Robust BVH Ray Traversal — FMA Correction

Thiago Ize
Solid Angle

Abstract

The original Robust BVH Traversal paper [Ize 2013] had a mistake in the numerical error derivation for the fused multiply-add (FMA) optimized ray-bounding box intersection. Instead of having only 2 ulps of error, this method can actually have extremely large amounts of error. The non-FMA intersection method, as used in the rest of the paper, and for which the measurements in the paper are based on, is still correct at only 2 ulps of error.

1. Introduction

The BVH traversal requires the evaluation of the intersection distance between the ray and axis-aligned planes. For dimension d, this ray-plane intersection distance, t, can be given by

$$t = (b.d - ray_{\text{origin}.d})\frac{1}{ray_{\text{dir}.d}}$$

Floating point arithmetic will produce an intersection distance $\tilde{t}$ which could be different from the true distance, t. The derivation for the numerical error present in $\tilde{t}$ for the non-FMA version is correctly shown in [Ize 2013] to be

$$|\tilde{t} - t| \leq 3\frac{ulp(t)}{2},$$

which corresponds to the floating point arithmetic introducing at most 2 ulps of error.

The FMA optimized version precomputes the following two terms:

$$X = -ray_{\text{origin}}\frac{1}{ray_{\text{dir}}} \tag{1}$$

$$Y = \frac{1}{ray_{\text{dir}}}, \tag{2}$$

which allows us to rewrite our computation as a FMA operation:

$$t = \text{FMA}(b, Y, X) = bY + X$$

However, the resulting numerical analysis for this FMA version was incorrect.

2. Mistake

The underlying mistake in the proof is that we were not differentiating between the floating point error introduced by each operation and assumed that each operation added exactly ε of error. This can cause problems since it then allows one to cancel error terms when two ε terms are subtracted from each other or prevents two truly equal error terms from being subtracted. This manifested itself when we went from Equation 3 to Equation 4

$$\tilde{t} = b\frac{1}{ray_{\text{dir}}}(1+\varepsilon)^2 - ray_{\text{origin}}\frac{1}{ray_{\text{dir}}}(1+\varepsilon)^3 \tag{3}$$

$$\tilde{t} \leq \left(b\frac{1}{ray_{\text{dir}}} - ray_{\text{origin}}\frac{1}{ray_{\text{dir}}}\right)(1+\varepsilon)^3 \tag{4}$$

by increasing the error of

$$b\frac{1}{ray_{\text{dir}}}(1+\varepsilon)^2$$

by $1+\varepsilon$. This would have been fine for the purposes of placing an upper bound on the total error had the ε been unique. Unfortunately, it was not, and so instead of adding error, we ended up subtracting out error from the

$$ray_{\text{origin}}\frac{1}{ray_{\text{dir}}}(1+\varepsilon)^3$$

term. If we reformulate the original proof using a more rigorous notation where arithmetic operation i adds a δ_i of error, and $|\delta_i| < \varepsilon$, then we get the following correct proof which shows that the error is in fact not bounded to just 2 ulps.

3. Correct proof

$$\tilde{t} = \left(b\left(\frac{1}{ray_{\text{dir}}}(1+\delta_1)\right) + \left(-ray_{\text{origin}}\left(\frac{1}{ray_{\text{dir}}}(1+\delta_1)\right)\right)(1+\delta_2)\right)(1+\delta_3) \tag{5}$$

$$\tilde{t} = \frac{b}{ray_{\text{dir}}}(1+\delta_1)(1+\delta_3) - \frac{ray_{\text{origin}}}{ray_{\text{dir}}}(1+\delta_1)(1+\delta_2)(1+\delta_3) \tag{6}$$

$$\tilde{t} = (\frac{b}{ray_{\text{dir}}} - \frac{ray_{\text{origin}}}{ray_{\text{dir}}})(1+\delta_1)(1+\delta_3) - \frac{ray_{\text{origin}}}{ray_{\text{dir}}}(1+\delta_1)(1+\delta_3)\delta_2 \tag{7}$$

$$\tilde{t} = t(1+\delta_1)(1+\delta_3) - \frac{ray_{\text{origin}}}{ray_{\text{dir}}}(1+\delta_1)(1+\delta_3)\delta_2 \tag{8}$$

$$\tilde{t} = t(1+\delta_1+\delta_3+\delta_1\delta_3) - \frac{ray_{\text{origin}}}{ray_{\text{dir}}}(\delta_2+\delta_1\delta_2+\delta_2\delta_3+\delta_1\delta_2\delta_3) \tag{9}$$

where as before, $|\delta| < \varepsilon$ and give the relative errors introduced by each floating point operation. Dropping the higher order δ terms gives us

$$\tilde{t} = t + t(\delta_1+\delta_3) - \frac{ray_{\text{origin}}}{ray_{\text{dir}}}\delta_2. \tag{10}$$

```
float b    = 1/6.99999;
float orig = 1/7.00000;
float dir  = 1e-6;
double b_d = b, orig_d = orig, dir_d = dir;

double t_ref = (b_d - orig_d) * (1 / dir_d);
float  t_f   = (b - orig) * (1 / dir);

float X = -orig * (1/dir);
float Y = (1/dir);
float t_fma  = FMA(b, y, X); // b*Y + X;
printf("double(reference)=%.15g,␣float=%.15g,␣FMA=%.9g\n",
       t_ref, t_f, t_fma);
// outputs:  double(reference)=0.193715096009103,
//           float            =0.19371509552002, FMA=0.18670845
}
```

Listing 1. Proof by example that the FMA version is susceptible to large amounts of error.

The error is thus

$$\tilde{t} - t = t(\delta_1 + \delta_3) - \frac{ray_{\text{origin}}}{ray_{\text{dir}}}\delta_2, \tag{11}$$

and can be bounded as

$$|\tilde{t} - t| \leq |t\delta_1| + |t\delta_3| + \left|\frac{ray_{\text{origin}}}{ray_{\text{dir}}}\delta_2\right| \tag{12}$$

$$|\tilde{t} - t| \leq 2\varepsilon|t| + \left|\frac{ray_{\text{origin}}}{ray_{\text{dir}}}\right|\varepsilon \tag{13}$$

The $\left|\frac{ray_{\text{origin}}}{ray_{\text{dir}}}\right|\varepsilon$ term can contribute significant amounts of error if it is larger than $ulp(t) = 2\varepsilon|t|$. A simple way to cause this to happen is to make $|t|$ small and $\left|\frac{ray_{\text{origin}}}{ray_{\text{dir}}}\right|$ large. This occurs when ray_{origin} and b are very close together, ray_{origin} has large magnitude, and ray_{dir} is small. Listing 1 gives an example of this where the non-FMA 32-bit floating point version has no ulps of error (at 32-bit precision), while the FMA version has 470,208 ulps of error. For this reason, the FMA optimization should be avoided when accuracy is important.

Acknowledgements

Sean Barrett pointed out that the original FMA error analysis was incorrect.

References

IZE, T. 2013. Robust BVH ray traversal. *Journal of Computer Graphics Techniques (JCGT) 2*, 2 (July), 12–27. 0

Thiago Ize, Robust BVH Ray Traversal — FMA correction, *Journal of Computer Graphics Techniques (JCGT)*, vol. 0, no. 0, 0–0, 2015
????

Received:	2014-10-24		
Recommended:	pending review	Corresponding Editor:	
Published:	pending review	Editor-in-Chief:	Morgan McGuire