Volume visualization with ray-guided segment query

Vinojan Rajendiran[1], Jonathan Sarton[1], Jean-Michel Dischler[1]

English Abstract—We propose a novel approach to address the traversal of unstructured grid volumes for volume rendering. Our method leverages the ray-tracing (RT) cores of modern GPUs and reduce the number of point location queries, while ensuring precise and efficient sampling along the ray path. By identifying the full sampling interval within each cell with a single query, regardless of the number of required samples based on cell size and (possibly adaptive) sampling step size, we optimize data traversal and achieve accurate interpolation of volumetric data.

1 Introduction

Volume rendering is a core topic in computer graphics, and especially in scientific visualization, where it aids in exploring large 3D datasets arising from experiments or numerical simulations. Lately, growing dataset sizes challenge real-time rendering: it has become more difficult to reach the high frame rates and interactivity needed for users to be able to extract meaningful insights and accelerate their analytical workflows. Volume rendering techniques must adapt to these constraints by developing efficient strategies for data storage, access, and interpolation. Among the data structures used, unstructured grids are particularly valuable in numerical simulations as they allow for localized spatial refinement. However, unlike regular grids, where cell traversal is straightforward, unstructured grids introduce additional complexity due to their irregular topology. One of the major challenges is thus to traverse these meshes efficiently while ensuring precise sampling of the volumetric field.

Two main approaches exist for traversing an unstructured grid volume in the context of ray tracing: (1) explicit cell-by-cell traversal (cell-marching) [3], where the ray is followed through each intersected cell by identifying successive intersections with adjacent cell faces, and (2) direct point location queries (point query) [4], where a sampling point is directly localized within its corresponding cell. Recently, the advent of ray-tracing (RT) cores on modern GPUs has significantly improved the efficiency of point location in complex meshes by leveraging dedicated hardware acceleration structures. However, their direct use in volume rendering still presents several challenges, particularly due to the potentially high number of samples required along a single ray, which can lead to an excessive number of costly ray tracing core (RT Core) queries.

Our work proposes a hybrid approach that combines the advantages of both previously mentioned strategies. Our method relies on RT cores to identify the entire sampling interval within a cell while minimizing the number of point location queries, allowing for efficient cell-to-cell traversal. A ray passing through a cell can be identified as a continuous and nonzero segment along which we aim to sample at multiple points and in an ordered manner along the ray direction. By taking these specificities into account, our approach ensures a structured and efficient sampling process, reducing redundant computations and improving rendering performance.

2 Background

Hardware-accelerated point query [2], [5] proposed to leverages RT cores in order to solve the cell location problem. These RT cores (1) construct a bounding volume hierarchy (BVH) structure on all the the tessellated triangles of the faces in the mesh, (2) traverse the BVH and perform the ray-triangle intersection test upon query using dedicated hardware, and (3) return the intersected geometry.

The point query algorithm, exploits the fact that any ray originating from a point that lies within a cell will always intersect with one of the faces of this cell. By triangularizing those faces and creating a hardware-accelerated BVH over them, it is possible to launch a hardware-accelerated ray in any direction from the sampling point position and intersect a triangle of the containing cell, and thus retrieve the cell thanks to meta-data attached to the triangle (Figure 1).

3 Method

We propose an optimization of the point query algorithm for traversal of unstructured meshes that aims to reduce the number of calls made to the RT cores

- [1]ICube UMR 7357, CNRS, University of Strasbourg, France
- E-mail : [v.rajendiran, sarton, dischler]@unistra.fr

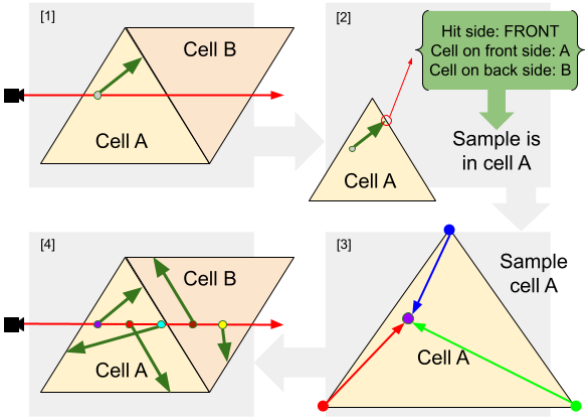


Fig. 1. Hardware-accelerated point query. [1] Launch a primary ray (red) through the mesh, for each sample along this ray, launch a secondary hardware-accelerated ray (green). [2] Hardware-accelerated point query returns meta-data of the intersected triangle, determine the containing cell. [3] Sample the cell. [4] Repeat point query for every subsequent samples.

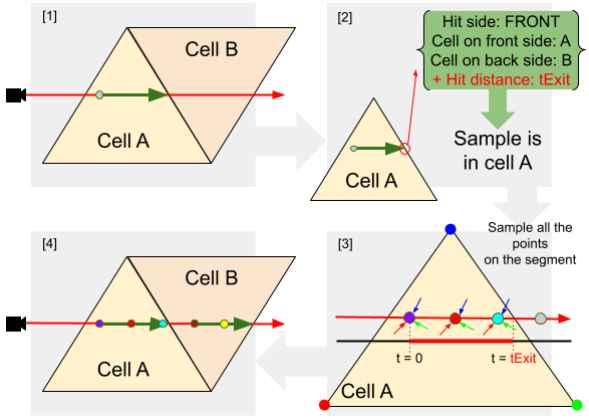


Fig. 2. Hardware-accelerated segment query. [1] Launch a primary ray (red) through the mesh, for the first sample in each cell along this ray, launch a secondary hardware-accelerated ray (green) in the same direction as the primary ray. [2] Hardware-accelerated segment query returns meta-data of the intersected triangle, as well as the exit distance, determine the containing cell. [3] Sample at all the desired sampling points in the cell. [4] Repeat segment query for every subsequent cells.

by taking advantage of successive samples position and cell location coherency on rays. We notice that during volume ray casting of an unstructured mesh, multiple consecutive samples along one ray are often contained in the same cell. This phenomenon is emphasized when increasing the number of samples (i.e. reducing the distance between two samples) to achieve greater visual quality. We exploit this spatial coherency between samples to introduce ray-guided segment query.

When using point query method, the ray-triangle intersection test returns an additional information that can be used: the hit distance from the origin of the ray. Our ray-guided segment query exploits this information by launching the secondary hardware-accelerated ray in the same direction as the primary ray (see Figure 2 [1]) instead of an arbitrary direction (Figure 1 [1]). By doing this, we can query at the same time, the cell containing the sampling point, as well as the exit point of the primary ray on a face of this cell. We thus have access to the entire sampling segment inside the cell: we can sample as desired on this segment without any additional RT core calls by testing if the sampling points belong to segment.

4 Results

We compare the performance of the hardware-accelerated point query method against our segment query method. We also use adaptive sampling [1] in addition to both methods.

Our results are shown in Table 1 for vertex-centered data and Table 2 for cell-centered data. We observe overall performance increase while using our segment query method. However, we notice that we obtain our highest performance increase for simpler sampler interpolation methods (cell-centered or vertex-centered tetrahedrons).

We provide visual representations of the number of total RT core calls per ray for each datasets, for our segment query method and the point query method (Figure 3).

5 Acknowledgments

We acknowledge funding from the French National Agency through the LUM-Vis project (ANR-21-CE46-0005). We thank John M. Patchett from Los Alamos National Laboratory for the DeepWaterImpact dataset.

References

[1] N. Morrical, W. Usher, I. Wald, and V. Pascucci. Efficient space skipping and adaptive sampling of unstructured volumes using hardware accelerated ray tracing. In *2019 IEEE Visualization Conference (VIS)*, pages 256–260. IEEE.

[2] N. Morrical, I. Wald, W. Usher, and V. Pascucci. Accelerating unstructured mesh point location with RT cores. 28(8):2852–2866.

[3] P. Muigg, M. Hadwiger, H. Doleisch, and H. Hauser. Scalable hybrid unstructured and structured grid raycasting. 13(6):1592–1599.

[4] B. Rathke, I. Wald, K. Chiu, and C. Brownlee. SIMD Parallel Ray Tracing of Homogeneous Polyhedral Grids. page 9, 2015.

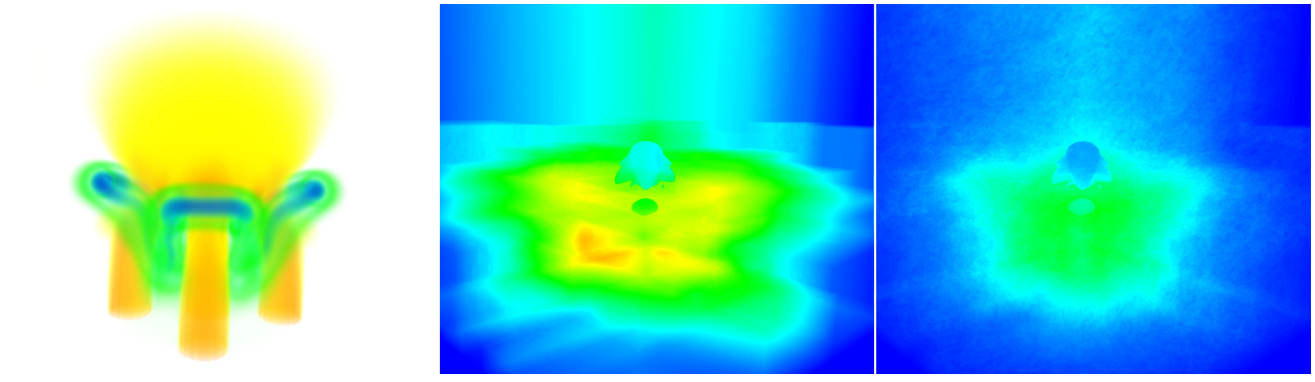
[5] I. Wald, W. Usher, N. Morrical, L. Lediaev, and V. Pascucci. Rtx beyond ray tracing: exploring the use of hardware ray tracing cores for tet-mesh point location. In *Proceedings of the Conference on High-Performance Graphics*, HPG '19, page 7–13, Goslar, DEU, 2019. Eurographics Association.

Dataset	Method	Minimum adaptive sampling step							Highest speed-up
		0.005	0.05	0.1	0.2	0.4	0.8	1.0	
jets	Segment query (ours)	20	31	37	44	54	69	76	×2.50
	Point query	8	18	25	35	50	70	77	
DeepWaterImpact (t=120)	Segment query (ours)	17	19	20	21	23	26	27	No speed-up
	Point query	17	20	21	22	24	26	27	

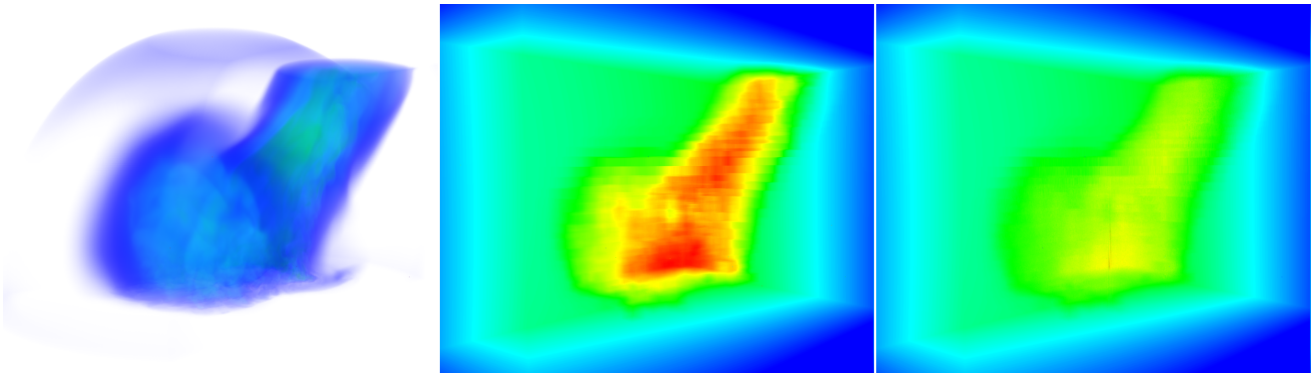
TABLE 1
FPS of the segment query (ours) and point query methods, using vertex-centered scalar fields datasets, for different minimum adaptive sampling step values (in green: performance increase, in orange: similar performance).

Dataset	Method	Minimum adaptive sampling step							Highest speed-up
		0.005	0.05	0.1	0.2	0.4	0.8	1.0	
jets	Segment query (ours)	26	40	46	53	63	79	86	×2.89
	Point query	9	20	27	38	53	72	80	
DeepWaterImpact (t=120)	Segment query (ours)	34	34	35	36	39	43	45	×1.21
	Point query	28	31	34	37	40	42	45	

TABLE 2
FPS of the segment query (ours) and point query methods, using cell-centered scalar fields datasets, for different minimum adaptive sampling step values (in green: performance increase, in orange: similar performance).



(a) Jets



(b) DeepWaterImpact (t=120)

Fig. 3. Render of 2 datasets using our segment query method. The number of hardware-accelerated RT core calls per pixel is represented on the middle (point query method) and right (segment query method, ours) images (from 0 RT calls in blue to the maximum RT calls for a single ray in red,). Rendered on a Nvidia RTX A6000 at 1024×1024 pixels.