MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Multi-resolution Multi-Field Ray Tracing: A mathematical overview

C. Gasparakis

TR99-15 December 1999

Abstract

A rigorous mathematical review of ray tracing is presented. The concept of fexible voxel formats as a means of a generic input to a graphics pipeline, along with a generic decoder is introduced. The blending equation is formalized in terms of function integrals. The issues pertaining to interpolation/classification order are defied and resolved. Opacity weighted color interpolation and its implications in lighting and compositing is discussed. In particular, the multi-resolution (along the ray) correction of the opacity-weighted color is derived. In addition, filtering of a supersampled image plane is shown to be expressible in terms of accumulation along a ray of auxiliary tensor fields. Furthermore, the correct continuum limit of the discrete opacity accumulation formula is presented.

IEEE Visualization 1999 Conference

MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

Multi-resolution Multi-field Ray Tracing: A mathematical overview

Dr. C. Gasparakis, Principal Senior Scientist,
Volume Graphics Organization, MERL

TR-99-15 March 1999

Abstract

A rigorous mathematical review of ray tracing is presented. The concept of flexible voxel formats as a means of a generic input to a graphics pipeline, along with a generic decoder is introduced. The blending equation is formalized in terms of function integrals. The issues pertaining to interpolation/classification order are defined and resolved. Opacity weighted color interpolation and its implications in lighting and compositing is discussed. In particular, the multi-resolution (along the ray) correction of the opacity-weighted color is derived. In addition, filtering of a supersampled image plane is shown to be expressible in terms of accumulation along a ray of auxiliary tensor fields. Furthermore, the correct continuum limit of the discrete opacity accumulation formula is presented.

To be submitted to IEEE Visualization 1999 Conference

1. First printing, TR99-15, July 1999

1 Introduction

Volume rendering is a well established and fastly growing field, with many applications in visualizing medical scans, geophysical and scientific datasets, among others. In this paper we focus our attention to the ray tracing approach to volume rendering. A volume rendering pipeline (in hardware or software) consists of some well established stages: the input (a general multi-field voxel), the decoder (which maps the input voxel to rgba), the interpolation unit (which can interpolate either voxel values or opacity weighted colors: $\tilde{\text{rgb}}\tilde{\text{a}}$), the lighting unit, and the compositing unit. In this paper we discuss the mathematics and some applications of the various stages of a general multi-field and multi-resolution ray-tracing pipeline.

This paper features:

- Flexible voxel formats, as generic input to a rendering pipeline. A voxel is understood to represent a collection of fields, each field corresponding to an attribute of the dataset in a given position in space. This concept is useful for multi-modal visualization [4]; for medical applications one can have multiple scans of the same anatomy, using a different acquisition mode (CT, MRI, PET, ultrasound). Also general 3D-scan converted fields, which can represent segmentation information, edgeness [6], shadows [11, 12], material stress and displacement fields, among others.

- Voxel decoder, which maps a (potentially multi-field) voxel to a rgba value. Such a mapping needs to be general enough as to accommodate for all usual cases of interest, yet constrained enough as to be easily implementable (in hardware or software). We introduce the concept of fixed-grammar variable length cascaded decoder, which can accommodate for most of the cases of interest. The decoder is a generalization of the transfer function concept [5], in a way that is well-suited for multi-field visualization.

- Interpolator unit [9], which examines the neighborhood of a sample, and assigns attributes to a sample, given the attributes of the voxel lattice around the sample. One can interpolate either voxel fields, or color-opacity fields.

In the case of voxel interpolation, one should allow for the ability to interpolate individual voxel subfields independently, and with potentially different interpolation modes. For example, one might want to interpolate intensity subfields linearly, but interpolate segmentation subfields either using a segmentation based probabilistic scheme, or simply do nearest neighbor interpolation (the reason for that is that there is no a priori notion of linear interpolation of segmentation fields, unless if one knows a priori (by construction) that the category field assignment is somehow monotonic).

In the case of color-opacity interpolation, one finds the rgba assignments of each of the voxels of a given sample, and interpolates those. Here there is a minor complication. One should consider the opacity-weighted colors and interpolate those, as was advocated in [2] to avoid the self-occlusion artifact. This follows from the mathematics of the blending equation, as we will prove for a general interpolation filter. This proof will rely heavily on a powerful definition of the blending process as a functional (Monte Carlo) integral. This approach is reminiscent to the path-tracing approach of [7].

We introduce our notation: We generically denote as c any r, g, or b component; a denotes the opacity. When those values refer to an average around sample s, we denote the average by $<>_s$. Let t denote transparency ($t = 1-a$). For a generic alpha weighted color component we use the usual tilde notation:

$$\tilde{c} \equiv c * a$$

We use capital letters for denoting the corresponding accumulated values: $\tilde{\mathrm{R}}$,$\tilde{\mathrm{G}}$,$\tilde{\mathrm{B}}$ (generically $\tilde{\mathrm{C}}$),A,T.

- Edgeness Modulation unit, which modulates the opacity-weighted color (generalizing the usual [8] or any lighting model, which is written in terms of c, and not $\tilde{\mathrm{c}}$) and the opacity [9]. The reason for expressing shading in terms of opacity-weighted colors is that the input of the shading stage is the output of the interpolation stage (which is $\tilde{\mathrm{c}}$).

- Blending unit, which calculates the contribution of the current sample when blended on the current ray, front to back. This is an integration

(or discrete summation) process along the ray. The correct continuum limit of the transparency accumulation is introduced. The theory is presented in a general way, so that it is independent of the parametrization of the line (i.e. independent of the choice of a local unit of length in the neighborhood of any sample on the ray). This formulation leads to a clean proof of the usual alpha correction ([13, pg133]), which is relevant when one supersamples along the ray [9], or in shear-warp ray tracing [3]. We examine the effect of supersampling along the ray, and we prove that one should alpha correct the interpolated alpha values, as opposed to interpolating the alpha corrected alpha values of the neighborhood. Furthermore, one needs to rescale the opacity-weighted colors, in a way that we specify.

- Finally, a pixel output unit, which in general has to filter the contributions of many adjacent rays to produce a pixel (supersampling on the image plane) [10]. It is shown that such a filtering process is equivalent to just using one ray, however taking into account higher derivative (tensor) fields, and accumulating those too.

2 Cascaded Multi-field Voxel Decoder

Traditionally, one thinks of voxel fields as intensity (of some sort) describing a volume. However, one can think of a voxel as a container of multiple fields, each describing different attributes of space. This can be multi-modal scans of a volume (CT, MRI, PET, SPECT, ultrasound) [4], category fields [1] (from manual or (semi)automatic segmentation), rgba pixel components, edgeness fields, or different state variables in scientific visualization (pressure, density, temperature, viscosity etc.). Other candidate fields can be encoding of depth, 3D stencil, shadows, fog, stress/displacement, voxelized embedded geometry etc. It is of interest for both software and hardware applications to be able to have a common framework of treating all special cases in a uniform fashion. Before presenting the general formalism, we look at a special case: Assume that we are looking for a transfer function of the form [13, pg. 89]

$$a = a(I, |\nabla I|).$$

[1] I would like to thank Vikram Simha for discussions on this

One can expand this function in terms of appropriate basis functions (fourier, wavelets, etc.) as

$$a = \tilde{\sum}_{m,n} a_{mn} * f_m(I) g_n(|\nabla I|).$$

In the above, $*$ stands for the usual multiplication and $\tilde{\sum}$ stands for the usual addition. However we use a special notation because we will generalize these operations in what follows. The above is a doubly infinite sum, which could be truncated given the sampling frequency of the dataset. Even if finite, the above decomposition would need to involve few terms, if it was to be useful. In what follows a scheme is proposed where one keeps a fixed number of generalized "summands", while allowing for arbitrary operations (logical or arithmetical).

The general formalism now follows: Let

$$V = (V_1, V_2, \ldots, V_n)$$

be a voxel V, with partial n voxel fields $V_1, \ldots, V_n$. A lookup table L_i acts on the field V_i to produce rgba values:

$$L_i : v \mapsto rgba_i(v), \quad \forall v \in V_i.$$

The notation indicates that each lookup table produces the partial rgba contribution of a given voxel field. Subsequently, one needs to combine those partial contributions, using logical or arithmetic operations. We denote by $*_{ij}$ the operator that combines the $rgba_i$ with $rgba_j$:

$$rgba_i *_{ij} rgba_j \mapsto rgba_{(ij)}.$$

This means that the operator $*_{ij}$ is really the tensor product of four operators, one for each rgba component. This scheme iteratively defines a tree, each node being the contraction of (at least) two parent nodes. A good compromise between full generality and realistic applications is achieved by using a fixed-grammar, variable width tree, according to

$$V = (F_3, F_2, F_1, F_0) \mapsto (L_3(F_3) *_{32} L_2(F_2)) *_{(32)(10)} (L_1(F_1) *_{10} L_0(F_0)). \quad (1)$$

As shown clearly in fig.1, The $*$ operators are composite operators, comprising of four component-wise operations, one for each of the color-opacity

channels. Useful operators (per rgba channel) are the sixteen logical operations, and some arithmetic ones, such as multiplication, multiplication of the complements with respect to one, addition, average, minimum, maximum, among others. It is to be emphasized that such a decoding scheme has fixed grammar (nodes); therefore it is easily and efficiently implementable by hardware or by software. If the bitwidths of the lookup tables in the decoder are not all equal (*asymmetric* decoder), there is an extra complication: one needs to define the fields using masks on the original voxel value. This is effectively a permutation of the labels of the subfields within the voxel (note that the masks can define subfields that overlap, which is very useful for luminance-alpha volumes).

It is easy to see that such a scheme can accommodate for many (if not most) cases of interest. Here we provide some examples:

- In the case of **rgba volumes**, each voxel is of the form (r, g, b, a). The lookup tables can be arbitrary functions, and all the $*$ operators can be $*_{\text{OR}} \equiv (\text{OR} \otimes \text{OR} \otimes \text{OR} \otimes \text{OR})$. Then

$$r \mapsto L_3(r) = (f_3(r), 0, 0, 0), \quad g \mapsto L_2(g) = (0, f_2(g), 0, 0),$$

and

$$L_3(r) *_{\text{OR}} L_2(g) \equiv L_{32}(r, g) = (f_3(r), f_2(g), 0, 0).$$

Similarly,

$$b \mapsto L_1(b) = (0, 0, f_1(b), 0), \quad a \mapsto L_0(a) = (0, 0, 0, f_0(a)),$$

and

$$L_1(b) *_{\text{OR}} L_0(a) \equiv L_{10}(b, a) = (0, 0, f_1(b), f_0(a)).$$

The final result is

$$L_{32} *_{\text{OR}} L_{10} = (f_3(r), f_2(g), f_1(b), f_0(a)).$$

- Similarly, if one of the voxel fields is a segmentation field, one can use the above strategy to modify the voxel's rgba assignment by a segmentation dependent rule. For concreteness, let us consider an example: Let (s, i) be a dual field, where s is a category index and i an intensity index. If

$$L_1(s = 1) = (1, 0, 0, 1), \; L_0(i) = (r, g, b, a)$$

and

$$*_{10} = (\text{AND} \otimes \text{AND} \otimes \text{AND} \otimes \text{OR}),$$

then

$$L_1(s=1) *_{10} L_0(i) = (r, 0, 0, 1).$$

The result is that the voxels that belong to the category $s = 1$ are made opaque, and only the red component of the voxel's classified intensity is kept. It is clear that one is limited only by one's imagination about the operators that one uses. In particular one can perform set-theory operations (such as find the union of all segments belonging to categories c_1, c_2, and make their opacity zero, i.e. make those two segments invisible.

- In the case of luminance-alpha volumes, let (l, a) be a general voxel. Then

$$L_1(l) = (r(l), g(l), b(l), 0),\ L_0(a) = (0, 0, 0, f(a))$$
$$L_1(l) *_{\text{OR}} L_0(a) = (r(l), g(l), b(l), f(a)).$$

The result is a pixel value, with extra bonus that one can apply a further lookup on the opacity values.

- As examples of the use of arithmetic operators, consider the following operators, which can be useful in combining two different scans of the same object together:

$$A *_c B \equiv \max\{A, B\},\quad A *_t B \equiv 1 - (1 - A)(1 - B)$$

One can use the $*_c$ operator for each color component, and the $*_t$ operator for the opacity:

$$rgba_1\ (*_c, *_c, *_c, *_t)\ rgba_2.$$

- One can also combine a volume with a shadow map of the same volume (representing the illumination of the volume by arbitrary light sources). The multi-field voxel is then (s, i), where s now stands for the shadow value, which is the opacity accumulated at the voxel position, when the volume is illuminated by the light sources. Simple shadowing can be performed by:

$$c' = (1 - s) * c$$

In terms of the decoder, consider the operation $*_{sh} = (\times, \times, \times, \times)$ (where $\times$ is the usual multiplication):

$$s \mapsto L_1(s) = (1-s, 1-s, 1-s, 1), \quad i \mapsto L_0(i) = (r, g, b, a),$$

and conclude that

$$L_1(s) *_{sh} L_0(i) = ((1-s)r, (1-s)g, (1-s)b, a).$$

- Further applications include depth shading (where one is given a depth field, which can be classified and combined with other subfields, thus simulating distance dependent fog), and stress/displacement fields (where one can modulate the sample's color by the local deformation stress of the volume).
- If one is also given the normal field magnitude data as an input to the decoder, then one can use it to further modulate the rgba above by the edgeness value. This concept can be easily generalized to providing for lighting of one or more classified fields, with subsequent combination with other classified subfield contributions.

We will not proceed further with exploring applications in the present paper.

3 Color weighted interpolation

For a given ray, one needs to partition it, and consider samples along the ray. In this section, we assume that we have somehow chosen the geometrical position of such a sample. Then there are two choices: Either interpolate (per field) all the voxel values in the neighborhood of the sample, and then apply the decoding process, or decode the voxels in that neighborhood, and interpolate the result. In either case, one would need to apply the front to back (FTB) blending equations, which we rewrite for convenience in terms of the transparency:

$$A' = A + a(1-A) \rightarrow T' = Tt, \tag{2}$$

$$C' = C + \tilde{c}T, \quad \tilde{c} \equiv c * a. \tag{3}$$

Equation (2) is a statistical assumption, namely that the transparency at each point defines a probability distribution, and that the distributions are independent for different sample points. If the neighborhoods that one uses around each sample point for interpolations are not disjoint, this assumption is not true, and one would have to take into account the correlation matrix of the distributions at consecutive sample points. In the case that one interpolates rgba values, it has been indicated in the literature [2] that one gets better images if one interpolates opacity-weighted colors, because one avoids the self-occlusion effect. Here follows a general proof of that.

Consider a ray r, transversing though a lattice grid. We are interested in accumulating color and opacity along this ray. This involves two steps: Choosing sample points along the ray, and for the purposes of this section we assume such a choice already given (the ramifications of such a choice will be discussed in a later section). In order to calculate the contribution of a sample to the blending equations, we need an interpolation procedure, which takes into account voxels in the neighborhood of the sample. Therefore, blending in D dimensions is best thought of as a D-dimensional integral. For any sample s along the ray, there corresponds a neighborhood $N(s)$, which will be used to deduce the pixel contributions of the sample to the blending equation. The motivation and physical interpretation of our approach is as follows: for each sample point, pick a voxel in its neighborhood (call it the *representative* of the sample). This choice has to obey the probability distribution of the interpolation measure in this neighborhood. For any such choice, calculate the accumulated color and opacity of the ray. Repeat the experiment many times, with different choices of representatives, each time calculating the accumulated color and opacity. Then, average the results, and define those averages to be the color and opacity accumulated along the ray. This process is very common in modern theoretical physics [14]. It has also been explored in a somewhat different context in [7]. Referring to fig.2, we show a ray, as it crosses six non-intersecting neighborhoods. For each of these neighborhoods, we have a well defined probability, as given by the (normalized) distance of each of the (two) voxels to the sample point (which is, in this case,the intersection of the ray with the neighborhood). Due to the statistical independence of the neighborhoods, the probability of a given choice of representatives for all the neighborhoods (choice function), is just the product of the individual probabilities.

We proceed with the formalism: we are given a ray r, and a collection

of uniformly spaced samples along it, enumerated by $s \in [0, S]$. For each sample s, we consider given a bounded interpolation neighborhood $N(s)$ [2], consisting of the voxels of which will be used to determine the pixel value at the sample position. Assume that

$$s \neq s' \Rightarrow N(s) \cap N(s') = \emptyset \ . \tag{4}$$

Assume a normalized measure of integration $d\mu_s$ on each neighborhood $N(s)$ given. This measure can be used to define interpolation averages on that neighborhood:

$$< f >_s \equiv \int_{N(s)} d\mu_s f. \tag{5}$$

Consider the space Φ_r of all *choice* functions $\phi : s \mapsto N(s)$. Clearly, for each such choice of function, we can define the quantities:

$$T_\phi(s) = \prod_{p=0}^{s} t(\phi(p)), \tag{6}$$

$$C_\phi(s) = \sum_{p=0}^{s} \tilde{c}(\phi(p)) T_\phi(s-1). \tag{7}$$

Those quantities involve only voxel positions, and they are therefore known a priori (without interpolation). Consider the measure induced on Φ_r by the product of the measures on each neighborhood:

$$d\mu_\Phi \equiv \prod_{i=0}^{s} d\mu_s. \tag{8}$$

Clearly, we should define the transparency and color accumulated along the ray as the Monte Carlo average (functional integral) over the space Φ_r, using the above measure.

$$T_s \equiv < T_\phi(s) >_{\phi \in \Phi_r} \ , \ \ C_s \equiv < C_\phi(s) >_{\phi \in \Phi_r} \ . \tag{9}$$

With the definitions above, we now proceed to prove that:

[2] This is an assumption, that a given interpolation scheme might, or might not, satisfy.

Theorem 1 *The color and transparency accumulated along the ray can be determined using the iteration equations:*

$$T_{s+1} = T_s < t >_{s+1}, \;\; C_{s+1} = C_s + < \tilde{c} >_{s+1} T_s \; . \tag{10}$$

Proof:
We first write the invariant definitions above in a more explicit way. $d\mu_s$ is a discrete probability measure, therefore:

$$< f >_s = \sum_{i \in N(s)} p_s(i) f(i) \equiv \sum_{\phi_s} p_s(\phi_s) f(\phi_s) \; , \;\; \sum_{i \in N(s)} p_s(i) \equiv \sum_{\phi_s} p_s(\phi_s) = 1. \tag{11}$$

In the above formula we introduced a shorthand notation for the choice function; for sample $k \in [0, s+1]$, let $\phi_k \in N(k)$ be its representative. Then it easily follows:

$$T_{s+1} = \sum_{\phi_0, ..., \phi_{s+1}} \prod_{k=0}^{s+1} p(\phi_k) \prod_{l=0}^{s+1} t(\phi_l) = T_s < t >_{s+1} \; . \tag{12}$$

Similarly

$$\begin{aligned} C_{s+1} &= \sum_{\phi_0, ..., \phi_{s+1}} \prod_{k=0}^{s+1} p(\phi_k) C_\phi(s+1) = \\ &= \sum_{\phi_0, ..., \phi_{s+1}} \prod_{k=0}^{s+1} p(\phi_k) (C_\phi(s) + \tilde{c}(\phi_{s+1}) T_\phi(s)) = \\ &= C_s + (\sum_{\phi_{s+1}} p(\phi_{s+1}) \tilde{c}(\phi_{s+1})) T_s = \\ &= C_s + < \tilde{c} >_{s+1} T_s. \end{aligned} \tag{13}$$

This concludes the proof. Equation (10) above proves that in the case of classification before interpolation one should interpolate $< c * a >$ per sample point, as has been speculated in the past. The above arguments are independent of the interpolation scheme, as long as the interpolation neighborhoods around the samples do not overlap. In the general case, the argument above is only approximate. It is intuitively clear however that interpolating $\tilde{c}$ is much more mathematically sound than interpolating c. Furthermore, we note that interpolating c, is equivalent to saying that

$$< c * a > = < c > * < a > \; .$$

This is clearly incorrect, because the color components and the opacity are derived by applying transfer functions (or a general decoder) on the same voxel value spatial distribution, which means that in general they are statistically correlated.

4 Normal Weighted Modulation (NWM) of color (lighting) and opacity

Given the decoder's output [3], one can apply one's favorite lighting equation, and also perform edge-weighted modulation of the opacity. In this section we address the issue of how to modify the lighting equation, under the assumption that one knows the interpolated opacity weighted color, and the interpolated opacity. Therefore it is of interest to rewrite the lighting and edge-weighting equations using only those variables.

In general we consider functions of the form $G(x, b)$, where x is a positive number and b is a boolean variable.

$$G(x, \text{true}) = \text{const} \times x, \quad G(x, \text{false}) = 1.$$

We define, as usual, the normal vector using the gradient of one of the input intensities (or general fields, such as depth scalar field):

$$\vec{n} = -\nabla v \ , \quad n = |\vec{n}| \ , \quad \hat{n} \equiv \vec{n}/n.$$

Then, the appropriate equations are:

$$< a >_s^{NWM} = G_1(n_s, b) < a >_s, \tag{14}$$

$$\begin{aligned} < \tilde{c} >_s^{NWM} \quad = \quad & (\kappa_{emiss} G_2(n_s, b') + \kappa_{diff} I_d[\hat{n}_s] G_3(n_s, b'')) < \tilde{c} >_s + \\ & \kappa_{spec} I_s[\hat{n}_s] G_4(n_s, b''') c_{spec} < a >_s \ . \end{aligned} \tag{15}$$

The equation above was derived by multiplying the usual expression for illumination of color components by equation (14). This multiplication is to be understood to take place before the interpolation, as explained in the

[3] In general, one could also embed the NWM unit within the decoder, which would allow for combining lit pixel values, according to logical or arithmetic operations.

previous section. Note that in equation (15) we allowed for edge-weighting the terms individually, allowing only terms linear in the length of the voxel value gradient. The physical assumption here is that the emissive, diffuse and specular components of the equation can be individually affected by the local edgeness.

5 General integration along rays

There are two "canonical" ways of doing raytracing. Either *always* (independent of the view angle) have fixed step per ray, or not. The second case arises when one changes the sampling frequency along the ray, or in (orthographic, for simplicity) shear-warp ray tracing, where the distance between samples is view-dependent. If one does not compensate for the non-constancy of the distance between samples in the situations above, one would see that the the color of the rendered image changes. The solution of this problem usually goes by the name of alpha correction. Here the issue is revisited, in light of the $< c * a >$ interpolation.

The solution of the usual transparency accumulation equation is

$$T(n) = \prod_{i=1}^{n} t(i). \tag{16}$$

Therefore

$$T(n) = \exp \sum_{i=1}^{n} \ln t(i). \tag{17}$$

The argument simplifies in the continuum limit, so we rewrite the equation as an integral along a ray r, parametrized by τ (where the "eye" is positioned at $\tau = 0$).

$$T(\tau) = \exp \int_0^{\tau} d\tau' \ln t(\tau'). \tag{18}$$

The integral in the equation above is to be understood as a shorthand for the discrete sum. If taken seriously as an integral, then it is clear that it is equivalent to the usual integral only in first order (with respect to the opacity distribution). This is not surprising, because the transition from a discrete equation to a continuum one is not uniquely defined. Equation (18) has the property that if

$$r \ni \tau_0 = 0 \rightarrow T(\tau_0) = 0.$$

This is in sharp contrast to the usual continuum limit, where

$$T(\tau_0) = \exp\,(-\int_0^{\tau_0} d\tau\; a(\tau)), \tag{19}$$

where the transparency does not accumulate to zero, even if all the opacities (absorption coefficients) along the ray are equal to one! Therefore (18) should be preferred over (19).

Clearly, when one changes the parametrization along the ray, one should get the same result, because the physics of the problem should not depend on the parametrization along the ray. Let $\tau' = \tau'(\tau)$ be a change of variables in the integral. Then

$$T = \exp \int_r d\tau' \frac{d\tau}{d\tau'} \ln t(\tau(\tau')) = \exp \int_r d\tau' \ln(t(\tau')^{\frac{d\tau}{d\tau'}}). \tag{20}$$

We see that reparametrization invariance of the accumulated transparency results in modifying the transparency locally, by exponentiation according to the local change of scale. This is the generalization of the known result, where $\tau \mapsto \tau' \equiv \tau/m$, with m being a constant (which depends on the view angle (for shear-warp) and the supersampling (along the ray) ratio). Then one need to apply the alpha correction on the transparency of the sample.

$$< t >_s \mapsto < t >_s^m, \;\; < a >_s \mapsto f_m(< a >_s) \equiv 1 - (1 - < a >_s)^m. \tag{21}$$

In the above equation we alpha-correct the opacity derived from the interpolated voxel value (as opposed to interpolating the alpha-corrected opacities for each voxel in the neighborhood of the sample). We do this is because the alpha correction arises from the change measure of integration along the ray; therefore one has to assume that the sample values are given and then alpha correct them.

The alpha correction equation has interesting convexity properties. Consider a ray segment r, of length l, and denote by $<>_r$ averages of functions defined on the ray. Then, using eq. (17), one can prove that

$$T(r) = \exp \sum_i \ln t_i = \exp\,(< \ln t >_r\, l) \leq \exp\,(\ln < t >_r\, l) =< t >_r^l \;. \tag{22}$$

In the above equation we used Jensen's inequality, which is a fancy way of saying that the logarithm is a convex down function. Therefore we proved that:

Theorem 2 *The transparency accumulated on a ray segment is less than or equal to the transparency that would be accumulated on the ray segment if it was uniform (with transparency equal to the average transparency of the given distribution).*

This observation is fundamental because any continuum integral is defined in terms of a partition of its domain, in the limit that the mesh of the partition goes to zero. We see that as we make the mesh finer (by adding points to the current partition of the ray), the argument above shows that the value of the accumulated transparency (under the assumption that each bin is uniform, with value equal to its true average) becomes less and less, and this process converges by definition to the transparency accumulated on the original continuum ray.

As an application of this, we consider the effect of supersampling along a ray. Consider a cell defined by two neighboring voxels (assume a one dimensional ray, for simplicity), with transparencies t_1, t_2. Further consider N samples between them, such that each bin of the partition has equal length. Let $T_{\text{cell}}(N)$ be the contribution of those N samples to the transparency accumulation equation, using the appropriate alpha correction per each interpolated sample. Then

$$T_{\text{cell}}(N) = \prod_{k=1}^{N} (\frac{k}{N+1} t_1 + (1 - \frac{k}{N+1}) t_2)^{1/N} \equiv \prod_{k=1}^{N} t(k,N)^{1/N}. \tag{23}$$

Then it follows that:

$$0 \leq \lim_{N \to \infty} T(N) \leq \ldots \leq T(N+1) \leq T(N) \leq \ldots \leq T(2) \leq T(1).$$

A point worth mentioning is that the average transparency of those samples is equal to the average of the transparencies of the end points of the cell:

$$\frac{1}{N} \sum_{k=1}^{N} t(k,N) = \frac{t_1 + t_2}{2}, \;\; \forall N \geq 1. \tag{24}$$

Numerical simulations indicate that the convergence is pretty fast. In particular, for all values of t_1, t_2 one obtains the following rough upper bounds:

$$\frac{T(1)}{T(2)} \leq 1.06, \; \frac{T(2)}{T(3)} \leq 1.03, \; \frac{T(3)}{T(4)} \leq 1.02, \frac{T(4)}{T(5)} \leq 1.01, \; \frac{T(12)}{T(13)} \leq 1.004.$$

Those bounds are pessimistic, and they are approached as $|t_1 - t_2| \to 1$. The philosophical statement that follows from this is that *one needs to supersample in such a way that consecutive samples have transparency difference smaller than a set threshold.* [4] This supersampling can be adaptive. It is to be emphasized that the above theorem (and numbers) applies (per cell) only in cases where (24) is satisfied. This is not necessarily strictly true for all interpolation schemes.

The color of the sample is not affected by this (local) change of scale, as is well known (and can be easily proved). The philosophy behind the alpha correction is that the ray segment stretches, therefore the "density" of the material there changes; however the material itself does not get affected.

Interpolating opacity weighted colors introduces a minor complication, which we now turn to. One can define as the *effective* sample's color

$$< c >_s^{eff} \equiv \begin{cases} \frac{<c*a>_s}{<a>_s} & \text{if } < a >_s \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{25}$$

In this expression it is clear that the self occlusion effect is avoided, because if the sample is transparent, its effective color (as defined above) also vanishes. It is known that the sample's color is not affected by changing the scale (this holds true even if the change of scale is local): $< c >_s^{eff} \mapsto < c >_s^{eff}$. Therefore

$$< c*a >_s = < c >_s^{eff} * < a >_s \mapsto < c >_s^{eff} * f_m(< a >_s) = < c*a >_s * \frac{f_m(< a >_s)}{< a >_s}.$$

We define for convenience

$$g_m(< a >_s) \equiv \frac{f_m(< a >_s)}{< a >_s}, \tag{26}$$

and we have proved that the sample's opacity weighted color needs to be rescaled according to

$$< \tilde{c} >_s \mapsto < \tilde{c} >_s \; g_m(< a >_s), \tag{27}$$

under a (potentially sample dependent) change of scale (by m) along the ray. Note that the function $g_m(x)$ has a well defined limit at $x \to 0$. This is the

[4] It is the author's opinion that this threshold can be as high as 0.2

alpha correction rule for the opacity weighted colors. Note that in the case where the voxels at the boundary of the cell are equal, the equations reduce to the usual ones.

Using the parametrization of samples within a cell as above, it is clear that considering N samples (using linear interpolation) within a (one dimensional) cell, one gets the following:

$$C_{\text{cell}}(N) = \sum_{k=1}^{N} \tilde{c}(k,N) g_m(a(k,N)) \prod_{l=1}^{k-1} t(l,N)^{1/N}. \tag{28}$$

In concluding this section, we would like to make a couple of points. First, in single-field voxel visualization one traditionally absorbs the change of scale in the lookup tables. In the multi-field visualization case, it would be incorrect in general to modify all the corresponding lookup tables and use the same operators to combine them. The philosophical statement here is that the decoder has the responsibility of determining a sample's rgba. This process is interpolation and classification (in either order), and it doesn't have anything to do with integration along the ray. Therefore, for multi-field visualization, it is imperative to use the alpha correction as part of the compositing unit. A similar argument applies to the calculation of the local normal. Such a calculation could be performed on the opacity value, or on the voxel value. This is again a local (derivative) operation that does not have anything to do with integration along the ray. Therefore, if one wanted to calculate normals as gradients of opacity, one would have to use the original (non alpha corrected) opacity.

Second, the convexity argument of eq. (22) applies to the case of classification before interpolation. In the case of classification after interpolation (i.e. we find the voxel value per sample, by interpolating each of the fields of the voxels in the neighborhood, and then we apply the decoder on that voxel value) one cannot make a general statement. However, in the case where the transparency transfer function is convex-down (at least in the range of the voxel values that are relevant in the neighborhood of the samples of interest along the ray), then theorem 2 still applies. The reason is that the logarithm is a convex-down function, and the composition of two convex-down functions is convex-down. For illustration of the argument, consider samples of voxel values v_1, v_2, and the sample of voxel value $\frac{1}{2}(v_1 + v_2)$, which one would use if one ray traced along that ray segment with a subsampling factor of

two. Then:

$$\frac{1}{2}(\ln t(v_1) + \ln t(v_2)) \leq \ln \frac{t(v_1) + t(v_2)}{2} \leq \ln(t(\frac{v_1 + v_2}{2})).$$

In the equation above, the first inequality comes from the fact that the logarithm is convex-down, and the second inequality comes from the *assumption* that the transparency transfer function is convex-down. Exponentiating the above, one indeed obtains that

$$t(v_1)^{1/2} t(v_2)^{1/2} \leq t(\frac{v_1 + v_2}{2}),$$

as advertised above. The philosophical statement is that *when one interpolates after classification, one is guaranteed to obtain a smoother image. If one interpolates before classification, one might get artifacts that are due to the non-convexity of the transfer functions.* As a numerical example, consider the transition (at the boundaries of an object) between empty space (voxel value $v_{em} = 0$), to a fixed value, say $v_{obj} = 100$. Say that $L(v_{obj}) = (1, 0, 0, 1)$, $L(v_{em}) = (0, 0, 0, 0)$, $L((v_{obj} + v_{em})/2) = (0, 1, 0, 1)$. If one interpolates after classification, one would obtain a pale red at the boundaries, whereas if one interpolates before classification one would obtain green color (artifact) at the boundary. In general, it is much safer to interpolate pixel values after classification. The classification after interpolation path imposes non-local constraints, i.e. the pixel value of the sample is determined only by the average voxel value in the sample's neighborhood, independent of the variance of the voxel value there. For example, neighborhood voxel values $\{0, 240\}$ and $\{118, 122\}$ would be mapped to the same pixel value (if one classified after interpolation), which is usually unreasonable. For the same reason, the classification after interpolation path is more sensitive to noise.

6 Supersampling on image plane

In a previous section we investigated how the line integral changes if one changes the unit of length along a given ray. However, ultimately one has to assign values to a pixel on an image. This is the issue of how one samples the image plane, which is equivalent to a (local) choice of lengths along the

image plane. In the discrete limit, one can use one or more rays to calculate a pixel (and this can be done either explicitly, or implicitly by creating a scaled texture and put the pixels there). Consider a ray r. For much of the discussion the ray can be an arbitrary curve in space $\vec{x}_r(s)$, parametrized by its natural length. Here we will only deal with the orthographic case, where the ray is a straight line, and all rays are parallel to each other. Given a displacement vector $\vec{w}$ on the image plane, one can consider the ray $r + \vec{w}$, which is simply the translation of all points of the initial master ray (parametrized as $\vec{x}_r(s)$, s being the local length) by $\vec{w}$ (see fig. 4). In the orthographic case we take

$$\frac{\partial \vec{w}}{\partial s} = 0.$$

Let $T(r + \vec{w})$ denote the transparency accumulated along that ray, as usual. Let $N(r)$ be a tubular neighborhood of the ray r, meaning a collection of $\vec{w}$s, which define an open set on the image plane. We introduce the quantity

$$T(N(r)) \equiv \int d^2 w p(\vec{w}) T(r + \vec{w}),$$

which represents an interpolation of all the transparency contributions of all the rays in the tubular neighborhood, according to probability distribution p. The problem at hand is to find an analytic expression for this quantity. The mental image that we have in mind is a process where we take the limit that the tubular neighborhood shrinks continuously to the ray. This limit will be a line integral on the original ray r, and it is of interest to know how is this limit related to the original line integral. Let the probability distribution be a 2d Gaussian filter:

$$p(\vec{w}) \equiv \frac{1}{2\pi\sigma^2} e^{-\frac{w^t w}{2\sigma^2}}.$$

Furthermore, from (18) [5]

$$T(r + \vec{w}) = e^{\int_{r+\vec{w}} \ln t(\vec{x}_r(s)+\vec{w}) ds}.$$

We now use the Taylor expansion of the transparency function, to second order in $\vec{w}$.

$$t(\vec{x}_r(s) + \vec{w}) = t(\vec{x}_r(s)) + w^t \nabla t(\vec{x}_r(s)) + \frac{1}{2} w^t H(\vec{x}_r(s)) w,$$

[5] One could of course use the more traditional (19), and make similar arguments. Or one can power expand the result that follows to linear order in the opacity, using that $\ln(1 - a) \approx -a$.

where H is the Hessian 3x3 matrix $H_{ij} = \partial_i \partial_j t$. Collecting terms, expanding the logarithm to second order, and performing the Gaussian integration (assuming that the $\vec{w}$s span a neighborhood of radius at least 3σ, in which case we approximate the bounded integral with an infinite one), one obtains that

$$T(N(r)) = T(r)(\det(I - \sigma^2 < M >_r))^{-1/2} e^{\frac{1}{2} < \nabla \ln t >_r^t (\frac{1}{\sigma^2} I - < M >_r)^{-1} < \nabla \ln t >_r}. \tag{29}$$

In the above formula we use the definition

$$M_{ij}(\vec{x}_r(s)) \equiv \partial_i \partial_j \ln t|_{\vec{x}_r(s)},$$

and also $< M >$ denotes a 3x3 matrix, the elements of which are the integrals (along the ray) of the corresponding elements of the matrix at each point on the ray. Similarly for the average of the gradient vector.

Equation (29) is a remarkable result. It explicitly collapses the accumulated opacity contributions of a tubular neighborhood of a given ray to a line integral along the ray. The new result is proportional to the old one, and the multiplicative modification depends on "second derivatives" along the ray. Therefore one need not explicitly calculate the alpha contributions of a family of sub-pixel positioned rays, and interpolate the result; one can automatically calculate the result, if one chooses to accumulate the first and second derivative expressions that appear above, along the fixed one ray.

As a sanity check, it is clear that if the transparency values transversely to the ray are uniform, then one should obtain the same result as with only the original ray. Indeed, (29) implies:

$$\Big(\int ds \nabla t(\vec{x}_r(s)) = 0_{3\text{x}1} \Big) \wedge \Big(\int ds M(\vec{x}_r(s)) = 0_{3\text{x}3} \Big) \Rightarrow T(N(r)) = T(r).$$

The technique above can be applied in deriving the effective color of a neighborhood of rays. The result is not very illuminating, and therefore omitted.

7 Conclusions

In this paper we presented a rigorous discussion of the mathematics of ray tracing. Our focus was multi-modal and multi-resolution invariant visualization. The concept of a general voxel decoder was introduced, which accommodates most if not all applications that are currently considered as important.

We introduced the concept of functional integration, as the unifying principle between sample classification, interpolation and blending. Using this, we proved in general that one should interpolate opacity weighted colors. The lighting equation was rewritten as to use the opacity-weighted colors. When multi-resolution considerations are important (as is the case in shear-warp raytracing and in supersampling along the rays) we proved that one should correct (renormalize) the interpolated opacity and the interpolated opacity-weighted colors (as opposed to interpolating the renormalized opacities and opacity-weighted colors), and we provided the necessary equations. Finally, we considered supersampling along the image plane (in orthographic), and we proved that the effect of filtering the supersampled image is equivalent to a modified blending equation, which also takes into account the accumulation of auxiliary tensor fields.

8 Acknowledgments

I would like to thank my colleagues in the Volume Graphics Group of Mitsubishi Electric for creating a very stimulating environment. In particular the present work was influenced by stimulating discussions with Dr. L. Seiler on the topic of multi-scale interpolation. Also, thanks to Vikram Simha for introducing the concept of cascaded lookup for dual modality visualization to me, and to to Dr. T.C. Zhao for providing valuable comments, and for reading the manuscript from a fellow physicist's perspective.

References

[1] T. Porter, T. Duff, "Compositing Digital Images", Computer Graphics, 18(3), pg 253-259, July 1984

[2] T. Malzbender, C. Wittenbrink, and M. Goss, "Opacity-Weighted Color Interpolation for Volume Sampling", HPLabs Technical Report, (HPL-97-31), February 1997

[3] P. Lacroute and M. Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation", Computer Graphics, 28(4), 451-458, Aug. 1994.

[4] K. Zuiderveld, "Visualization of Multimodality Medical Volume data using Object-Oriented Methods", Ph.D Thesis.

[5] G. Kindlemann, J. Durkin, "Semi-Automatic Generation of Transfer Functions for Direct Volume Rendering", 1998 Symposium on Volume Visualization, ACM SIGGRAPH, pg. 79-86.

[6] S. Gibson "Using Distance Maps for Accurate Surface Representation in Sampled Volumes", 1998 Symposium on Volume Visualization, ACM SIGGRAPH, pg 23-30.

[7] J.T. Kajiya "The Rendering Equation", SIGGRAPH '86 Conference Proceedings, 20(4), pg 143-149.

[8] B.T. Phong, "Illumination for computer generated pictures", Communications of the ACM 18(6), pg. 49-59

[9] M. Levoy, "Rendering of surfaces from volume data", IEEE Computer Graphics and Applications 8(3), pg. 28-37

[10] J.D. Foley, A. van Dam, K. Feiner and J.F. Hughes, "Computer Graphics-Principles and Practice", second edition, Addison-Wesley, Reading, Massachusetts.

[11] M. Levoy, "A hybrid ray tracer for rendering polygon and volume data", IEEE Computer Graphics and Applications 10(3), pg. 33-40.

[12] U. Behrens, R. Ratering, "Adding Shadows to a Texture-Based Volume Renderer", 1998 Symposium on Volume Visualization, ACM SIGGRAPH, pg 39-46.

[13] B. Lichtenbelt, R. Crane, S. Naqvi "Introduction to Volume Rendering", Hewlett-Packard Professional Books, Prentice-Hall.

[14] J.W. Negele, H. Orland "Quantum Many-Particle Systems", Frontiers in Physics Lecture Note Series, Addison Wesley

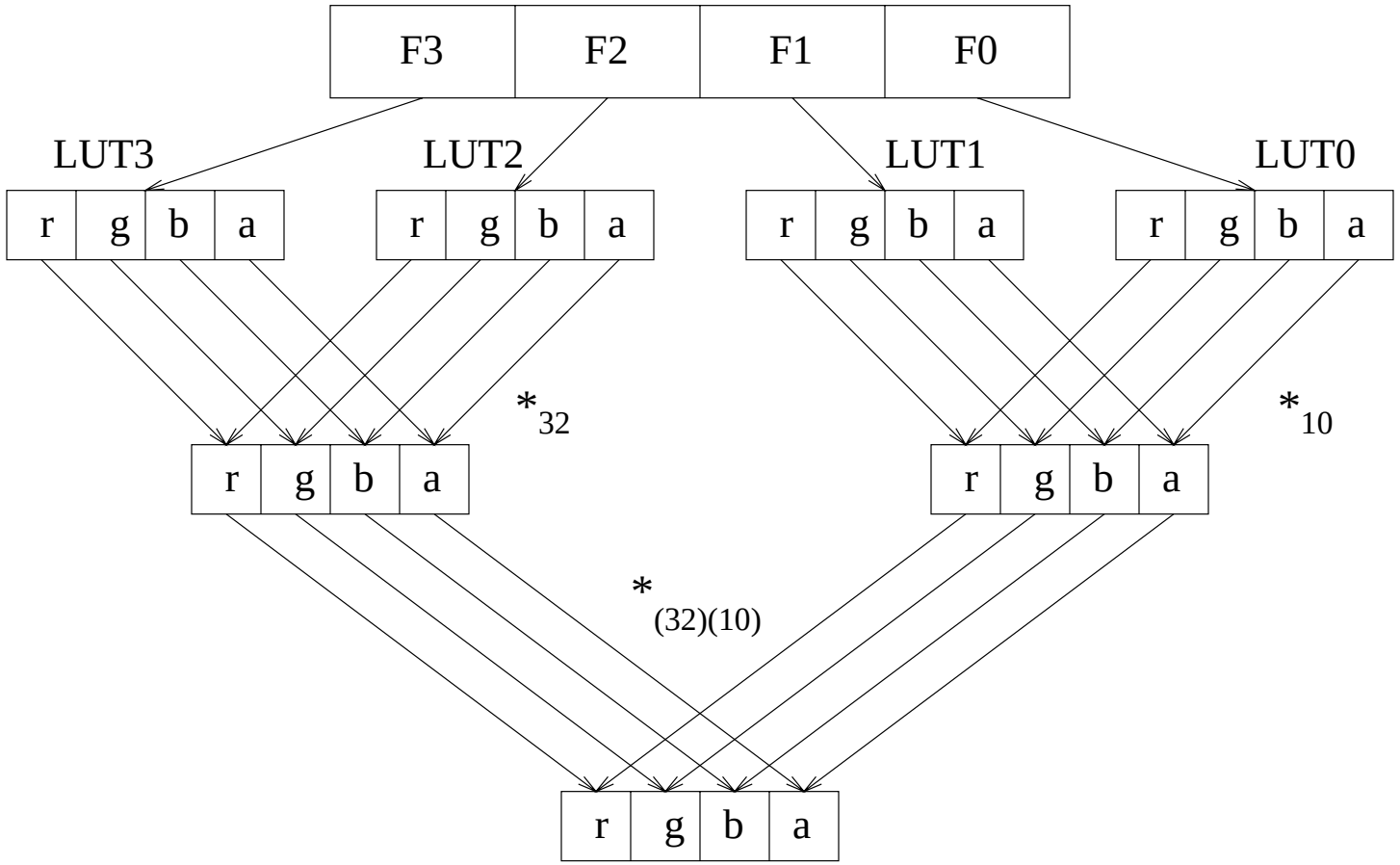

Fig. 1. Fixed-grammar variable width cascaded decoder.

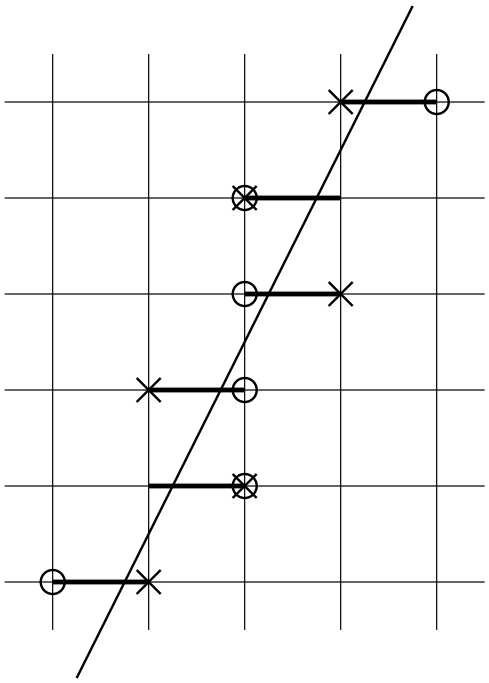

Fig. 2. Non-intersecting interpolation neighborhoods along a ray, and two different paths (choice functions). The probability of each path is the product of the probabilities for the choice of representative (denoted with circles for one path and crosses for the other), for all the neighborhoods.

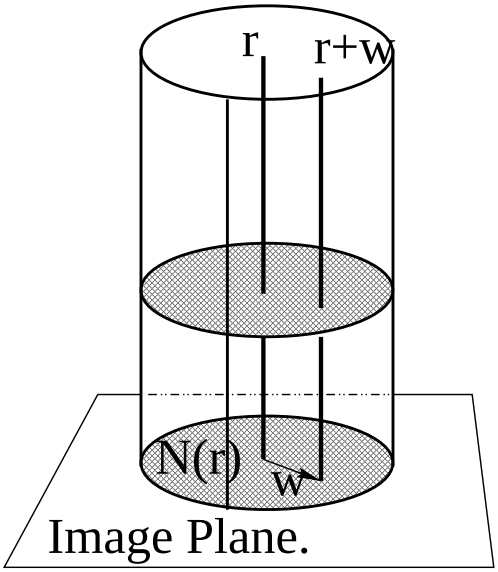

Fig. 3. Bundle of rays in the neighborhood N(r) of ray r are filtered according to p(w).