Appendix: Multi-level Tetrahedralization-based Accelerator for Ray-tracing Animated Scenes

Aytek Aman[1] | Serkan Demirci[1] | Uğur Güdükbay[1] | Ingo Wald[2]

[1]Department of Computer Engineering, Bilkent University, Ankara, Turkey
[2] NVIDIA, Salt Lake City, UT, USA

A | FINDING INITIAL TETRAHEDRON

Nearest-hit traversal on tetrahedralization requires determining the initial tetrahedron that the ray hits (cf. Figure 1). If the ray's origin is inside the boundary of the tetrahedralized area, we use the point-location method in[1] to find the tetrahedron that contains the ray's origin as the initial tetrahedron. If the ray originates from outside the tetrahedralization, we use the faces of the bounding box of the tetrahedralization to find the boundary tetrahedron that the ray first hits. We first locate the point that the ray hits on the boundary. Then we find the face that contains the intersection point. After finding the boundary face, we find the tetrahedron that the face belongs to using a lookup table.

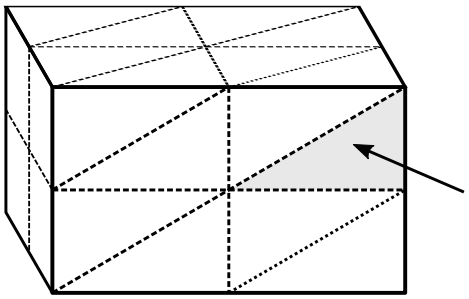


FIGURE 1 Finding the initial tetrahedron in a tetrahedralization. The initial tetrahedron can be found using the bounding face that the ray hits (shaded region).

B | NEAREST-HIT COST CALCULATION

We describe three ways to approximate the average nearest-hit cost; *sampling-based cost calculation*, *average depth-based cost calculation*, and *face count-based cost calculation*. For evaluating the approximation methods, we use tetrahedralizations of models from the Thingi10k[2] data set.

[0]**Abbreviations:** BVH, Bounding Volume Hierarchy; BTH, Bounding Volume Hierarchy-Tetrahedralization Hybrid Acceleration Structure; *k*-d tree, *k*-dimensional tree

B.1 | Sampling-based Cost Calculation

We calculate the cost using the following Monte Carlo approach; given a tetrahedralization, we randomly sample rays that originate at the boundary of the tetrahedralization. Then we traverse the tetrahedralization along the ray, counting the number of tetrahedra traversed. We calculate the average number of tetrahedra traversed from sampled rays as

$$N_{sampling}(\mathcal{T}_{node}) \approx \frac{1}{n}\sum_{0}^{n} N_{ray_i}(\mathcal{T}_{node}) \tag{1}$$

where n is the number of sampled rays and $N_{ray_i}(\mathcal{T}_{node})$ is the tetrahedra count for the randomly sampled ray ray_i. With this approach, the approximated average tetrahedra count's accuracy gets better as the number of sampled rays increases. Although this approach can approximate the tetrahedra count well, it requires an existing tetrahedralization. Constructing a tetrahedralization for each possible node in the BVH hierarchy is slow and not feasible. We only used this method to evaluate other approximation methods that approximate the tetrahedra count without requiring a tetrahedralization to be constructed.

B.2 | Average Depth-based Cost Calculation

The average depth-based calculation estimates the average number of tetrahedra traversed during nearest-hit traversal by estimating the rays' average depth sampled on the boundary of the tetrahedralization. The relation between the average ray depth and tetrahedra count is formalized using Theorem 1:

Theorem 1. Let s be a line segment starts and ends within the given tetrahedralization $\mathcal{T}$. s must stab at least $\frac{\|s\|}{l^*_{max}}$ tetrahedra of $\mathcal{T}$, where l^*_{max} is the length of longest edge in $\mathcal{T}$.

$$N_s(\mathcal{T}) \geq \frac{\|s\|}{l^*_{max}}. \tag{2}$$

Proof. Let τ be a tetrahedron that the line segment intersects. It can be shown that the length of the intersection cannot be greater than l^τ_{max}, where l^τ_{max} is the length of the longest edge of τ. Therefore,

$$\|s \cap \tau\| \leq l^\tau_{max}. \tag{3}$$

By using Equation 3 for each tetrahedron that intersects s, we can sum all the lengths, as in Equation 4:

$$\|s\| = \sum_{\tau\in\mathcal{T}} \|s \cap \tau\| \leq \sum_{\tau\in\mathcal{T}} l^\tau_{max}. \tag{4}$$

Since $l^\tau_{max} < l^*_{max}$ for all $\tau \in \mathcal{T}$ from Equation 4, we get Equation 5.

$$\|s\| \leq N_s(\mathcal{T}) l^*_{max}. \tag{5}$$

□

By using Theorem 1, we can derive an approximation for the average number of traversed tetrahedra as follows:

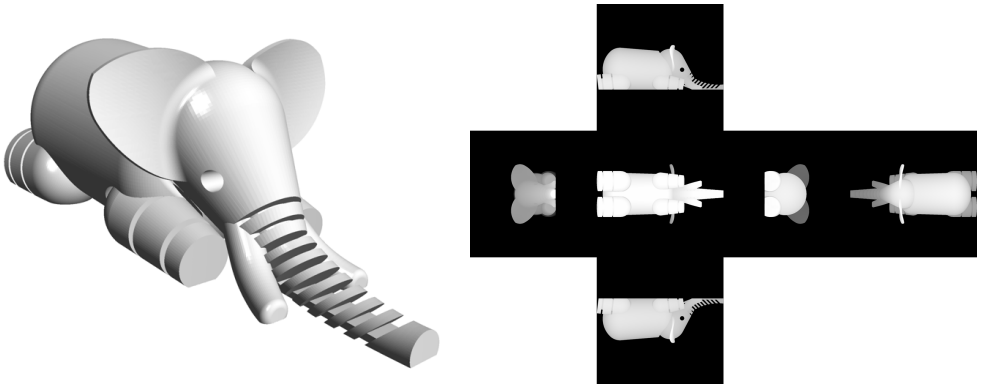
$$N_{depth}(\mathcal{T}_{node}) \approx \frac{d_{avg}}{l^*_{avg}}, \tag{6}$$

where d_{avg} is the average depth of rays in the tetrahedralization and l^*_{avg} is the average length of the edges of the constrained faces. To estimate the average depth of rays, d_{avg}, we used the z-buffer algorithm to calculate a depth map (see Figure 2) for each side of a bounding box of the constrained faces. The calculation of the depth map with resolution R requires $\mathcal{O}(N R^2)$ time complexity in the worst case. Using the calculated depth map, we estimated the average depth. Although the depth map only contains the depth of equally spaced axis-aligned rays, it is fast, and it can estimate average depth well for most of the models.

Figure 3 (a) compare the average depth- and sampling-based methods. As it is seen in the figure, the average depth-based cost calculation method underestimates the cost. To resolve this, we modified Equation 6 as

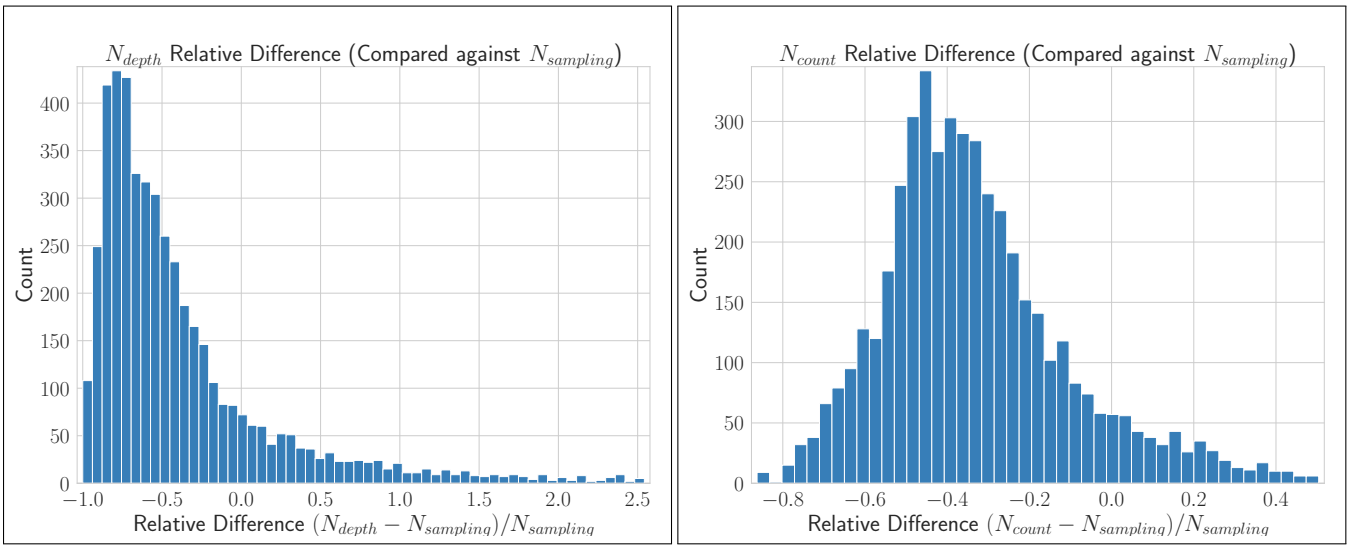
$$N_{depth}(\mathcal{T}_{node}) \approx \frac{d_{avg}}{l^*_{avg}} + C, \tag{7}$$

where C is a constant value.



(a) Rendering **(b)** Depth map

FIGURE 2 The average depth estimation using depth map. The elephant model is the courtesy of Zhou and Jacobson[2].



(a) **(b)**

FIGURE 3 Comparison of the cost estimation methods. (a) The histogram of relative difference of the tetrahedron count between the average depth- and sampling-based estimation methods. (b) The histogram of relative difference of the tetrahedron count between the primitive count-based and the sampling-based estimation methods.

B.3 | Primitive Count-based Cost Calculation

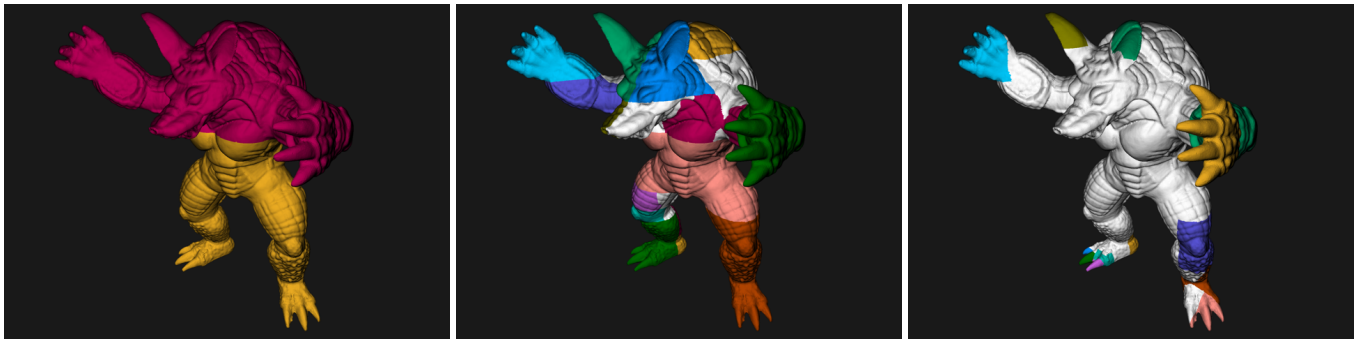
We observe that the average traversed tetrahedra count is related to the cube root of the number of primitives, which is similar to the runtime complexity of regular grids ($\Theta(\sqrt[3]{N})$) shown by Clearly and Wyvill[3]. We define primitive count-based cost as

$$N_{count}(\mathcal{T}_{node}) \approx \sqrt[3]{|F|}. \tag{8}$$

where $|F|$ is the number of primitives. Figure 3 (b) show the comparison of primitive count- and sampling-based methods. We can see that the N_{count} can approximate the average tetrahedron count well.

C | TETRAHEDRALIZATION NEAREST-HIT COST SELECTION

The average depth and primitive count-based tetrahedralization cost methods allow us to select tetrahedralizations based on their estimated cost automatically. Both cost estimation methods rely on C_{tet}, the cost of traversing a single tetrahedron in



(a) $C_{tet} = 0.1$ (b) $C_{tet} = 0.2$ (c) $C_{tet} = 0.3$

FIGURE 4 Tetrahedralizations of BTH constructed using different C_{tet} values. Different tetrahedralizations are colored differently. Gray color is used for faces in the BVH structure.

TABLE 1 BTH32 statistics for varying C_{tet}. "Render" is the rendering time using raytracing. "Constr" is the construction time for the acceleration structure. "Tet_pri" is the number of primitives at the leaf nodes of the BVH that are tetrahedralized. "BVH_pri" is the number of primitives at the leaf nodes of the BVH that are not tetrahedralized. Times are in milliseconds (ms).

		Primitive count-based cost					Average depth-based cost				
Scenes	C_{tet}	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0
Armadillo	Render	134	137	86	80	83	137	101	81	80	81
	Constr	18,592	16,297	1,298	482	479	16,777	4,368	2,151	2,024	2,044
	Tet_pri	345,938	330,809	30,980	0	0	345,938	88,923	1,024	0	0
	BVH_pri	0	15,129	314,958	345,938	345,938	0	257,015	344,914	345,938	345,938
Lumberyard	Render	467	492	476	489	478	481	485	482	475	480
	Constr	6,155	4,800	3,179	2,552	1,971	6,322	4,516	3,420	2,686	2,349
	Tet_pri	119,242	81,051	55,083	32,720	20,121	117,371	80,258	56,919	34,636	24,717
	BVH_pri	901,665	939,856	965,824	988,187	1,000,786	903,536	940,649	963,988	986,271	996,190
Sponza	Render	413	411	409	414	404	410	408	405	408	408
	Constr	364	361	331	310	308	365	369	344	338	316
	Tet_pri	4,764	4,764	2,035	0	0	4,764	4,764	2,035	2,035	0
	BVH_pri	257,503	257,503	260,232	262,267	262,267	257,503	257,503	260,232	260,232	262,267

tetrahedralization. Different values of C_{tet} leads to different BTH structures (see Figure 4). In general, as C_{tet} increased, the BTH construction algorithm selects and constructs fewer tetrahedralizations.

Although we defined C_{tet} as the cost of traversing a single tetrahedron, we used C_{tet} as a parameter for the construction of BTH. We performed a grid search on various models to select the best value for C_{tet}. We used the construction and rendering times in Tables 1 and 2 to select the best C_{tet}. Because BTH32 and BTH20 have different tetrahedra traversal times, we decided to select C_{tet} values for them separately. For BTH32, we decided to use 0.7 for C_{tet}. For BTH20, we decided that 0.5 is a good choice for C_{tet}. Additionally, Tables 1 and 2 show the effect of changing C_{tet}. As C_{tet} increases, more primitives at the leaves of the BVH structure are tetrahedralized; hence, the construction of the structure takes more time.

We also utilize Tables 1 and 2 to select the cost method. These tables show that tetrahedralizations built by both the average-depth-based cost and primitive count-based cost methods can speed-up the rendering times of BTH. Although the average depth-based cost method's calculation takes more computation time than the primitive count-based cost method, the rendering times of the average depth-based cost method are usually faster than the primitive count-based cost method. In the experiments, we only use the average depth-based method to construct BTH structures.

TABLE 2 BTH20 statistics for varying C_{tet}. Confirm Table 1 caption for details.

		Primitive count-based cost					Average depth-based cost				
Scenes	C_{tet}	0.1	0.3	0.5	0.7	1.0	0.1	0.3	0.5	0.7	1.0
Armadillo	Render	112	111	75	81	79	112.60	84.38	77.47	77.93	76.99
	Constr	18,883	15,812	1,302	466	491	16,751	4,200	2,031	2,028	1,985
	Tet_pri	345,938	330,809	30,980	0	0	345,938	88,923	1,024	0	0
	BVH_pri	0	15,129	314,958	345,938	345,938	0	257,015	344,914	345,938	345,938
Lumberyard	Render	449	447	450	451	450	451	449	452	451	449
	Constr	6,204	4,371	3,139	2,398	1,981	6,374	4,405	3409	2699	2329
	Tet_pri	119,242	81,051	55,083	32,720	20,121	117,371	80,258	56,919	34,636	24,717
	BVH_pri	901,665	939,856	965,824	988,187	1,000,786	903,536	940,649	963,988	986,271	996,190
Sponza	Render	392	392	388	381	380	387	376	379	376	379
	Constr	364	377	333	307	304	376	365	332	336	310
	Tet_pri	4,764	4,764	2,035	0	0	4,764	4,764	2035	2035	0
	BVH_pri	257,503	257,503	260,232	262,267	262,267	257,503	257,503	260,232	260,232	262,267

References

1. Aman A, Demirci S, and Güdükbay U. Compact Tetrahedralization-based Acceleration Structure for Ray Tracing. arxiv preprint. 2021;**arXiv:2103.02309**.

2. Zhou Q, and Jacobson A. Thingi10K: A Dataset of 10,000 3D-Printing Models. arXiv preprint. 2016;**arXiv:1605.04797**.

3. Cleary JG, and Wyvill G. Analysis of an Algorithm for Fast Ray Tracing Using Uniform Space Subdivision. The Vis Comp. 1988;**4**(2):65–83.