Closed-Form Approximations to the Volume Rendering Integral with Gaussian Transfer Functions

Joe Kniss Simon Premože Milan Ikits
Aaron Lefohn Charles Hansen

UUCS-03-013

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

July 25, 2003

Abstract

In direct volume rendering, transfer functions map data points to optical properties such as color and opacity. We have found transfer functions based on the Gaussian primitive to be particularly useful for multivariate volumes, because they are simple and rely on a limited number of free parameters. We show how this class of transfer function primitives can be analytically integrated over a line segment under the assumption that data values vary linearly between two sampled points. Analytically integrated segment can then be composited using standard techniques.

Closed-Form Approximations to the Volume Rendering Integral with Gaussian Transfer Functions

Joe Kniss Simon Premože Milan Ikits
Aaron Lefohn Charles Hansen

1 Introduction

In direct volume rendering, data points are directly mapped to optical properties such as color and opacity that are then composited along the viewing direction into an image. This mapping is achieved using *transfer functions*. These functions have to be able to efficiently classify data features and produce various different outputs such as color, opacity, emission, phase function, etc. Typically these functions have many parameters that have to be set by the user by hand or through interactive exploration of the volume data. As the survey by Kindlmann [3] on transfer functions and generation methods shows, the process of creating expressive transfer functions can be a very time consuming and frustrating task. For multivariate volumes, this problem becomes even more daunting since the number of parameters grows with the number of dimensions, sometimes exponentially. It is therefore important, especially for multivariate datasets, to have transfer functions with simple expressions that rely on a limited number of free parameters. We have found transfer functions based on the Gaussian primitive to be particularly useful. We show how this class of transfer function primitives can be analytically integrated over a line segment under the assumption that data values vary linearly between two sampled points.

2 Closed-Form Approximation Derivation

The emission-absorption volume rendering equation over a line segment is defined as [5]:

$$I(a,b) = \int_a^b C\rho(v(u))\; e^{-\int_a^u \tau\rho(v(t))dt} du \tag{1}$$

Assuming constant color and extinction over the segment yields [4]:

$$I(a,b) = \int_a^b C\rho(v(u))\; e^{-\int_a^u \tau\rho(v(t))dt} du = \frac{C\alpha}{\tau} \tag{2}$$

where the opacity α is expressed as:

$$\alpha = 1 - e^{-\tau\int_a^b \rho(v(t))dt} \tag{3}$$

If we assume that data values vary linearly between samples along the ray, the opacity becomes:

$$\alpha = 1 - e^{-\tau l \int_0^1 \rho(v_1 + t(v_2 - v_1))dt} = 1 - e^{-\tau l\, \rho'(v_1, v_2)} \tag{4}$$

where $l = b - a$ and $\rho'(v_1, v_2)$ is the density integral over the segment. With a one dimensional density function we get [2]:

$$\rho'(v_1, v_2) \;=\; \int_0^1 \rho(v_1 + t(v_2 - v_1))dt = \frac{1}{v_2 - v_1}\int_{v_1}^{v_2} \rho(v)dv = \frac{R(v_2) - R(v_1)}{v_2 - v_1} \tag{5}$$

where R is the density integral function:

$$R(v) = \int \rho(z)dz \tag{6}$$

Now if we represent ρ with a multivariate Gaussian function:

$$\rho(\vec{v}) = e^{-(\vec{v}-\vec{c})^T \mathbf{K}^T \mathbf{K}\,(\vec{v}-\vec{c})} \tag{7}$$

with center $\vec{c}$ and linear transformation $\mathbf{K}$, the density integral can be expressed in closed form:

$$\begin{aligned}
\rho'(v_1, v_2) &= \int_0^1 \rho(\vec{v}_1 + t(\vec{v}_2 - \vec{v}_1))dt && (8)\\
&= \int_0^1 e^{-(\vec{v}_1 + t(\vec{v}_2-\vec{v}_1)-\vec{c})^T \mathbf{K}^T \mathbf{K}\,(\vec{v}_1 + t(\vec{v}_2-\vec{v}_1)-\vec{c})} dt && (9)\\
&= \frac{\sqrt{\pi}}{2}\,\frac{e^{-(\vec{v}_1-\vec{c})^T \mathbf{K}^T \mathbf{K}\,(\vec{v}_1-\vec{c}) + \frac{((\vec{v}_2-\vec{v}_1)^T \mathbf{K}^T \mathbf{K}\,(\vec{v}_1-\vec{c}))^2}{(\vec{v}_2-\vec{v}_1)^T \mathbf{K}^T \mathbf{K}\,(\vec{v}_2-\vec{v}_1)}}}{\sqrt{(\vec{v}_2-\vec{v}_1)^T \mathbf{K}^T \mathbf{K}\,(\vec{v}_2-\vec{v}_1)}} && (10)\\
&\quad \left[\mathrm{erf}\left(\frac{(\vec{v}_2-\vec{v}_1)^T \mathbf{K}^T \mathbf{K}\,(\vec{v}_2-\vec{c})}{\sqrt{(\vec{v}_2-\vec{v}_1)^T \mathbf{K}^T \mathbf{K}\,(\vec{v}_2-\vec{v}_1)}}\right) - \mathrm{erf}\left(\frac{(\vec{v}_2-\vec{v}_1)^T \mathbf{K}^T \mathbf{K}(\vec{v}_1-\vec{c})}{\sqrt{(\vec{v}_2-\vec{v}_1)^T \mathbf{K}^T \mathbf{K}(\vec{v}_2-\vec{v}_1)}}\right)\right] && (11)\\
&= \frac{\sqrt{\pi}}{2}\,\frac{S}{\|\vec{d}\|}(\mathrm{erf}(B) - \mathrm{erf}(A)) && (12)
\end{aligned}$$

where:

$$\begin{aligned}
A &= \frac{\vec{d}\cdot\vec{v}_1'}{\|\vec{d}\|} && (13)\\
B &= \frac{\vec{d}\cdot\vec{v}_2'}{\|\vec{d}\|} = A + \|\vec{d}\| && (14)\\
S &= e^{-\|\vec{v}_1'\| + A^2} && (15)
\end{aligned}$$

and:

$$\begin{aligned}
\vec{d} &= \vec{v}_2' - \vec{v}_1' && (16)\\
\vec{v}_1' &= \mathbf{K}(\vec{v}_1 - \vec{c}) && (17)\\
\vec{v}_2' &= \mathbf{K}(\vec{v}_2 - \vec{c}) && (18)
\end{aligned}$$

Proof: Assume $\mathbf{K}^T\mathbf{K} = \mathbf{I}$ and $\vec{c} = \vec{0}$. Then the integrand becomes:

$$e^{-(\vec{v}_1+t(\vec{v}_2-\vec{v}_1))\cdot(\vec{v}_1+t(\vec{v}_2-\vec{v}_1))dt} = e^{-\vec{v}_1\cdot\vec{v}_1+2t\vec{v}_1\cdot(\vec{v}_2-\vec{v}_1)+(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)dt} \tag{19}$$

After changing the integration variable to:

$$u = t\sqrt{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)} + \frac{(\vec{v}_2-\vec{v}_1)\cdot\vec{v}_1}{\sqrt{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}} \tag{20}$$

we obtain a closed form solution:

$$\begin{aligned}
\rho'(\vec{v}_1,\vec{v}_2) &= \int_0^1 e^{-\vec{v}_1\cdot\vec{v}_1+2t\vec{v}_1\cdot(\vec{v}_2-\vec{v}_1)+(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)dt} && (21)\\
&= \frac{e^{-\vec{v}_1\cdot\vec{v}_1+\frac{((\vec{v}_2-\vec{v}_1)\cdot\vec{v}_1)^2}{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}}}{\sqrt{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}} \int_{\frac{(\vec{v}_2-\vec{v}_1)\cdot\vec{v}_1}{\sqrt{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}}}^{\frac{(\vec{v}_2-\vec{v}_1)\cdot\vec{v}_2}{\sqrt{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}}} e^{-u^2}du && (22)\\
&= \frac{\sqrt{\pi}}{2}\,\frac{e^{-(\vec{v}_1\cdot\vec{v}_1)+\frac{((\vec{v}_2-\vec{v}_1)\cdot\vec{v}_1)^2}{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}}}{\sqrt{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}}\cdot && (23)\\
&\quad \left[\mathrm{erf}\left(\frac{(\vec{v}_2-\vec{v}_1)\cdot\vec{v}_2}{\sqrt{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}}\right) - \mathrm{erf}\left(\frac{(\vec{v}_2-\vec{v}_1)\cdot\vec{v}_1}{\sqrt{(\vec{v}_2-\vec{v}_1)\cdot(\vec{v}_2-\vec{v}_1)}}\right)\right] && (24)
\end{aligned}$$

The general case is obtained if we substitute $\vec{v}_1$ with $\mathbf{K}(\vec{v}_1-\vec{c})$ and $(\vec{v}_2-\vec{v}_1)$ with $\mathbf{K}(\vec{v}_2-\vec{v}_1)$.

3 Error Function Evaluation

There are many numerical approximations to the error function. One of the simplest approximations[1] is:

$$\mathrm{erf}(z) \approx 1 - \left(a_1 t + a_2 t^2 + a_3 t^3\right) e^{-z^2}, \quad z \geq 0 \tag{25}$$

$$t = \frac{1}{1 + .47047 * z}$$

$$a_1 = .3480242, \quad a_2 = -.0958798, \quad a_3 = .7478556$$

Equation 25 is only defined for $z \geq 0$. However, erf is an odd function and $\mathrm{erf}(-z) = -\mathrm{erf}(z)$.

References

[1] Milton Abramowitz and Irene A. Stegun. *Handbook of Mathematical Functions*. Dover, June 1974.

[2] K. Engel, M. Kraus, and T. Ertl. High-Quality Pre-Integrated Volume Rendering Using Hardware-Accelerated Pixel Shading. In *Eurographics / SIGGRAPH Workshop on Graphics Hardware '01*, pages 9–16, 2001.

[3] Gordon Kindlmann. Transfer functions in direct volume rendering: Design, interface, interaction. Siggraph Course Notes, 2002.

[4] Nelson Max, Pat Hanrahan, and Roger Crawfis. Area and volume coherence for efficient visualization of 3D scalar functions. In *Computer Graphics (San Diego Workshop on Volume Visualization)*, volume 24, pages 27–33, November 1990.

[5] Paolo Sabella. A rendering algorithm for visualizing 3D scalar fields. In *Computer Graphics (SIGGRAPH '88 Proceedings)*, volume 22, pages 51–58, August 1988.