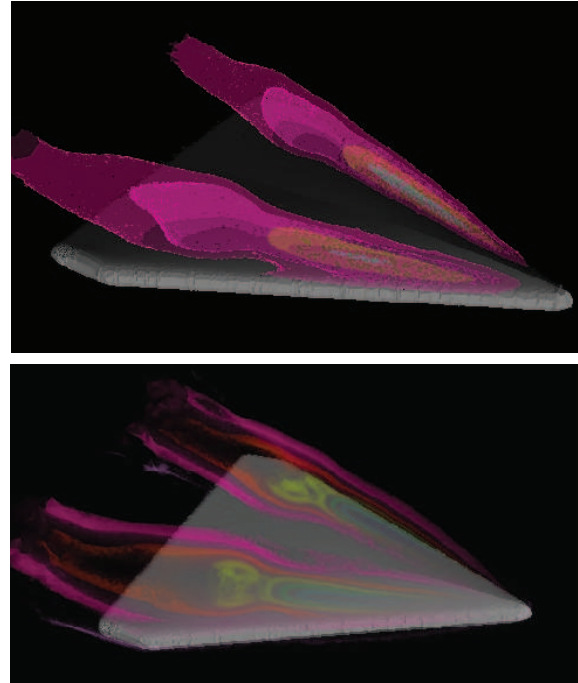


Figure 9: Illustrative rendering effects for the Delta Wing dataset: (top) semi-transparent rendering based on a banded triangle wave transfer function; (bottom) view-dependent enhancement using a bivariate triangle wave transfer functions.

We can also employ two-dimensional transfer function design and per-vertex gradient computation to perform feature enhancement. Various illustrative effects can be supported through our schemes for effectively depicting the oriented features. Semi-transparent isosurface layers can be produced when only the peak lines of the banded transfer functions are rendered. The view-dependent enhancement, or the standard silhouette enhancement, can also be fulfilled by incorporating the viewing vector \vec{V} in the pre-integrated opacity $\alpha_{pre-int}$:

$$\alpha = \alpha_{pre-int} \times |1 - \nabla g \cdot \vec{V}|^p$$

where p is the enhancement coefficient. ∇g is the unit vector in the direction of the gradient of the scalar function s where $\nabla g = \frac{\nabla s}{\|\nabla s\|}$.

Figure 9 shows the semi-transparent multi-layer rendering and silhouette illustration applied on the Delta Wing dataset respectively.

6.3. Discussion

For the derivation of the triangle wave and sinusoid wave transfer functions to be valid, there is at most one peak along a view ray in a cell. This requirement can not be met when the frequency of the banded transfer functions is too high. To determine if a frequency is too high for a dataset, data analysis on the minimum value difference among vertices of each cell is performed upon loading of the dataset. The corresponding frequency will be used as the frequency constraint in our system on the banded transfer function design.

Our implementation of GPU-based Projected Tetrahedra is successful. Cell decomposition and cell shading are both performed on the GPU with much less data needing to be streamed from the CPU. However, these improvements are significantly slowed down by our fragment processing, which appears as a bottleneck. The per fragment clipping and mathematical computations applied in our methods cause our system to be slower than other current systems. We expect our system to achieve much better speed with the significant improvement of fragment processing power of future 3D graphics hardware.

The pre-classification with approximated pre-integration mode performs much faster than the post-classification with analytical pre-integration mode, as seen in Table 1. Though the approximation error may not be guaranteed to below a certain level, the viewing ray can be better approximated if we increase the number of segments.

Though our method is slower than 1D transfer function based pre-integration method, our system supports bivariate or even higher dimensional transfer function rendering on unstructured grids. Such features have not yet been developed or exploited for unstructured grid rendering. Two-dimensional transfer function rendering, gradient shading, semi-transparent isosurfaces and silhouette rendering effects can be achieved through our system.

7. Conclusions

We have shown that our novel pre-integrated Projected Tetrahedra (PT) rendering technique is able to apply bivariate transfer functions on unstructured grids. Analytical forms that pre-integrate the contribution of a ray segment in one tetrahedron have been derived for common bivariate transfer function primitives. The pre-integration table is used to compute the color and opacity of ray segments on-the-fly. Moreover, approximated volume rendering of bivariate transfer functions is achieved by subdividing the integral range into small ones and accumulating their contributions.

In the future, we expect to further simplify the analytic form computation of the proposed bivariate transfer functions and derive more forms for other ones. More illustrative rendering techniques can be incorporated into our system for effective feature enhancements. An automated method for

subdividing the viewing ray into a set of segments so that the error is below a manageable level will be studied.

8. Acknowledgment

This work has been funded by the US Department of Homeland Security Regional Visualization and Analytics Center (RVAC) Center of Excellence (Nos. 0081581, 0121288), the US National Science Foundation (No. 0328984), and Natural Science Foundations China (No. 60873123).

References

- [CM93] CRAWFIS R., MAX N.: Texture splats for 3D scalar and vector field visualization. In *Proceedings of IEEE Visualization* (1993), pp. 261–266.
- [EKE01] ENGEL K., KRAUS M., ERTL T.: High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *Proceedings of Eurographics/SIGGRAPH Workshop on Graphics Hardware* (October 2001), pp. 9–16.
- [HKRS*06] HADWIGER M., KNISS J. M., REZK-SALAMA C., WEISKOPF D., ENGEL K.: *Real-time Volume Graphics*. A. K. Peters, Ltd., 2006.
- [KD98] KINDLMANN G., DURKIN J.: Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium On Volume Visualization* (1998), pp. 79–86.
- [KG01] KONIG A., GROELLER M. E.: Mastering transfer function specification by using VolumePro technology. In *Proceedings of the 17th Spring Conference on Computer Graphics* (2001), pp. 279–286.
- [KKH02] KNISS J., KINDLMANN G., HANSEN C.: Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics* 8, 3 (2002), 270–285.
- [KPI*03] KNISS J., PREMOZE S., IKITS M., LEFOHN A., HANSEN C., PRAUN E.: Gaussian transfer functions for multi-field volume visualization. In *Proceedings of IEEE Visualization* (2003), pp. 497–504.
- [KQE04] KRAUS M., QIAO W., EBERT D. S.: Projecting tetrahedra without rendering artifacts. In *Proceedings of IEEE Visualization* (2004), pp. 27–34.
- [MA04] MORELAND K., ANGEL E.: A fast high accuracy volume renderer for unstructured data. In *Proceedings of IEEE Symposium on Volume Visualization and Graphics* (October 2004), pp. 9–16.
- [PLB*01] PFISTER H., LORENSEN W., BAJAJ C., G.KINDLMANN, SCHROEDER W., AVILA L., MARTIN K., MACHIRAJU R., LEE J.: The transfer function bake-off. *IEEE Computer Graphics and Applications* 21, 3 (2001), 16–22.
- [SET*06] SVAKHINE N., EBERT D., TEJADA E., ERTL T., GAITHER K.: Pre-integrated flow illustration for tetrahedral meshes. In *Proceedings of the International Workshop on Volume Graphics* (2006), pp. 687–694.
- [ST90] SHIRLEY P., TUCHMAN A.: A polygonal approximation to direct scalar volume rendering. In *Proceedings of Volume Visualization* (1990), pp. 9–16.