Workshop on Visualisation in Environmental Sciences (EnvirVis) (2016)
A. Middel, K. Rink, and D. Zeckzer (Editors)

Short Paper

Visualization of Scanned Cave Data with Global Illumination

Nico Schertler, Mirko Salm, Joachim Staib, Stefan Gumhold

TU Dresden, Chair of Computer Graphics and Visualization, Germany

Figure 1: Three views of our visualization of a cave data set. The data set consists of 1.4 million points and is illuminated by one red area light (see left picture), one purple area light (see right picture) and two white spot lights. Light paths with multiple reflections can be observed in all three pictures.

Abstract

3D scans of caves are acquired for virtual exploration, quantitative analysis, and communication purposes. Especially when presented to a non-professional audience, near realistic visualization is important because it improves acceptance and familiarity. A major part of realistic visualizations is the reasonable simulation of light transport, including global effects such as diffuse inter-reflections. In this paper, we present a direct visualization method for cave data that are represented as unstructured point clouds. Global light transport is approximated with a volumetric scene representation. Using hierarchical data structures, our system achieves interactive frame rates.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms

1. Introduction

Many natural caves bear important evidence on the geological formation of a region and are important economical factors as tourist attractions. Visualizations of caves, e.g. in the form of virtual flythroughs, can thus help local economy as well as give a better understanding of geology.

Present laser scanners and similar devices allow the acquisition of a cave's geometry with high precision. However, visualization of these data sets in the aforementioned context comes with a few problems. Scenes are commonly represented as point clouds, whereas traditional renderers require meshes or similar surface representations. A direct application of traditional methods is thus not possible. Furthermore, natural lighting conditions in caves are usually very dark because of their small entrances, through which light can enter the cave. This makes virtual lights in the visualization inevitable. However, simple local illumination models that only account for direct light fail to capture a reasonable light distribution due to the variety of the cave system's branching, which requires simulation of light paths with multiple reflections. Such realistic light simulation is especially important for communication purposes in order to improve acceptance by non-professionals.

However, the high computational complexity of light transport makes comprehensive simulations problematic for interactive applications. A large amount of this complexity is induced by the complicated paths that the light can take through the scene before being detected by the virtual sensor. Our renderer attempts to capture many important visual effects originating from multiple light reflections while maintaining interactive frame rates. These effects include diffuse inter-reflections and color bleeding.

In this paper, we present a method that directly visualizes cave geometry data represented as point clouds with approximate global illumination and interactive frame rates. We use the work of Crassin et al. [CNS*11] as the basis for our approach. The original algorithm is modified to support a large amount of progressively built light bounces as well as to reduce memory consumption and to simplify the overall approach.

Our cave data set is acquired through multiple LiDAR (light detection and ranging) scans that are registered and combined to a single point cloud, where each point carries its position, normal, and color.

2. Related Work

While reasonable global illumination can be simulated by offline renderers, interactive methods can only approximate light distribution. In the following, we give a brief overview of the most relevant techniques.

Instant Radiosity [Kel97] distributes virtual point lights (VPLs) into the scene by ray casting from the primary light sources over multiple bounces. The large number of costly VPLs makes Instant Radiosity not directly applicable to interactive scenes. Furthermore, while Instant Radiosity is well suited to simulate diffuse light propagation, support for glossy materials is limited since strongly focused reflections expose the VPLs to the viewer. The derived algorithms described in the following inherit this limitation.

Laine et al. [LSK*07] propose an incremental real-time capable approach for semi-static scenes and a single bounce of indirect light where only a subset of VPLs has to be updated per frame. Ritschel et al. [RGK*08] introduce *Imperfect Shadow Maps* (ISMs) to tackle the issue of the costly shadow map rendering used in VPL-based approaches. Instead of rasterizing continuous geometry, ISMs employ point based rendering to accelerate shadow map generation, utilizing a sparse, splat-based scene approximation. A hole filling approach is applied that attempts to partially recover continuous depth values. An aspect worth considering is that while ISMs map naturally to point-based rendering, point clouds usually contain much more points than would be feasible to render for each ISM. Therefore the use of an importance sampled scene description can in general not be avoided. In the context of their point cloud-based global illumination approach, Preiner et al. [PW10] propose to use the original set of points for ISM rendering but nonetheless render every scene sample only once for a single randomly selected VPL.

Cascaded Light Propagation Volumes (LPVs) [KD10] build a dense volumetric representation of the scene around the observer, where each voxel can emit light on its own. In every frame, direct lighting in the scene is injected into a subset of these voxels and propagated through the volume by an iterative diffusion-like process. Crassin et al. [CNS*11] propose to discretize the scene in a sparse, hierarchical voxel grid that allows to continuously query scene information at different levels of precision while avoiding to waste large amounts of memory for empty voxels as is typically the case with dense grids. During the rendering phase, *voxel cone tracing* (VCT), which traverses a ray across different levels in this hierarchy, is used to estimate the incident radiance at each fragment. McLaren [McL14] also proposes to use VCT to simulate the light propagation but turns back to nested, dense grids. Compared to Crassin et al., the scheme is further simplified by storing diffusely reflected radiance at each voxel instead of directional distributions of incoming radiance.

3. Algorithm

In the following section, we outline our rendering algorithm. It supports dynamic point lights as well as area lights and simulates light paths with multiple diffuse reflections. Direct light can be modeled with any (also non-diffuse) BRDF. The final rendering step can also be modified to allow light paths from a light source over several diffuse reflections and one specular reflection to the observer. Our renderer is primarily based on the work of Crassin et al. [CNS*11, Cra11] but also incorporates some ideas of McLaren [McL14] to simplify the approach and to reduce memory consumption.

The algorithm runs primarily on the GPU and requires a pre-processing step, in which the scene is converted into a volumetric representation. We use a Sparse Voxel Octree (SVO) to describe the scene hierarchically. Nodes that are deeper in the tree represent smaller parts of the scene but exhibit higher accuracy. This SVO represents both scene geometry and the distribution of light in the scene. After this pre-processing step, each frame requires two phases: The first phase approximates light transport in the scene. New light emitted by light sources is injected into the SVO representation and reflected light is evaluated through Voxel Cone Tracing (VCT) and stored in the SVO. In the second phase, the scene is rendered by fetching global lighting information from the SVO and combining it with analytically evaluated direct lighting.

3.1. Sparse Voxel Octree Representation

The pre-processing step requires the scene to be represented as a set of scene samples. Scene samples are points that are characterized by their geometry attributes, consisting of position, opacity, RGB albedo, and normal distribution function (NDF), which encodes the distribution of normals in the area represented by the sample [Fou92]. The NDF is represented by a 3D vector, whose direction corresponds to the NDF's mean and whose length corresponds to its variance [Tok04]. Additionally, scene samples can carry a radiance if they are part of an area light. The input point cloud can be used directly as scene samples.

The octree is stored in a linear GPU buffer as an index-based tree. To construct the tree, every scene sample induces a traversal to its according leaf node. All leaf nodes are on the lowest octree level and can be identified with their integer Morton Index [Mor66]. During this traversal, non-existent nodes are created. Leaf nodes are characterized by the same attributes as scene samples. Once a traversal reaches a leaf node, the according sample's geometry attributes are splatted additively into the node.

3.2. Light Injection and Propagation

At the beginning of the ligth injection and propagation phase, the leaf voxels of the SVO contain a volumetric representation of the

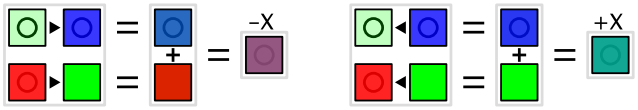

Figure 2: 2D depiction of the anisotropic filtering scheme. Four isotropic child voxels (eight in 3D) are blended together pairwisely along the principal axes (only -X and +X are shown). For each axis, the resulting two values (four in 3D) are subsequently averaged which accounts for the spatial pre-integration. The visibility of the circles illustrates the opacity values.

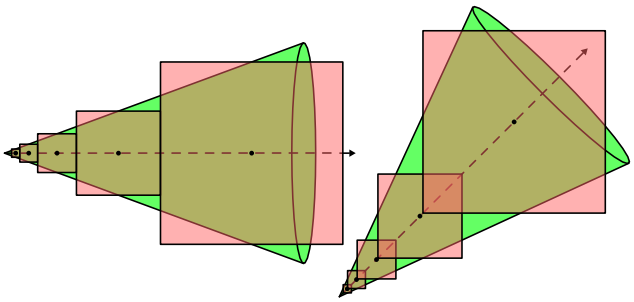

Figure 3: The VCT sampling scheme. The footprint of the cone (green) is approximated by samples (black dots) of different resolutions taken along the cone's main axis (dashed arrow). The cone is partially under- as well as overestimated by the footprints of the samples (red squares). The approximation error also depends on the direction of the axis.

entire scene. This phase simulates light transport in the scene by injecting new light into the leaf voxels and propagating it through the SVO, including inner nodes. Therefore, the inner nodes of the SVO do not carry geometry attributes but only opacity and radiance values, where radiance represents the amount of light that a voxel emits on its own plus the amount it reflects from incoming light. Radiance is stored anisotropically, i.e. a separate value exists for each of the principal axes. These values correspond to the radiance emitted in the according directions. This representation allows a directional pre-integration in the subsequent mipmapping step. The radiance values in the SVO are preserved across subsequent frames. This allows to simulate additional light bounces as more frames are computed.

The process of light injection depends on the type of the light source. Area lights are represented by scene samples with a non-zero radiance. Therefore, they are already part of the SVO and do not need to be injected separately. Point lights are evaluated analytically at every leaf node with a local diffuse illumination model and a shadow map and then splatted additively into the leaf voxels.

The anisotropic radiance information at the leaf voxels is then mipmapped onto higher levels of the octree, such that nodes that are closer to the root contain a coarser approximation of their child nodes' light distribution. The mipmapping scheme of anisotropic radiance values uses alpha-blending with pre-multiplied values [PD84] in the according direction, followed by averaging the resulting four values (see figure 2).

The hierarchical representation of emitted and reflected light is then used to propagate radiance to other leaf voxels. This is realized as a gathering process for each leaf voxel. The incoming radiance is then added to the voxel's existing radiance. In a physically plausible model, the irradiance at a surface can be computed from the integral of cosine-weighted incoming radiance over the surface hemisphere, characterized by its normal (note that leaf voxels are isotropic). We use Voxel Cone Tracing (VCT, see figure 3) to approximate this integral. The hemisphere around the NDF's mean is divided into a small number of conical sections (usually four to ten). Then, the cone's principal axis is traversed and radiance and opacity values are queried from the SVO. These values are used in a front-to-back compositing scheme [Max95] to approximate the irradiance in this cone. The cosine-weighted sum of irradiance from all cones serves as an approximation of the integral over the entire hemisphere. In order to query radiance information from anisotropic voxels, the values of the three sides that face the query direction are blended together based on the dot product of their normal in the query direction. While ray samples near the ray's origin use finer levels of the SVO, samples that are farther away use coarser levels. In order to sample the SVO continuously, two trilinear samples are first calculated from the two closest mipmap levels and then blended as a quadrilinear sample.

After this process, every leaf voxel contains radiance information that represents how much light the voxel emits and reflects diffusely in every direction. This information is used by the final rendering step.

3.3. Final Rendering

The final rendering of the scene uses deferred shading [ST90] to prevent an overhead due to shading computations performed for occluded fragments and to reduce the amount of shader permutations. In a first pass, the scene is rendered to a G-buffer that includes the samples' geometry attributes. We use spherical surfels as a surficial representation of the input points. The second pass computes the actual shading for every screen pixel based on the attributes in the G-buffer, the light sources, and the reflected radiance stored in the leaf voxels of the SVO. During this shading pass, the flux resulting from direct lighting of point lights is evaluated analytically in a first step. Then, the reflected radiance stored in the leaf voxels of the SVO is sampled trilinearly or tricubically at the fragment positions and added to the flux. Additionally, it is possible to compute a bounce of specular reflections from previously diffusely propagated light by sending a single cone starting from the fragment's position along the reflection vector into the SVO. This can be used to visualize wet cave walls that exhibit more focused specular reflections. To cheaply simulate the integration over time, which converts the flux of each screen pixel to energy, the flux is multiplied with an empirical exposure duration. The resulting energy is finally *tonemapped* and *gamma-encoded* [TFCRS11] to map more values into a displayable range and to account for the conversion from RGB space to sRGB space, respectively.

(a) (b) (c) (d)

Figure 4: Sample views of our cave data set

fig. 4c	800 x 600 px	1600 x 900 px
SVO depth 7	32 fps	22 fps
SVO depth 8	24 fps	18 fps

fig. 4d	800 x 600 px	1600 x 900 px
SVO depth 7	30 fps	20 fps
SVO depth 8	22 fps	16 fps

Table 1: Rendering performance on an NVIDIA GeForce GTX 560 Ti

4. Results

Our renderer simulates plausible light distribution inside caves. Figure 4 shows four more sample views of our data set. The importance of light paths with multiple reflections becomes evident in all four pictures, especially in figure 4b. If only direct light was considered, this view would result in a black image because neither of the four light sources have an unblocked line of sight to this part of the scene.

As shown in table 1, our method achieves interactive frame rates. More than half of a frame's rendering time is spent on the g-pass, which renders geometry attributes to the g-buffer without calculating illumination. More involved selection strategies of rendered points and caching strategies could additionally enhance the resulting frame rate significantly.

Our implementation uses a single octree, in which light propagation is simulated. In order to support big cave systems, multiple SVOs may be used and propagation may be calculated only in the relevant ones.

5. Conclusion

In this paper, we presented a method for direct visualization of cave data sets with global illumination that achieves interactive frame rates. This makes it appropriate for applications, in which the user can control the rendering, e.g. for individual virtual exploration of a cave system. The realistic light simulation allows our system to be used for expedition planning because the virtual and real cave have similar appearances, enhancing orientation.

Although our implementation supports only a limited extent of the cave, we presented ideas how big cave systems can be visualized.

In summary, our system solved all problems stated in section 1 and is well-suited for communication purposes. Furthermore, our implementation does not require high-end hardware and can be used on common mid-range consumer computers.

References

[CNS*11] CRASSIN C., NEYRET F., SAINZ M., GREEN S., EISEMANN E.: Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum (Proceedings of Pacific Graphics 2011) 30*, 7 (sep 2011). 2

[Cra11] CRASSIN C.: *GigaVoxels: A Voxel-Based Rendering Pipeline For Efficient Exploration Of Large And Detailed Scenes*. PhD thesis, UNIVERSITE DE GRENOBLE, July 2011. 2

[Fou92] FOURNIER A.: Normal distribution functions and multiple surfaces. In *Graphics Interface '92 Workshop on Local Illumination* (Vancouver, BC, Canada, 1992), pp. 45–52. 2

[KD10] KAPLANYAN A., DACHSBACHER C.: Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2010), I3D '10, ACM, pp. 99–107. doi:10.1145/1730804.1730821. 2

[Kel97] KELLER A.: Instant radiosity. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 49–56. doi:10.1145/258734.258769. 2

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007* (2007), Eurographics Association, pp. 277–286. 2

[Max95] MAX N.: Optical models for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics 1*, 2 (June 1995), 99–108. doi:10.1109/2945.468400. 3

[McL14] MCLAREN J.: Cascaded voxel cone tracing in the tomorrow children. http://fumufumu.q-games.com/archives/2014_09.php, 2014. 2

[Mor66] MORTON G.: *A Computer Oriented Geodetic Data Base and a New Technique in File Sequencing*. International Business Machines Company, 1966. 2

[PD84] PORTER T., DUFF T.: Compositing digital images. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1984), SIGGRAPH '84, ACM, pp. 253–259. doi:10.1145/800031.808606. 3

[PW10] PREINER R., WIMMER M.: Real-time global illumination for point cloud scenes. *Computer Graphics & Geometry 12*, 1 (2010), 2–16. 2

[RGK*08] RITSCHEL T., GROSCH T., KIM M. H., SEIDEL H.-P., DACHSBACHER C., KAUTZ J.: Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 129:1–129:8. doi:10.1145/1409060.1409082. 2

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 197–206. doi:10.1145/97879.97901. 3

[TFCRS11] THOMPSON W., FLEMING R., CREEM-REGEHR S., STEFANUCCI J. K.: *Visual Perception from a Computer Graphics Perspective*, 1st ed. A. K. Peters, Ltd., Natick, MA, USA, 2011. 3

[Tok04] TOKSVIG M.: Mipmapping normal maps. http://www.nvidia.com/object/mipmapping_normal_maps.html, 2004. 2