Interpreting OpenEXR Deep Pixels

Florian Kainz, Industrial Light & Magic, September 24, 2013

Contents

Overview

Starting with version 2.0, the OpenEXR image file format supports *deep images*. In a regular, or *flat* image, every pixel stores at most one value per channel (OpenEXR supports sub-sampled channels where values are stored only at a subset of all pixels). In contrast, each pixel in a deep image can store an arbitrary number of values or *samples* per channel. Each of those samples is associated with a *depth*, or distance from the viewer. Together with the two-dimensional pixel raster, the samples at different depths form a three-dimensional data set.

The open-source OpenEXR file I/O library defines the file format for deep images, and it provides convenient methods for reading and writing deep image files. However, the library and the existing documentation (as of September 2013) do not explain in detail how deep images are meant to be interpreted.

This document attempts to describe what the deep data in a file mean and how compositing of deep images works. The document also points out numerical issues with the representation of volumetric samples in OpenEXR 2.0, as well as a possible way to address those issues.

Definitions

A pixel at location (x, y) in a deep image has $n(x, y)$ *samples*. The number of samples varies from pixel to pixel. Any non-negative number of samples, including zero is allowed.

Every channel of a given pixel has one stored value per sample. The number of stored values is the same for all channels in the pixel. The stored values in each channel are numbered from 0 to $n(x, y) - 1$.

In the following we will discuss a single pixel; for readability we will omit the coordinates of the pixel: expressions such as n and $S_i(c)$ are to be understood as $n(x, y)$ and $S_i(c, x, y)$ respectively.

Every deep image has a Z channel and optionally a $ZBack$ channel. All samples in the Zchannel must be greater than or equal to zero, and the Z values must be monotonically increasing:

$$S_i(Z) \geq 0,$$

$$S_i(Z) \leq S_{i+1}(Z)$$

If a deep image has no $ZBack$ channel, then a $ZBack$ channel can be formed by simply copying the Z channel:

$$S_i(ZBack) = S_i(Z)$$

The samples in the Z and $ZBack$ channels assign a depth range to the corresponding samples in the other channels of the pixel. For any channel c, except Z and $ZBack$, $S_i(c)$ covers the depth interval

$$\left[S_i(Z), \quad \max\left(S_i(Z), S_i(ZBack)\right)\right[.$$

In other words, if $Z \geq ZBack$, then a single depth value, Z, is assigned to the corresponding samples in the other channels, but if $Z < ZBack$, then the depth range $[Z, ZBack[$ is assigned to the samples in the other channels. A sample with a single depth value is a *point sample*; a sample with a depth range is a *volumetric sample*.

Volumetric samples must not overlap with other samples; $S_i(ZBack)$ must be less than or equal to $S_{i+1}(Z)$ for $0 \leq i < n - 1$.

Alpha as a Function of Depth

Every deep OpenEXR image must contain either a single alpha channel, A, or three alpha channels, RA, GA and BA.

The samples in an alpha channel, α, define a function of depth, $z \mapsto \alpha(z)$, where $\alpha(z)$ represents the cumulative opacity of all objects between the viewer and a point at depth z. The function $\alpha(z)$ for a pixel with n samples consists of a series of $n + 1$ segments, where each segment is either constant, an exponential curve, or a combination of an exponential curve and a constant part. The segments, called

$$\alpha_{-1}(z), \alpha_0(z), \alpha_1(z), \ldots\ \alpha_{n-1}(z),$$

are defined as follows:

$$\alpha_i(z) = \begin{cases} 0, & i = -1 \\ f + b \cdot (1 - f), & i \geq 0 \end{cases}$$

where

$$f = \alpha_{i-1}\big(S_i(Z)\big)$$

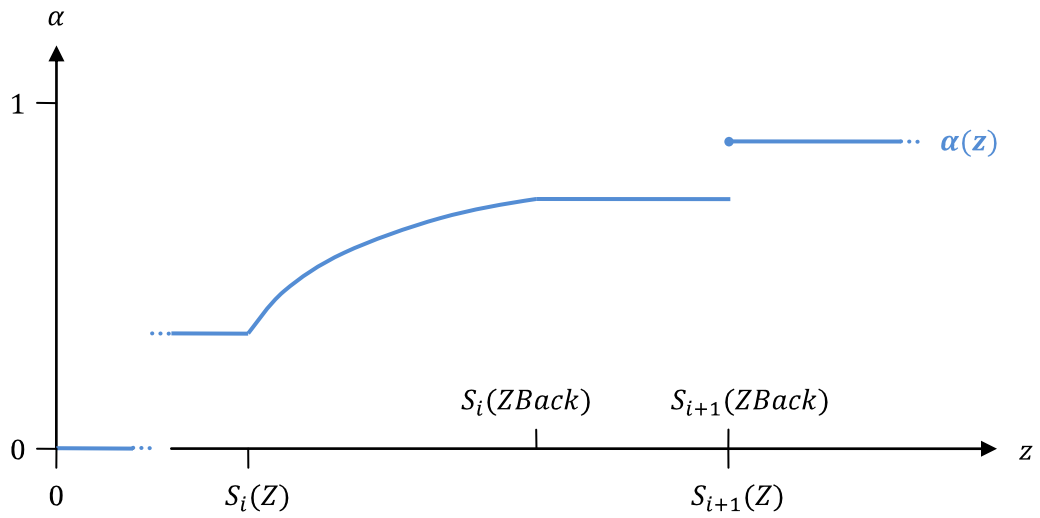
and

$$b = \begin{cases} S_i(\alpha), & z \geq\ S_i(ZBack) \\ 1 - \big(1 - S_i(\alpha)\big)^{\frac{z - S_i(Z)}{S_i(ZBack) - S_i(Z)}}, & z < S_i(ZBack) \end{cases}$$

With this definition of the segments, the overall function $\alpha(z)$ looks like this:

$$\alpha(z) = \begin{cases} \alpha_{-1}(z), & z < S_0(Z) \\ \alpha_i(z), & S_i(Z) \leq z < S_{i+1}(Z) \\ \alpha_{n-1}(z), & S_{n-1}(Z) \leq z \end{cases}$$

The figure below shows an example of alpha as a function of depth. Sample number i is volumetric; its $ZBack$ is greater than its Z. Alpha increases gradually between Z and $ZBack$ and then remains constant. Sample number $i + 1$, whose Z and $ZBack$ are equal, is a point sample where alpha discontinuously jumps to a new value.



Color as a Function of Depth

Any layer in a deep OpenEXR image may contain a set of RGB color channels. For each set of color channels the R, G and B samples define corresponding functions of depth, $z \mapsto r(z)$, $z \mapsto g(z)$ and $z \mapsto b(z)$, where the triple $(r(z), g(z), b(z))$ represents the cumulative pre-multiplied color of all objects between the camera and a point at depth z.

The function $z \mapsto c(z)$ for a color channel c with n samples consists of a series of $n+1$ segments, where each segment is either constant, an exponential curve, or a combination of an exponential curve and a constant part. The definition of the segments requires an alpha channel α. If a deep image has only an A channel, then A is used as the alpha channel for R, G and B. Otherwise, RA, GA and BA are used as the alpha channels for R, G and B respectively. The segments, called

$$c_{-1}(z), c_0(z), c_1(z), \ldots\ c_{n-1}(z)$$

are defined as follows:

$$c_i(z) = \begin{cases} 0, & i = -1 \\ f + b \cdot \left(1 - \alpha_{i-1}\left(S_i(Z)\right)\right), & i \geq 0 \end{cases}$$

where

$$f = c_{i-1}\left(S_i(Z)\right)$$

and

$$b = \begin{cases} S_i(c), & z \geq\ S_i(ZBack), \quad S_i(\alpha) > 0 \\ S_i(c) \cdot \dfrac{1 - \left(1 - S_i(\alpha)\right)^{\frac{z - S_i(Z)}{S_i(ZBack) - S_i(Z)}}}{S_i(\alpha)}, & z < S_i(ZBack), \quad S_i(\alpha) > 0 \\ S_i(c) \cdot \dfrac{z - S_i(Z)}{S_i(ZBack) - S_i(Z)}, & z < S_i(ZBack), \quad S_i(\alpha) = 0 \end{cases}$$

With this definition of the segments, the overall function $c(z)$ looks like this:

$$c(z) = \begin{cases} c_{-1}(z), & z < S_0(Z) \\ c_i(z), & S_i(Z) \leq z < S_{i+1}(Z) \\ c_{n-1}(z), & S_{n-1}(Z) \leq z \end{cases}$$

Note that in the definition of b, above, the third case is the limit of the second case as $S_i(\alpha)$ approaches zero.

Image Compositing

One of the most important uses of deep images is *deep compositing*. For flat images — without deep pixels — the most common compositing operation is called $over$. Given two RGBA images, f and b, the $over$ operation creates a composite image c by merging f and b according to the A channel of f. In the composite image, f is visible wherever f is completely opaque ($A = 1$), b is visible where f is completely transparent ($A = 0$), and a mixture of f and b is visible where f is semi-transparent ($0 < A < 1$). For pre-multiplied RGBA pixels, the operation

$$c = over\ (f, b)$$

works like this:

$$R_c = R_f + R_b \cdot (1 - A_f)$$
$$G_c = G_f + G_b \cdot (1 - A_f)$$
$$B_c = B_f + B_b \cdot (1 - A_f)$$
$$A_c = A_f + A_b \cdot (1 - A_f)$$

For deep images, instead of the $over$ operation we want two operations:

- $merge$ combines two deep images and forms a new deep image by interlacing the samples from the original images.
- $flatten$ performs a series of $over$ operations on the samples in a single deep image and produces a new flat image.

Flatten

Flattening an image is simple; repeatedly replace samples 0 and 1 with the result of

$$\text{sample } 0\ over\ \text{sample } 1$$

until only one sample is left. For any channel c in the image, the final value in the flattened image will be $c\big(S_{n-1}(ZBack)\big)$.

Merge

The $merge$ operation is more complicated. For a given pixel, it works as follows:

1) Form a new list of samples by combining the lists of samples from the two input pixels.

2) Sort the new list according to the Z coordinates of the samples. If two samples have the same Z, then sort according to $ZBack$. If two samples have the same Z and $ZBack$, then their order does not matter.

3) Remove overlapping samples:

If two point samples, numbered i and $i+1$, have the same Z coordinate, that is, $S_i(Z) = S_{i+1}(Z)$, then replace them with a new single sample, i.

For an alpha channel α,

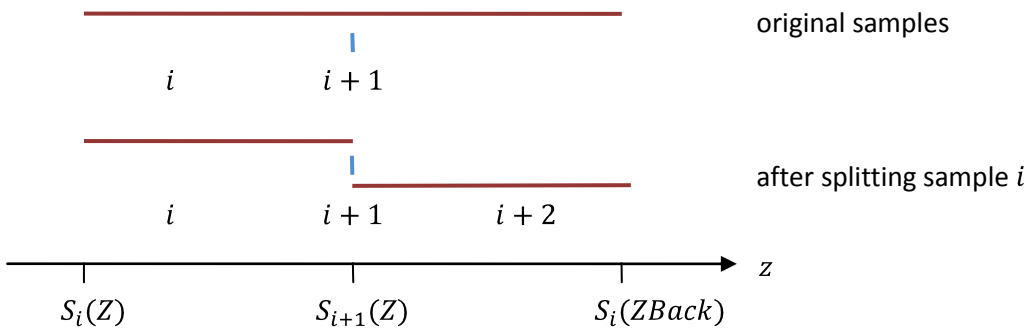
$$S_{i,new}(\alpha) = S_i(\alpha) + S_{i+1}(\alpha) \cdot \big(1 - S_i(\alpha)\big).$$

For a color channel c,

$$S_{i,new}(c) = S_i(c) \cdot \left(1 - \frac{S_{i+1}(\alpha)}{2}\right) + S_{i+1}(c) \cdot \left(1 - \frac{S_i(\alpha)}{2}\right).$$

Note that the value of the new sample $S_{i,new}(c)$ is the average of the results of compositing sample i over sample $i+1$ and compositing sample $i+1$ over sample i.

If a point sample is inside a volumetric sample, then split the volumetric sample. With the original volumetric sample covering the interval $[S_i(Z),\ S_i(ZBack)[$, and the point sample located at $S_{i+1}(Z)$, replace the volumetric sample with two new samples, numbered i and $i+2$, that cover the intervals $[S_i(Z),\ S_{i+1}(Z)[$ and $[S_{i+1}(Z),\ S_i(ZBack)[$ respectively.



For an alpha channel α,

$$S_{i,new}(\alpha) = 1 - \big(1 - S_i(\alpha)\big)^{\frac{S_{i+1}(Z) - S_i(Z)}{S_i(ZBack) - S_i(Z)}}$$

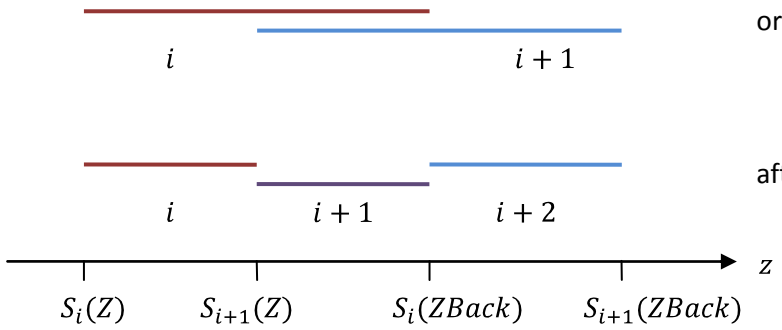
$$S_{i+2,new}(\alpha) = 1 - \big(1 - S_i(\alpha)\big)^{\frac{S_i(ZBack) - S_{i+1}(Z)}{S_i(ZBack) - S_i(Z)}}$$

For a color channel c,

$$S_{i,new}(c) = \begin{cases} S_i(c) \cdot \dfrac{S_{i,new}(\alpha)}{S_i(\alpha)}, & S_i(\alpha) > 0 \\ S_i(c) \cdot \dfrac{S_{i+1}(Z) - S_i(Z)}{S_i(ZBack) - S_i(Z)}, & S_i(\alpha) = 0 \end{cases}$$

$$S_{i+2,new}(c) = \begin{cases} S_i(c) \cdot \dfrac{S_{i+2,new}(\alpha)}{S_i(\alpha)}, & S_i(\alpha) > 0 \\ S_i(c) \cdot \dfrac{S_i(ZBack) - S_{i+1}(Z)}{S_i(ZBack) - S_i(Z)}, & S_i(\alpha) = 0 \end{cases}$$

If two volumetric samples overlap, but neither sample is completely contained inside the other, then create three non-overlapping samples. Replace the original samples, which cover the intervals $[S_i(Z),\ S_i(ZBack)[$ and $[S_{i+1}(Z),\ S_{i+1}(ZBack)[$, with three new samples, numbered i, $i+1$ and $i+2$, that cover the intervals $[S_i(Z),\ S_{i+1}(Z)[$, $[S_{i+1}(Z),\ S_i(ZBack)[$ and $[S_i(ZBack),\ S_{i+1}(ZBack)[$ respectively.



For an alpha channel α,

$$S_{i,new}(\alpha) = 1 - \left(1 - S_i(\alpha)\right)^{\frac{S_{i+1}(Z) - S_i(Z)}{S_i(ZBack) - S_i(Z)}}$$

$$S_{i+2,new}(\alpha) = 1 - \left(1 - S_{i+1}(\alpha)\right)^{\frac{S_{i+1}(ZBack) - S_i(ZBack)}{S_{i+1}(ZBack) - S_{i+1}(Z)}}$$

$$S_{i+1,new}(\alpha) = f + b \cdot (1 - f),$$

where

$$f = 1 - \left(1 - S_i(\alpha)\right)^{\frac{S_{i+1}(Z) - S_i(ZBack)}{S_i(ZBack) - S_i(Z)}},$$

and

$$b = 1 - \left(1 - S_{i+1}(\alpha)\right)^{\frac{S_{i+1}(Z) - S_i(ZBack)}{S_{i+1}(ZBack) - S_{i+1}(Z)}}.$$

For a color channel c,

$$S_{i,new}(c) = \begin{cases} S_i(c) \cdot \dfrac{S_{i,new}(\alpha)}{S_i(\alpha)}, & S_i(\alpha) > 0 \\ S_i(c) \cdot \dfrac{S_{i+1}(Z) - S_i(Z)}{S_i(ZBack) - S_i(Z)}, & S_i(\alpha) = 0 \end{cases}$$

$$S_{i+2,new}(c) = \begin{cases} S_{i+1}(c) \cdot \dfrac{S_{i+2,new}(\alpha)}{S_{i+1}(\alpha)}, & S_{i+1}(\alpha) > 0 \\ S_{i+1}(c) \cdot \dfrac{S_{i+1}(ZBack) - S_i(ZBack)}{S_{i+1}(ZBack) - S_{i+1}(Z)}, & S_{i+1}(\alpha) = 0 \end{cases}$$

$$S_{i+1,new}(c) = S_i(c) \cdot r \cdot \left(1 - \frac{b}{2}\right) + S_{i+1}(c) \cdot s \cdot \left(1 - \frac{f}{2}\right),$$

where

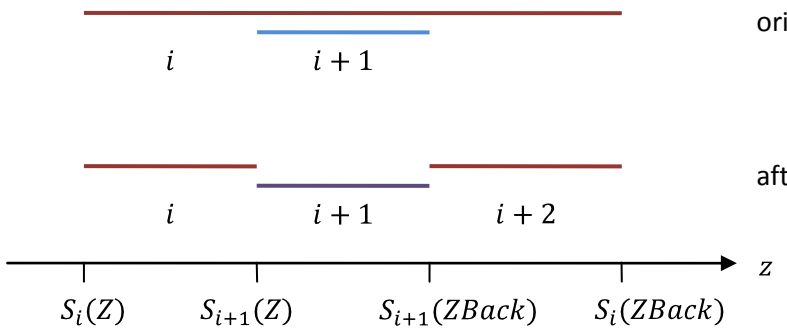
$$r = \begin{cases} \dfrac{f}{S_i(\alpha)}, & S_i(\alpha) > 0 \\ \dfrac{S_{i+1}(Z) - S_i(ZBack)}{S_i(ZBack) - S_i(Z)}, & S_i(\alpha) = 0 \end{cases}$$

and

$$s = \begin{cases} \dfrac{b}{S_{i+1}(\alpha)}, & S_{i+1}(\alpha) > 0 \\ \dfrac{S_{i+1}(Z) - S_i(ZBack)}{S_{i+1}(ZBack) - S_{i+1}(Z)}, & S_{i+1}(\alpha) = 0 \end{cases}$$

The rather elaborate expression for computing $S_{i+1,new}(c)$ is the result of the following procedure: take the part of $S_i(c)$ that overlaps $S_{i+1}(c)$, and call it U. Take the part of $S_{i+1}(c)$ that overlaps $S_i(c)$, and call it V. The color of U is $S_i(c) \cdot r$, and the color of V is $S_{i+1}(c) \cdot s$. (See the definition of $c(z)$ in "Color as a Function of Depth," above.) The value of $S_{i+1,new}(c)$ is the average of compositing U over V and compositing V over U.

If two volumetric samples overlap, and one sample is completely contained inside the other, then create three non-overlapping samples. Replace the original samples, which cover the intervals $[S_i(Z),\ S_i(ZBack)[$ and $[S_{i+1}(Z),\ S_{i+1}(ZBack)[$, where $S_i(ZBack) > S_{i+1}(ZBack)$, with three new samples, numbered i, $i+1$ and $i+2$, that cover the intervals $[S_i(Z),\ S_{i+1}(Z)[$, $[S_{i+1}(Z),\ S_{i+1}(ZBack)[$ and $[S_{i+1}(ZBack),\ S_i(ZBack)[$ respectively.



For an alpha channel α,

$$S_{i,new}(\alpha) = 1 - \left(1 - S_i(\alpha)\right)^{\frac{S_{i+1}(Z) - S_i(Z)}{S_i(ZBack) - S_i(Z)}}$$

$$S_{i+2,new}(\alpha) = 1 - \left(1 - S_i(\alpha)\right)^{\frac{S_i(ZBack) - S_{i+1}(ZBack)}{S_i(ZBack) - S_i(Z)}}$$

$$S_{i+1,new}(\alpha) = f + b \cdot (1 - f),$$

where

$$f = 1 - \left(1 - S_i(\alpha)\right)^{\frac{S_{i+1}(ZBack) - S_{i+1}(Z)}{S_i(ZBack) - S_i(Z)}},$$

and

$$b = S_{i+1}(\alpha).$$

For a color channel c,

$$S_{i,new}(c) = \begin{cases} S_i(c) \cdot \dfrac{S_{i,new}(\alpha)}{S_i(\alpha)}, & S_i(\alpha) > 0 \\ S_i(c) \cdot \dfrac{S_{i+1}(Z) - S_i(Z)}{S_i(ZBack) - S_i(Z)}, & S_i(\alpha) = 0 \end{cases}$$

$$S_{i+2,new}(c) = \begin{cases} S_i(c) \cdot \dfrac{S_{i+2,new}(\alpha)}{S_i(\alpha)}, & S_{i+1}(\alpha) > 0 \\ S_i(c) \cdot \dfrac{S_i(ZBack) - S_{i+1}(ZBack)}{S_i(ZBack) - S_i(Z)}, & S_{i+1}(\alpha) = 0 \end{cases}$$

$$S_{i+1,new}(c) = S_i(c) \cdot r \cdot \left(1 - \frac{b}{2}\right) + S_{i+1}(c) \cdot s \cdot \left(1 - \frac{f}{2}\right),$$

where

$$r = \begin{cases} \dfrac{f}{S_i(\alpha)}, & S_i(\alpha) > 0 \\ \dfrac{S_{i+1}(ZBack) - S_{i+1}(Z)}{S_i(ZBack) - S_i(Z)}, & S_i(\alpha) = 0 \end{cases}$$

and

$$s = 1$$

Analogous to the previous case, $S_{i+1,new}(c)$ is computed as the average of compositing U over $S_{i+1}(c)$ and compositing $S_{i+1}(c)$ over U, where U is the section of $S_i(c)$ that overlaps $S_{i+1}(c)$.

Numerical Issues

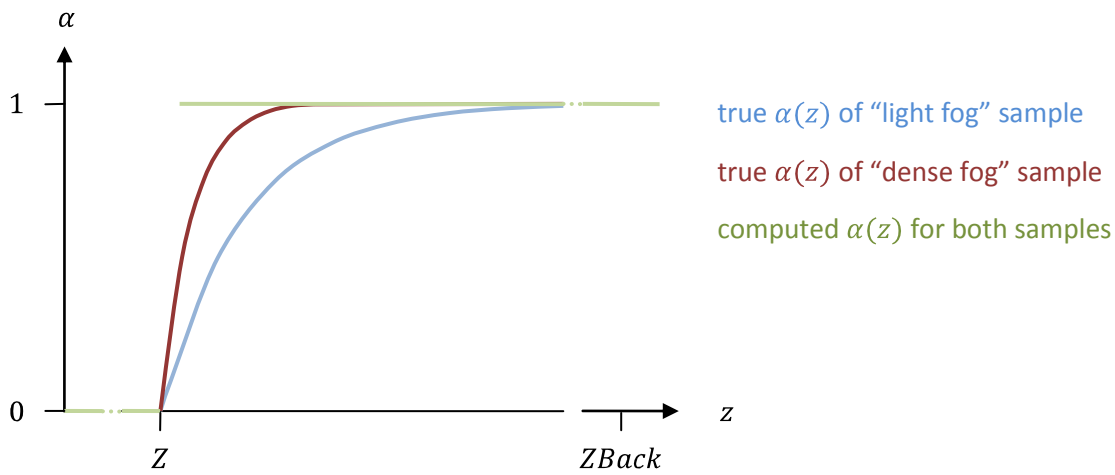
The representation of volumetric samples as described above suffers from a few numerical problems:

- If a volumetric sample is deep enough to become completely opaque, then any information about how quickly alpha and color build up with increasing depth is lost.
- If alpha for a volumetric sample is very close to zero, then evaluating color as a function of depth can produce large rounding errors.
- No volumetric sample can extend out to infinity.

Opaque Volumetric Samples

Volumetric samples represent regions along the z axis of a pixel that are filled with a medium that absorbs light and also emits light towards the camera. The intensity of light traveling through the medium falls off exponentially with the distance traveled. For example, if a one unit thick layer of fog absorbs half of the light and transmits the rest, then a two unit thick layer of the same fog absorbs three quarters of the light and transmits only one quarter. Volumetric samples representing these two layers would have alpha 0.5 and 0.75 respectively. As the thickness of a layer increases, the layer quickly becomes nearly opaque. A fog layer that is twenty units thick transmits less than one millionth of the light entering it, and its alpha is 0.99999905. If alpha is represented using 16-bit floating-point numbers, then it will be rounded to 1.0, making the corresponding volumetric sample completely opaque. With 32-bit floating-point numbers, the alpha value for a 20 unit thick layer can still be distinguished from 1.0, but for a 25 unit layer, alpha rounds to 1.0. At 55 units, alpha rounds to 1.0 even with 64-bit floating-point numbers.

Once a sample effectively becomes opaque, all information about the density of the light-absorbing medium is lost. A one-unit layer of a light fog might absorb half of the light, while a dense fog might absorb three quarters of the light for a layer of thickness 1.0, but the representation of a 60-unit layer as a volumetric sample is exactly the same for the light and the dense fog. For a sample that extends from Z to $ZBack$, the function $\alpha(z)$ evaluates to 1.0 for any $z > Z$. Any object within this layer would be completely hidden, no matter how close it was to the near boundary of the layer.



Color in Highly Transparent Samples

The function $c(z)$ as defined in the section "Color as a Function of Depth," above, is well-behaved in many cases. However, if a volume sample has a very small but non-zero alpha, then computing $c(z)$ requires evaluating an expression that essentially boils down to

$$b = \frac{1-(1-\alpha)^x}{\alpha},$$

where $0 \leq x \leq 1$ and $0 < \alpha \leq 1$.

If $\alpha \ll 1$, then $(1-\alpha)^x$ is very close to 1.0, and floating-point rounding leaves relatively few significant digits. Subtracting this rounded number from 1.0 and dividing the result by the tiny α amplifies the rounding error drastically.

The following table shows the approximate maximum rounding error for any x between zero and one when b is computed using 32-bit floating-point arithmetic:

α	$error$	
0.1	6×10^{-7}	
0.01	4×10^{-6}	
0.001	4×10^{-5}	
0.0001	0.0005	
6.10×10^{-5}	0.0009	smallest positive normalized 16-bit number ($\alpha = HALF\_NRM\_MIN$)
5.96×10^{-8}	0.5	smallest positive 16-bit number ($\alpha = HALF\_MIN$)

When b is computed using 64-bit floating-point arithmetic, then α can be much closer to zero before the rounding error becomes noticeable. For $\alpha = 10^{-16}$ the error can be as large as 0.006, but if α is stored as a 16-bit floating-point number, then any α value less than about 3×10^{-8} will be rounded to zero, and the numerically unstable formula for b no longer applies.

In order to avoid noticeable artifacts due to rounding during operations such as merging two deep images, alpha and color should be stored as half-precision (16-bit) floating-point numbers, but evaluation of color as a function of depth should be done using double precision (64 bits).

Samples that Extend to Infinity

Occasionally one may want the last volumetric sample in a pixel to extend from a finite depth to infinity. Such a sample would essentially say "from here on out, all the way to infinity, space is filled with the medium described by this sample." However, with the way volumetric samples are currently represented (by storing Z, $ZBack$, and the opacity of the entire sample), this is not possible. If $ZBack$ is set to infinity for any sample, then the functions $\alpha(z)$ and $c(z)$ can no longer evaluated. Setting $ZBack$ to the maximum positive floating-point value doesn't work either; any sample with a very large depth ($ZBack \gg Z$) becomes either almost completely transparent or completely opaque.

Alternative Representation of Volumetric Samples

The numerical problem with opaque volumetric samples described above can be avoided if the representation of the samples is changed: instead of interpreting the alpha channel of a volumetric sample as the total opacity of the entire sample, interpret alpha as a scaling factor for z, and re-define the segments $\alpha_{-1}(z)$, $\alpha_0(z)$, $\alpha_1(z)$, … $\alpha_{n-1}(z)$ of the function $\alpha(z)$ as follows:

$$\alpha_i(z) = \begin{cases} 0, & i = -1 \\ f + b \cdot (1 - f), & i \geq 0 \end{cases}$$

where

$$f = \alpha_{i-1}\big(S_i(Z)\big)$$

and

$$b = \begin{cases} S_i(\alpha), & S_i(Z) \geq S_i(ZBack) \\ 1 - \exp\big(-S_i(\alpha) \cdot (\min(z, S_i(ZBack) - S_i(Z))\big), & S_i(Z) < S_i(ZBack) \end{cases}$$

With this interpretation of alpha, for any color channel c, the segments $c_{-1}(z)$, $c_0(z)$, $c_1(z)$, … $c_{n-1}(z)$ of the function $c(z)$ are defined as follows:

$$c_i(z) = \begin{cases} 0, & i = -1 \\ f + b \cdot \Big(1 - \alpha_{i-1}\big(S_i(Z)\big)\Big), & i \geq 0 \end{cases}$$

where

$$f = c_{i-1}\big(S_i(Z)\big)$$

and

$$b = \begin{cases} S_i(c), & S_i(Z) \geq S_i(ZBack) \\ S_i(c) \cdot \dfrac{1 - \exp\big(-S_i(\alpha) \cdot (\min(z,\ S_i(ZBack) - S_i(Z))\big)}{1 - \exp\big(-S_i(\alpha) \cdot (S_i(ZBack) - S_i(Z))\big)}, & S_i(Z) < S_i(ZBack), \quad S_i(\alpha) > 0 \\ S_i(c) \cdot \dfrac{\min(z,\ S_i(ZBack) - S_i(Z))}{S_i(ZBack) - S_i(Z)}, & S_i(Z) < S_i(ZBack), \quad S_i(\alpha) = 0 \end{cases}$$

Interpreting the alpha channel of a volumetric sample as a scaling factor for z still does not support samples where $ZBack$ is set to infinity, but nearly the same effect can be achieved by setting $ZBack$ to the maximum possible floating-point value (FLT_MAX for 32-bit numbers).

Changing how alpha is interpreted for volumetric samples breaks backward compatibility. Depending on how many deep OpenEXR files with volumetric samples exist already, this may or may not be a problem. If backward compatibility must be maintained, “new” volumetric samples could be distinguished from “old” ones by making alpha for new samples negative. In this case the sign of the alpha samples in the expressions above must be flipped: $S_i(\alpha)$ becomes $-S_i(\alpha)$ and vice versa.