Postprint of VCT Global Illumination Algorithm Based on Cascaded Voxel Textures

Authors: Zhang Jing, Wang He, Zhang Xiaodong

Date: 2018-12-13T00:00:00+00:00

Abstract

This paper investigates the problem that rendering global illumination for large-scale scenes often fails to meet real-time requirements due to excessive computational cost, and proposes a high-performance Voxel Cone Tracing (VCT) global illumination algorithm. The algorithm comprises: a) In the voxelization stage, a novel scene lighting information representation structure termed "cascaded voxel texture" structure, which substantially reduces memory consumption for scene storage and exhibits anisotropic properties, offering advantages of high speed and high quality when processing large-scale scenes; b) In the lighting computation stage, an improved cone tracing filter based on the cascaded voxel texture structure, which enhances the efficiency of light injection and reflected light calculation; c) An improved scene voxelization update strategy that further elevates the algorithm' s runtime frame rate. Experimental results demonstrate that the proposed algorithm significantly outperforms the original VCT algorithm in reducing memory footprint and improving speed, thereby satisfying the real-time requirements for global illumination computation in large-scale scenes.

Full Text

Preamble

Vol. 37 No. 2
Application Research of Computers
ChinaXiv Partner Journal

VCT Global Illumination Algorithm Based on Cascade Voxel Texture

Zhang Jing[1],[2],[3], **Wang He**[1], **Zhang Xiaodong**[1]
(1. College of Computer Science & Technology, Harbin Engineering University, Harbin 150006, China;
2. College of Engineering, Shantou University, Shantou Guangdong 515063,

China;
3. School of Information Science & Engineering, Jinan University, Jinan 250022, China)

Abstract: To address the challenge that global illumination rendering of large-scale scenes often fails to meet real-time requirements due to excessive computational overhead, this paper proposes a high-performance voxel cone tracing (VCT) global illumination algorithm. The algorithm comprises: (a) a novel scene illumination information representation structure called "cascade voxel texture" for the voxelization phase, which dramatically reduces memory consumption for scene storage while providing anisotropic properties, offering superior speed and quality for large-scale scenes; (b) an improved cone tracing filter based on the cascade voxel texture structure for the illumination calculation phase, which enhances the efficiency of light injection and reflected light computation; and (c) an improved scene voxelization update strategy that further increases runtime frame rates. Experimental results demonstrate that the proposed algorithm significantly outperforms the original VCT algorithm in both memory reduction and speed improvement, satisfying real-time global illumination computation requirements for large-scale scenes.

Keywords: virtual reality; large-scale scene; real-time global illumination; cascade voxel texture; voxel cone tracing

---

0 Research Status

In virtual environments, realistic lighting is a crucial source of immersion, with global illumination being the most important technical approach. Effective simulation of global illumination can substantially enhance scene rendering realism. However, due to the high difficulty of simulating indirect illumination in complex scenes, global illumination algorithms demand high real-time performance and accuracy. Consequently, a faster, approximate, and adaptive solution is needed to address real-time global illumination computation in large-scale complex scenes. Currently, mainstream global illumination algorithms include the ray tracing algorithm proposed by Turner Whitted in 1979 [**?**] and its improvements \cite{2~4}, the photon mapping method proposed by Jensen in 1996 [**?**] and its improvements [**?**, **?**], and the instant radiosity algorithm proposed by Keller in 1997 [**?**, **?**] and its improvements [**?**, **?**]. To date, photon mapping algorithms have not achieved real-time performance and suffer from temporal flickering artifacts. While ray tracing and instant radiosity algorithms have achieved interactive performance under complex lighting conditions, they remain unsuitable for real-time global illumination in large-scale complex scenes.

To meet the demand for dynamic global illumination algorithms, Crytek first proposed the Cascaded LPV algorithm in 2010 [**?**, **?**], which used a cascaded texture structure to store illumination information, achieving significant performance improvements that inspired this work. In 2011, Crassin et al. [**?**]

proposed the voxel cone tracing algorithm (VCT), also known as the VXGI algorithm. Cyril introduced a sparse voxel octree structure for voxel storage in the VCT algorithm, which offered high illumination computation efficiency and utilized approximate computation to simulate various indirect illumination effects with better real-time performance than LPV. However, it required enormous storage space and exhibited poor voxel traversal efficiency. In 2012, Laine et al. [**?**] proposed an improved sparse voxel octree structure (ESVO) that somewhat reduced storage overhead, but still allocated approximately 40% of storage space for node encoding.

In 2013, Hornung et al. [**?**] proposed Morton-code-based octree node encoding (MSVO), which could reduce some pointers, but its specified storage method caused redundant storage between identical nodes. In 2017, Yuan Yuwei et al. [**?**] proposed a sparse voxel directed acyclic graph-based illumination computation acceleration structure (SVDAG), which greatly reduced voxel storage space while accelerating the rendering process. However, since this algorithm achieves storage reduction by merging nodes, its effectiveness is unsatisfactory when handling scenarios with substantial information, making it disadvantageous for global illumination requiring multiple types of information. In 2018, Yuan Lu [**?**] improved global illumination efficiency by computing virtual point lights through the graphics rendering pipeline and performing deferred shading calculations with clipped light sources, but still could not satisfy real-time rendering for large dynamic scenes.

Based on the above research and addressing the limitations of the original VCT algorithm, this paper proposes a VCT global illumination algorithm for large-scale scenes based on a cascade voxel texture structure. The improvements primarily include: (a) By studying large-scale scene structures and human visual characteristics, we propose using cascade voxel textures to store voxels. This hierarchical scene representation dramatically reduces scene voxel scale and improves node access efficiency. Based on the cascade voxel texture structure, we further improve the voxelization algorithm, making scene voxelization faster, more efficient, and anisotropic. (b) The primary complexity of global illumination computation lies in the massive, repetitive ray tracing tasks for parallel light rays. According to the characteristics of the cascade texture storage structure, we propose a more efficient cone filter. The improved cone filter significantly enhances the illumination computation efficiency of the VCT global illumination algorithm. (c) We propose a scene voxel update strategy suitable for the cascade structure based on the improved cone filter' s structural characteristics, which improves runtime throughput and reduces rendering cycles.

Finally, through multiple comparative experiments, we demonstrate that the proposed method substantially improves efficiency while maintaining illumination rendering quality and reduces memory overhead, offering better performance and realism.

1.1 Cascade Voxel Texture Structure

In current global illumination research, most methods employ sparse octree and improved sparse octree storage structures. However, in sparse octree structures, the number of voxels increases with scene scale, leading to explosive growth in large-scale scenes. Additionally, due to the tree structure's inefficiency for traversal and lookup and its lack of anisotropy, voxelization is often time-consuming.

To address these issues, this paper proposes a novel voxel storage structure: the cascade voxel texture structure. Based on the characteristic that human eyes are more sensitive to nearby objects while less sensitive to distant changes, this structure divides the scene into n distinct regions by distance. These n regions are represented by nested grids of different sizes [**?**]. Regions closer to the viewer use smaller grids for finer object partitioning with higher resolution, while distant regions use larger grids with lower resolution. The n region grids are nested and extend along the viewing direction, forming nested cube shapes, as shown in [Figure 1: see original paper]. This hierarchical representation achieves large-scale scene representation with fewer grids, avoiding explosive voxel count growth with increasing scene scale while reducing propagation iteration counts required for ray tracing. For convenience, we set $n = 6$ in this paper.

[Figure 1: see original paper]

The smallest texel unit in the cascade voxel texture corresponds to a voxel in the scene. Through this one-to-one correspondence, geometry-related attribute data [**?**] from voxels is stored in 3D textures called G-buffers [**?**]. Each G-buffer attribute and light buffer requires a cascade texture for storage. To achieve anisotropy, this structure stores information for each of the six faces of every voxel. The cascade texture structure is illustrated in [Figure 2: see original paper]. The X-axis represents the six faces of voxels, the Y-axis represents the n cascade levels, and each texture unit consists of 32$×32×$32 voxels. Based on the image characteristics in our experiments, we set $n = 6$, providing reasonable scene coverage from near to far. Since voxels in this structure are anisotropic [**?**], illumination computation accuracy is ensured.

[Figure 2: see original paper]

This structure facilitates trilinear interpolation between voxels, but care must be taken to avoid sampling errors from other levels or faces during computation. We optimize the voxelization process according to the cascade voxel texture storage structure, with the improved voxelization workflow shown in [Figure 3: see original paper].

[Figure 3: see original paper]

1.2 Improved Cone Tracing Based on Enhanced Cone Filter

Global illumination computation typically requires extensive ray tracing. While ray tracing produces excellent rendering results, it is computationally expensive. In 2011, Crassin proposed cone tracing, which combines spatial and directional coherence of light rays with anti-aliasing techniques [**?**, **?**], effectively addressing the high computational cost of ray tracing.

Crassin' s cone filter size corresponds to the mip levels of the voxel tree storage structure. Each illumination collection using the filter requires traversing the tree structure to locate corresponding voxels, resulting in low computational efficiency. To more efficiently collect radiance from scenes through cone filters, this paper proposes a novel cone filter structure based on the cascade voxel texture storage structure.

The improved structure consists of n sample modules of different sizes arranged sequentially along the cone filter direction. Structural details are shown in [Figure 4: see original paper].

[Figure 4: see original paper]

The improved cone filter corresponds to different cascade texture levels at different positions. During radiance collection, no traversal lookup is required. Instead, the sample size, filter diameter, and cascade level are dynamically calculated based on sampling coordinates to determine the specific voxel position in the cascade texture, as shown in [Figure 5: see original paper].

[Figure 5: see original paper]

As shown in [Figure 6: see original paper], the hemispherical integral in the rendering equation [**?**] can approximate a sphere as the sum of integrals of n cones. Voxel cone tracing approximates their increments, and weighted results yield accumulated increments at the cone origin. Assuming a diffuse surface where the bidirectional reflectance distribution function is constant for all incoming and outgoing angles, and incoming radiance in each cone is also constant, the reflected radiance L_r at point x can be rewritten as:

$$L_r(x) = \rho \sum_{k=1}^{n} W_k L_k(x, \omega_i)$$

where is the diffuse coefficient or albedo [**?**], W_k is a weight function with sum 2π to simulate diffuse reflection, and *ω_i* represents the preset cone direction. The incoming radiance L_k within the cone filter is obtained by splitting the cone into continuous sample elements and accumulating them, using the sampling value blending method in our cone filter.

Each cone undergoes filter sampling to obtain sampled node values ΣW, and the accumulated radiance collected by 16 cones is stored as the voxel' s radiance

in the V-buffer. Equation (3) dynamically updates the reflected light color c and ambient occlusion value α:

$$c = c_2 + (1 - \alpha_2)c, \quad \alpha = \alpha_2 + (1 - \alpha_2)\alpha$$

where c_2 and α_2 can be directly obtained from the current voxel, while c and α are accumulated values of c_2 and α_2 [**?**]. After direct illumination is injected into the light buffer, diffuse reflection is simulated using 16 wide cones distributed in the normal direction, while specular reflection uses a single narrow cone in the reflection direction. All sampling results are weighted by the voxel' s albedo and added to the direct illumination value to obtain the voxel' s radiance, including direct illumination and first-bounce indirect illumination. Repeating the anisotropic filtering process yields second-bounce indirect illumination. Simultaneously, using the ambient occlusion value α and ambient occlusion term ao, the ambient occlusion effect is calculated according to Equation (4):

$$ao = \frac{1}{1 + \lambda r}$$

where r is the cone' s current radius and λ is the attenuation coefficient of *f(r)* with distance. Multiple reflections are computed to generate the final rendered image.

1.3 Improved Scene Voxel Update Strategy Based on New Structure

Traditional methods voxelize the entire scene initially, then divide it into dynamic and static parts for subsequent updates. When dynamic parts change, their voxelization structure is deleted and re-voxelized. This process occurs every frame, causing rapid computational overhead increase and reduced efficiency.

Therefore, based on the characteristic that human eyes are sensitive to nearby objects but less perceptive of distant ones, we propose an improved update method using hierarchical updates for scene voxels on the cascade voxel texture foundation. Different update frequencies f are set according to cascade level d, calculated as shown in Equation (5). Lower cascade levels (closer to the viewpoint) have higher update frequencies, while higher levels have lower frequencies. This method reduces system node processing frequency, improves runtime throughput and efficiency, decreases computational load, and ensures nearest illumination updates occur at reasonable speeds.

When updating different cascade texture regions, if the observer moves, the cascade texture center is updated first. For the updated center, corresponding data including light buffers and geometry buffers are updated within the cascade.

For voxels at cascade region edges, data is scrollingly obtained from the next cascade level to approximate edge illumination Mipmap values [**?**]. If cascade levels have shifted, newly appeared or changed geometry at cascade voxel texture edges must be re-voxelized. This algorithm does not populate geometry buffers for dynamic objects like people or vehicles, as motion causes excessive repeated voxelization work that is detrimental to global illumination computation.

The improved scene voxel update strategy workflow based on the new structure is shown in [Figure 7: see original paper].

[Figure 7: see original paper]

During light injection, for each voxel in the obtained cascade texture, 16 fixed-direction cone filters are defined within a hemispherical range starting from its average normal vector to collect direct illumination information. For each filter, the distance l from the sample collector to the cone apex and angle β determine the sample device' s diameter. Based on radius and voxel width W_s in cascade level d, the cascade level d is calculated. Using d, r, and angle β, the voxel position is determined.

In summary, the VCT global illumination algorithm based on cascade voxel texture structure proposed in this paper consists of the following steps:

a) Construct cascade voxel textures, partition n hierarchical regions, and set update frequencies.

b) Voxelize each region separately, compute anisotropy, and store regional data in corresponding cascade voxel texture positions.

c) Determine if the view volume has changed; if yes, execute step d), otherwise execute step e).

d) Update view volume position and cascade level regions, voxelize new scene portions.

e) Render scene from light sources, compute direct illumination, and store illumination information in cascade voxel textures.

f) Use the improved cone tracing filter to collect indirect illumination, compute diffuse and specular reflections, and store illumination information in cascade voxel textures.

g) Perform final rendering, update screen, and return to step c).

---

2 Experimental Results and Analysis

To verify that our improved global illumination algorithm is suitable for real-time illumination rendering in large-scale scenes and provides substantial performance improvements, we selected both static and dynamic scenes for testing. Static scenes used Cornell boxes containing the Stanford dragon, Stanford bunny, and Stanford Happy Buddha from the Stanford public dataset. Dynamic scenes used the Sponza dynamic scene created by Frank Meinl and the Sibenic Cathedral dynamic scene created by Marko Dabrovic. We first conducted performance comparison experiments on the cascade voxel texture-based voxelization process from three aspects—voxel scale, speed, and memory consumption—to demonstrate the advantages of our proposed cascade texture structure. We then compared the speed of our improved cone tracing method with the original method at various illumination stages to prove its superiority. Finally, by comparing frame rates and memory consumption of four illumination algorithms during dynamic updates, we demonstrated that our proposed update strategy provides significant performance improvements, and through rendering effect comparisons of the four algorithms, we proved our algorithm' s strong practicality.

2.1 Performance Comparison of Voxelization Based on Cascade Voxel Texture Structure

To demonstrate the superiority of cascade voxel textures in the voxelization phase for large-scale scenes, this section compares cascade voxel textures, 3D textures, sparse octrees, the MSVO structure from literature [**?**], and the SVDAG structure mentioned in [**?**] across voxel scale, voxelization time, and storage overhead.

For more intuitive illustration of our voxel structure, we provide a simple explanation using voxel visualization comparison images before experiments. [Figure 8: see original paper] shows: (a) a scene represented using our proposed cascade voxel texture structure, clearly showing hierarchical voxel arrangement with fine voxels nearby and larger voxels covering broader areas in the distance; (b) a scene represented by sparse octree structure, where voxels are finely arranged regardless of distance, yielding good rendering results but excessive voxel scale causing high computational cost and long time consumption, making it unsuitable for real-time applications; and (c) a scene represented by 3D texture structure, which covers the entire scene and reserves storage space even where no voxels exist, causing significant space waste and low illumination computation efficiency.

[Figure 8: see original paper]

Table 1 compares voxelization scales for static scenes across different structures, while **Table 2** compares dynamic scenes. As shown in both tables, 3D textures have the largest total voxel scale at all resolutions due to lack of region differentiation. Sparse octree scene voxel scale is much lower than 3D textures but higher

than cascade voxel textures. MSVO and SVDAG provide some improvement over sparse octrees but remain unsatisfactory for large-scale scenes.

The tables reveal that at low resolutions, voxel scales between sparse octrees and cascade voxel textures are similar. However, as resolution increases, the gap widens. This is because the advantage of hierarchical scene representation in cascade textures diminishes in small scenes but becomes increasingly apparent at higher resolutions. Cascade voxel textures partition scenes hierarchically with voxel sizes increasing linearly by multiples between adjacent levels. Under different resolutions, scene representation only requires increasing level count; with fixed level count, voxelization scale remains constant. This structural characteristic enables cascade voxel textures to drastically reduce voxel count while ensuring rendering quality in large-scale scenes.

Table 1 Scale comparison of voxels in static scenes ($×10^{6}$)
| Structure | Bunny | Dragon | Buddha | |——-|—-|—-|—-| | Sparse Octree | | | | | SVDAG | | | | | Cascade Voxel Texture | 0.098 | | |

Table 2 Scale comparison of voxels in dynamic scenes ($×10^{6}$)
| Structure | Sponza | Sibenic Cathedral | |——-|—-|————-| | Sparse Octree | | | | SVDAG | | | | Cascade Voxel Texture | 4,282 | |

2.1.2 Voxelization Speed Comparison

Table 3 and **Table 4** show time overhead for different schemes. Due to the complex logical structure of sparse octrees, which are not conducive to fast construction, access, and traversal, their voxelization time exceeds that of 3D textures. Similarly, MSVO and SVDAG based on sparse octrees share this issue with slightly higher time consumption. In contrast, the hierarchical structure of cascade voxel textures facilitates traversal and lookup, substantially shortening voxelization processes and enabling smoother GPU collaboration. Our improved voxelization algorithm based on cascade voxel texture structure achieves significant speed improvements in both static and dynamic scenes.

Although our improved voxelization algorithm adds anisotropy computation for voxels, it does not perform redundant rendering and still meets temporal requirements. In summary, the cascade voxel texture scheme offers substantial advantages in voxelization speed across both small and large scenes, satisfying voxelization demands for dynamic global illumination rendering in large-scale complex scenes.

Table 3 Voxelization time comparison in static scene (ms)
| Structure | Bunny | Dragon | Buddha | |——-|—-|—-|—-| | Sparse Octree | | | | | SVDAG | | | | | Cascade Voxel Texture | 5.179 | | |

Table 4 Voxelization time comparison of dynamic scenes
| Structure | Sponza | Sibenic Cathedral | |——-|—-|————-| | Sparse Octree | | | | SVDAG | | |
| Cascade Voxel Texture | | |

2.1.3 Memory Consumption Comparison

For memory overhead, **Table 5** and **Table 6** compare different structures. Since various pixel-related information occupies similar space, to eliminate interference from multiple information types, this experiment only statistics transparency attribute space consumption as an example. Each node has one transparency attribute occupying one byte.

Our proposed cascade voxel texture is anisotropic, storing six-face information per voxel, while other methods store only one-face information. However, due to its hierarchical structure, cascade voxel texture storage overhead is significantly smaller than other methods. As resolution increases, memory consumption for 3D textures, sparse octrees, and MSVO rises exponentially. SVDAG memory consumption decreases noticeably compared to sparse octrees but still increases exponentially, causing enormous overhead in large-scale scenes. In contrast, cascade voxel textures increase linearly. For small-scale static scenes, cascade voxel texture memory overhead is approximately 1/3 of SVDAG; for large-scale dynamic scenes, it is only a few dozenths of SVDAG. Clearly, cascade voxel texture structure demonstrates excellent performance and significant advantages in large-scale scenes, substantially saving memory consumption.

Table 5 Memory consumption comparison in static scene (MB)
| Structure | Bunny | Dragon | Buddha | |———-|——-|——-|——-| | Sparse Octree | | | | | SVDAG | | | | | Cascade Voxel Texture | | | |

Table 6 Memory consumption comparison of dynamic scenes
| Structure | Sponza | Sibenic Cathedral | |———-|——-|————-| | Sparse Octree | | | | SVDAG | | | | Cascade Voxel Texture | | |

These experiments prove that our cascade voxel texture-based voxelization scheme excels in voxel scale, voxelization speed, and memory consumption. Particularly in large-scale scenes, it drastically reduces voxel count for scene representation, accelerates voxelization time, saves memory, and provides anisotropic voxel representation advantages. This indicates the structure not only reduces global illumination computation but also benefits illumination effect rendering.

2.2 Performance Comparison of Improved Cone Tracing Method at Various Illumination Stages

Cone tracing is an excellent tracking method originally pioneered in the VCT algorithm (VXGI algorithm). This section demonstrates our improved method's superiority by comparing time consumption between our improved cone tracing and VXGI' s cone tracing at various light injection and cone tracing stages.

Table 7 compares per-frame rendering time for each global illumination effect between VXGI and our improved algorithm. In dynamic scenes, VXGI requires approximately 15.5 ms for dynamic interactive updates per frame, totaling 36.7 ms for all illumination effects (average ~20 fps) and 14.2 ms without specular

reflection (average ~30 fps). Our algorithm requires only ~2.0 ms for dynamic interactive updates, totaling 26.5 ms for all effects (average ~36 fps) and 11.3 ms without specular reflection (average ~43 fps). This demonstrates that our improved cone filter performs excellently in cascade texture scenes, enabling faster computation at light injection and cone tracing stages and significantly improving global illumination computation efficiency.

Table 7 Time consumption of each illumination calculation process (ms)
| Algorithm | Dynamic Update | All Effects | No Specular | |——-|———-|———-|———-| | VXGI | 15.5 | 36.7 | 14.2 | | Our Algorithm | 2.0 | 26.5 | 11.3 |

2.3 Performance Testing of Improved Update Strategy Based on Cascade Voxel Texture Structure

Using Sponza and Sibenic Cathedral dynamic scenes, this experiment tests our algorithm in terms of frame rate and memory consumption, comparing results with LPV, PM, and VXGI global illumination algorithms to comprehensively verify conclusions.

2.3.1 Frame Rate Comparison [Figure 9: see original paper] shows frame rate comparison line charts for original VCT, LPV, PM, our algorithm without update strategy improvements, and our algorithm with improved update strategy. The chart shows significant frame rate improvement after update strategy enhancement, demonstrating the strategy' s effectiveness. Even without update improvements, our algorithm slightly outperforms VXGI. Both VXGI and our algorithm achieve real-time rendering frame rates, while LPV and PM perform poorly, maintaining only 5-20 fps with severe stuttering that may cause dizziness. Our algorithm employs more efficient cone tracing and improved voxel update strategies, achieving approximately 10 fps higher than VXGI on average. This proves our cascade voxel texture-based improved scene voxel update strategy is more suitable for large-scale scenes, enabling faster overall algorithm speed and lighter computation.

[Figure 9: see original paper]

2.3.2 Memory Consumption Comparison Our algorithm uses an improved voxelization update strategy that does not re-voxelize the entire scene (all cascade levels) during runtime. Instead, it updates different cascade regions and boundary scenes at different frequencies, further reducing memory consumption. With minimal memory overhead, it achieves substantial efficiency improvements while ensuring good illumination effects. [Figure 10: see original paper] shows memory consumption over time for VXGI, LPV, PM, our algorithm without improvements, and our algorithm with improvements. The comparison proves that the improved update strategy effectively reduces memory consumption.

The chart shows PM has the lowest memory consumption using photon maps

for global illumination, and LPV also has relatively small memory usage with its hierarchical geometry grid representation. However, both have poor speed and are unsuitable for real-time computation in complex scenes. VXGI and our algorithm voxelize scenes, resulting in higher memory consumption from voxel data. Compared to VXGI, our algorithm' s memory consumption is only 40% of VXGI' s while achieving very similar effects.

[Figure 10: see original paper]

2.4 Rendering Effect Comparison of Four Global Illumination Algorithms

Since indirect illumination rendering is far more challenging than direct illumination, rendering realism primarily depends on indirect illumination simulation. The PM algorithm performs poorly in this regard and lacks comparative significance. Therefore, using the Sponza dynamic scene, this experiment compares rendering effects of LPV, VXGI, and our proposed global illumination algorithm to comprehensively verify conclusions.

[Figure 11: see original paper] shows rendering effect comparisons among VXGI (top-left), our algorithm (top-right), LPV (bottom-left), and ray tracing (bottom-right), where ray tracing represents the closest approximation to real lighting currently achievable in global illumination and serves as the benchmark for measuring rendering quality.

The comparison reveals that VXGI and our algorithm produce nearly identical rendering effects, both closely approaching real scenes with good visual quality. Although our algorithm uses larger voxels for approximate rendering at far distances, the results remain very close to real effects even at great distances. Therefore, we conclude that our cascade voxel structure-based global illumination algorithm dramatically improves efficiency and reduces memory consumption while guaranteeing rendering quality, meeting real-time global illumination rendering requirements for large-scale complex scenes.

We conducted experiments on diffuse reflection, specular reflection, and ambient occlusion using our algorithm to demonstrate its practicality through detailed effect images.

[Figure 12: see original paper] shows illumination effects without (left) and with (right) diffuse reflection. After accumulating brightness sample values from multiple diffuse cones, both sides of pillars become clearly visible, more closely approximating real-world lighting environments.

[Figure 12: see original paper]

[Figure 13: see original paper] shows illumination effects without (left) and with (right) specular reflection. Specular reflection is achieved by tracing a single specular cone, with intensity determined by material smoothness.

These detailed demonstrations show that our algorithm' s global illumination conforms to physical laws, produces natural rendering effects with reasonable light and shadow, and achieves more realistic illumination.

[Figure 13: see original paper]

3 Conclusion

As virtual reality applications deepen in military, commercial, educational, and entertainment fields, global illumination has become a crucial component in realistic simulation, interaction, image analysis in computer vision, and image fusion in augmented reality. This paper' s contributions to global illumination technology include: First, we propose an efficient voxel storage structure for large-scale scenes—cascade voxel texture—which achieves substantial performance improvements in both time and space. Additionally, based on cascade voxel textures, we improve the original voxelization algorithm flow, making scene voxelization more efficient and anisotropic. Second, we propose an improved cone tracing filter that better adapts to cascade voxel texture structure, offering high accuracy and speed during filtering. Finally, by analyzing large-scale scene structural characteristics and human visual perception patterns, we identify a more suitable and naturally realistic voxel update scheme that significantly improves algorithm speed.

Through multiple experiments, we demonstrate that our algorithm achieves realistic rendering effects for direct illumination, diffuse reflection, specular reflection, and other lighting effects in large-scale scenes with high efficiency and strong practicality. Compared to existing algorithms, our approach solves the challenging problem of slow global illumination rendering for large-scale scenes, holding significant practical and guiding importance for technology applications across various fields.

References

[14] Crassin C, Neyret F, Sainz M, et al. Interactive indirect illumination using voxel cone tracing [C]//Proc of Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2011: 207-207.
[15] Laine S, Karras T. Efficient sparse voxel octrees. [J]. IEEE Trans on Visualization & Computer Graphics, 2011, 17 (8): 1048-1059.
[16] Hornung A, Kai M W, Bennewitz M, et al. OctoMap: an efficient probabilistic 3D mapping framework based on octrees [J]. Autonomous Robots, 2013, 34 (3): 189-206.
[17] 袁昱纬, 全吉成, 吴晨, 等. 基于稀疏体素有向无环图的光照计算加速结构 [J]. 光学学报, 2017, 37 (8): 259-272. (Yuan Yuwei, Quan Jicheng, Wu Chen, et al. Light computing accelerated structure based on sparse volume prime directed acyclic

graph [J]. Journal of Optics, 2017, 37 (8): 259-272.)
[18] 袁璐. 基于立即辐射度的实时全局光照算法 [J]. 现代计算机: 专业版, 2018 (2): 24-28. (Yuan Lu. Real-time global illumination algorithm based on immediate radiance [J]. Modern Computer:Professional, 2018 (2): 24-28.)
[1] 大野義夫. An improved illumination model for shaded display [J]. Ipsj Magazine, 1981, 22 (2): 22-26.
[2] Calazan R M. Parallel ray tracing for underwater acoustic predictions [J]. Computational Science and Its Applications, 2017, 18 (5): 53-59.
[3] Kao C C, Miao Y T, Hsu W C. A pipeline-based ray-tracing runtime system for HSA-compliant frameworks [J]. IEEE Trans on Multimedia, 2017, 19(11): 2450-2462.
[4] Shivaraju S, Pudur G. Splay thread cooperation on ray tracing as a load balancing technique in speculative parallelism and GPGPU [J]. International Arab Journal of Information Technology, 2018, 15 (1): 53-59.
[5] Jensen H W. Global illumination using photon maps[M]. Vienna: Springer, 1996: 21-30.
[6] Guzek K, Napieralski P. Efficient rendering of caustics with streamed photon mapping [J]. Bulletin of the Polish Academy of Sciences Technical Sciences, 2017, 65 (3): 361-368.
[7] Dong Zong. Pacific conference on computer graphics and applications [Z]. Hawaii:IEEE Computer Society, 2007: 77-86.
[8] Goral C M, Torrance K E, Greenberg D P, et al. Modeling the interaction of light between diffuse surfaces [J]. ACM SIGGRAPH Computer Graphics, 1984, 18(3): 213-222.
[9] De Bonet J S. Multiresolution sampling procedure for analysis and synthesis of texture images [C]// Proc of the 24th Annual Conference on Computer Graphics and Interactive Techniques.[S.l.]:ACM Press/Addison-Wesley Publishing Co, 1997: 361-368.
[10] Ritschel T, Grosch T, Kim M H. Imperfect shadow maps for efficient computation of indirect illumination [J]. ACM Trans on Graphics, 2008, 27 (5): 1-8.
[11] Hollander M, Ritschel T, Eisemann E, et al. ManyLoDs: parallel many-view level-of-detail selection for real-time global illumination [C]//Proc of Eurographics Conference on Rendering. [S.l.]:Eurographics Association, 2011: 1233-1240.
[12] Kaplanyan A. Light propagation volumes in CryEngine 3[J]. SIGGRAPH, 2009, 12 (3): 11-18.
[13] Kaplanyan A, Dachsbacher C. Cascaded light propagation volumes for real-time indirect illumination [C]//Proc of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. New York: ACM Press, 2010: 99-107.
[19] Zou Changjun, Yin Yong, Jing Qianfeng. Real-time fluid simulation with complex boundary based on slice voxelization method[C]// Proc of Asian Simulation Conference. Berlin: Springer, 2017: 319-326.
[20] Crassin C, Neyret F, Sainz. Interactive indirect illumination using voxel-based cone tracing: an insight [J]. Computer Graphics Forum, 2011, 30 (7): 1921-1930.
[21] Blessing R H. An empirical correction for absorption anisotropy [J]. Acta Crystallographica Section A Foundations of Crystallography, 2014, 51 (1): 33-

38.
[22] Martin T, Tan T S. Anti-aliasing and continuity with trapezoidal shadow maps [C]//Proc of the 15th Eurographics Workshop on Rendering Techniques. 2004: 153-160.
[23] Franke T A. Delta voxel cone tracing [C]//Proc of IEEE International Symposium on Mixed and Augmented Reality. Piscataway, NJ: IEEE Press, 2014: 39-44.
[24] Kajiya J T. The rendering equation [J]. ACM Computer Graphics, 1986, 20 (4): 143-150.
[25] Chuah S P, Cheung N M, Yuen C. Layered coding for mobile cloud gaming using scalable blinn-phong lighting [J]. IEEE Trans on Image Process, 2016, 25 (7): 3112-3125.
[26] 《中国百科大辞典》总委员会. 中国百科大辞典 (普及版) [M]. 北京: 中国大百科全书出版社, 2005. (Chairman of the General Committee of the Chinese Department of General Dictionary. China encyclopedia dictionary (popular version) [M]. Beijing. China Encyclopedia Press, 2005.)
[27] Chang H R. Energy accumulation and emanation at low latitudes, part III: forward and backward accumulation. [J]. Journal of Atmospheric Sciences, 1995, 52 (13): 2384-2403.
[28] Tanner C C, Migdal C J, Jones M T. The clipmap: a virtual mipmap [C]// Proc of the 25th annual conference on Computer graphics and interactive techniques. New York: ACM Press,, 1998: 151-158.

Note: Figure translations are in progress. See original paper for figures.