Academic Journal of Science and Technology
ISSN: 2771-3032 | Vol. 8, No. 3, 2023

The Reviewed of Ray Tracing Technology Optimization

Danni Sun[1, a]

[1]Xiangjiang International High School, Guangzhou, 511399, China
[a]alictius@gmail.com

Abstract: The technology of ray tracing is a graphics rendering technique that enables the simulation of realistic lighting effects, including accurate shadows, reflections, and refractions. The fundamental principle of ray tracing technology lies in the emission of one or more rays from the camera, which traverse along the path intersecting with objects within the scene. Subsequently, it computes the color and intensity of light based on object materials and light source positions, ultimately amalgamating these rays into a synthesized final image. Ray tracing technology enhances the authenticity and precision of graphics, albeit at the expense of substantial computational overhead and resource consumption, necessitating optimization for improved rendering efficiency and performance. The present paper provides a comprehensive overview of various optimization techniques employed in ray tracing technology, encompassing spatial acceleration structures, importance sampling, differential rendering, noise reduction, among others. Through rigorous experimentation and analysis, the advantages, disadvantages, and applicable scenarios of these methods are thoroughly examined.

Keywords: Ray tracing, Optimization, Review.

1. Introduction

Ray tracing encompasses various categories, with the most common being forward ray tracing. This technique involves emitting rays from a light source and calculating their intersections with objects as well as the direction of reflection until they reach the viewpoint or exceed the maximum depth. While capable of simulating shadows, reflections, and refractions, this method suffers from inefficiency due to a significant portion of light failing to reach the viewpoint. Reverse ray tracing, a technique where light is emitted from the viewpoint and its intersection with objects and direction of reflection are calculated along the light path until it reaches the source or exceeds the maximum depth, enables simulation of phenomena such as global illumination, soft shadows, and indirect illumination. However, this method suffers from inefficiency due to significant light loss before reaching the light source. Distributed ray tracing, a method based on forward or reverse ray tracing, involves emitting multiple random directions of secondary rays at each intersection to simulate light phenomena such as scattering and diffraction. This approach enables the simulation of effects like depth of field, motion blur, and soft reflection; however, it is less efficient due to the increased computational load required for accurate light calculations. Monte Carlo ray tracing is a probabilistic and statistical technique that employs random numbers to generate and sample light, enabling an approximation of real-world physical models. This method exhibits the capability to simulate intricate phenomena such as global illumination, indirect illumination, and high dynamic range; however, it suffers from lower efficiency due to the computation of substantial amounts of light and potential noise generation. In 2015, a ray tracing acceleration method based on Intel's multi-core processor was proposed [1], primarily addressing scene traversal and ray intersection challenges in photorealistic rendering engines. This paper manually vectorizes the ray tracing algorithm using SIMD instruction set and designs a suitable BVH structure considering the characteristics of MIC architecture. The effectiveness and universality of the proposed method are validated through experimental analysis in this paper, and compared with alternative acceleration techniques. Moreover, the applicability of the presented method extends beyond Intel multi-core processors, as it can be adapted to various CPU platforms by adjusting parameters related to SIMD instruction set and BVH structure.

2. Rapid Segmentation Algorithm Based on A Greedy Method

The traditional KD-tree is a spatial partitioning-based data structure utilized for the storage and querying of multidimensional data. The fundamental concept involves recursively dividing the data space into two subspaces, with each partition selecting a dimension as the segmentation axis and determining the segmentation position based on the median value of data points within this dimension. The subspace in the KD-tree corresponds to each node, where the root node represents the entire data space and the leaf node represents either a single data point or a group of similar data points. However, one drawback of traditional KD-trees is that their segmentation effectiveness deteriorates when dealing with high-dimensional data, leading to a decrease in query efficiency. Furthermore, traditional KD-trees are unsuitable for handling dynamic data due to the disruption of tree balance and the need for rebuilding when inserting or deleting data points. To enhance the efficiency of KD-Tree creation, a rapid segmentation algorithm based on a greedy method was employed in this study, as reported by the Chinese Academy of Sciences in 2015 [2]. The fundamental concept of the algorithm lies in selecting an empty subspace as the segmentation result during each space segmentation. In cases where no empty subspace exists, direct segmentation is performed based on the midpoint position. This approach eliminates the need for ordering slice coordinates, thereby reducing computational complexity and memory access. The algorithm is also well-suited for GPU implementation as it does not necessitate synchronization or communication operations, only simple logical judgments and mathematical computations. Consequently, an efficient method for KD-Tree

creation is achieved, exhibiting approximately 20% greater efficiency compared to traditional KD-trees. Moreover, GPU-based ray tracing demonstrates a six-fold increase in efficiency over CPU-based approaches.

3. Dynamic Optimization of Grid Resolution Algorithm

In 2008, the State Key Laboratory of Computer Science at the Institute of Software, Chinese Academy of Sciences, proposed a surface-based cost pre-estimation method for optimizing grid organization in ray tracing calculations. Through experimental validation, an algorithm was developed to dynamically optimize grid resolution using this approach [3]. In contrast to existing methods, the approach proposed in this paper eschews the use of points or lines for surface simulation and instead directly leverages geometric information and distribution characteristics to estimate cost. Furthermore, it takes into account potential premature termination during ray tracing. Consequently, our method achieves more accurate grid cost estimation and enables more effective optimization of grid resolution. The method proposed in this paper demonstrates adaptability to dynamically changing scenarios, distinguishing it from existing methods that are limited to static scenarios. Experimental results validate the effectiveness and superiority of the proposed approach, while also showcasing its potential for seamless extension to other uniform mesh-based acceleration structures – a promising avenue for future research in this domain.

4. Approach for Efficient NURBS Surface Search During Ray Tracing

In 2021, a method for optimizing ray tracing for clipping NURBS surfaces on GPUs was published in the Wiley Journal [4]. The utilization of NURBS surfaces as representations for geometric models is a prevalent technique within the CAD industry. In CAD applications, surface drawing commonly employs mesh division and Z-buffer drawing techniques. However, due to the intricate computations involved, real-time performance remains challenging to achieve when employing ray tracing on NURBS surfaces, which explains its limited industrial usage. In this paper, we propose a novel approach for efficient NURBS surface search during ray tracing. Our proposed kd-tree-based two-dimensional data structure not only accelerates the speed of ray tracing but also exhibits lower memory consumption and shorter preprocessing time compared to previously published methods. Furthermore, we have incorporated cutting-edge ray tracing technology to efficiently trim NURBS surfaces on GPUs. Through meticulous design and implementation, our approach achieves real-time performance by enabling the GPU to handle thousands to hundreds of millions of ray casts per second in medium to large complex scenes comprising numerous NURBS surfaces and pruning curves.

5. Conclusion

The present paper provides a comprehensive summary of four ray tracing acceleration methods. These methods encompass the utilization of Intel multi-core processors, SIMD instruction sets, and MIC architecture to optimize both the ray tracing algorithm and BVH structure, thereby enhancing the efficiency and universality of photorealistic rendering. The greedy fast segmentation algorithm prioritizes the selection of empty subspaces during KD-tree construction, thereby eliminating the need for sorting operations and reducing computational complexity and memory access. This approach is well-suited for implementation on GPUs. The cost estimation method based on surface is employed to estimate the mesh cost by utilizing geometric information and distribution characteristics of the surface. Additionally, the mesh resolution is dynamically optimized according to this cost, enabling it to adapt to dynamic scene changes. Moreover, this approach can be extended to other uniform mesh-based acceleration structures. The present study optimizes the GPU-based ray tracing method for Nurbs surface clipping by employing a two-dimensional kd-tree data structure to enhance point search efficiency. This approach exhibits advantages such as low memory consumption, short preprocessing time, and real-time performance. The paper provides a concise introduction to the principles, characteristics, and effects of these methods while highlighting their potential applications and future developments in both industry and academia.

References

[1] Song, Y., Wang L.,Meng X.,(2015) Parallel Ray Tracing Acceleration Method based on Intel Integrated Multi-core Architecture. Journal of Computer Aided Design and Graphics.

[2] Yue T., Zhao H., Hua H. (2015) Research and Implementation of CUDA-based Ray Tracing Optimization Algorithm. Shenyang Institute of Automation, Chinese Academy of Sciences.

[3] Li J., Wang W., Wu E., (2008) Grid Optimization for Accelerating Ray Tracing Calculation. Journal of Computer-Aided Design and Graphics.

[4] Sloup J.,Havran V.(2021)Optimizing Ray Tracing of Trimmed NURBS Surfaces on the GPU. Computer Graphics Forum, Volume 40: Issue 7, 161-172.