Relighting Forest Ecosystems

Jay E. Steele* and Robert Geist**

Clemson University

Abstract. Real-time cinematic relighting of large, forest ecosystems remains a challenging problem, in that important global illumination effects, such as leaf transparency and inter-object light scattering, are difficult to capture, given tight timing constraints and scenes that typically contain hundreds of millions of primitives. A solution that is based on a lattice-Boltzmann method is suggested. Reflectance, transmittance, and absorptance parameters are taken from measurements of real plants and integrated into a parameterized, dynamic global illumination model. When the model is combined with fast shadow rays, traced on a GPU, near real-time cinematic relighting is achievable for forest scenes containing hundreds of millions of polygons.

1 Introduction

Rendering large-scale, forest ecosystems is a topic of wide interest, with applications ranging from feature film production to environmental monitoring and ecosystem management. Important global illumination effects, such as leaf transparency and inter-object scattering, are difficult to capture, given tight timing constraints and models that potentially contain hundreds of millions of primitives.

Early approaches to rendering such systems relied on conventional rasterization using billboard clouds [1]. More recently, focus has shifted to geometry-based techniques that rely on ray-tracing. Ray-tracing generally provides superior visual results, and new acceleration structures combined with low-cost, highly parallel execution environments can make ray-tracing competitive in execution time. In this vein, Geist et al. [2] extended the approach of Dietrich et al. [3] to include diffuse leaf transparency and inter-object scattering while maintaining near real-time rendering for scenes that comprised hundreds of millions of polygons. They used a lattice-Boltzmann (LB) photon transport model [4] to solve for global illumination in a pre-processing step and a highly parallel ray tracer, built on NVIDIA's Compute Unified Device Architecture (CUDA) [5], to achieve acceptable execution time.

Nevertheless, as noted in [2], there remain drawbacks to this work. First, although the LB lighting solution captured the desired global effects, it was constrained by the use of geometric instancing. The scenes of [2] each show approximately 1,000 trees, but there are only 5 distinct tree geometries. The LB transport model was solved only once per geometry. As a result, the common method

* e-mail: jesteel@cs.clemson.edu

** e-mail: geist@cs.clemson.edu

of achieving desired forest scene variability, random instance rotation, yields an incorrect lighting solution for any rotated tree. Second, relighting a scene (for example, in response to the movement of the sun) required re-computing the full LB lighting solution for each tree model based on the current sun direction, which, for near real-time frame rates and complex scenes, is not feasible with today's hardware. Finally, the LB lighting solution for each tree model was computed assuming there was no external occlusion of indirect illumination, which is not valid for large forest scenes.

The principal goal of this effort is to extend the approach of [2] to obtain a compact, parameterized lighting solution that is accurate for any light source position and intensity. This parameterized solution allows both arbitrary rotation and translation of geometric instances as well as dynamic global illumination, i.e., real-time movement of the sun.

2 Background

2.1 Related Work

Dorsy et al. [6] achieved beautiful results in rendering small collections of plant leaves using carefully constructed bidirectional reflectance and transmittance functions that were based on measurements of real plants. Their method is computationally intensive, with large memory requirements, and so as yet unsuitable for real-time rendering of large-scale, high-density ecosystems.

Reeves et al. [7] represented trees as a particle system. A probabilistic shading model shaded each particle based on the particle's position and orientation. Hegeman et al. [8] ignore physical accuracy in a technique that attempts to achieve visual plausibility and fast computation through approximating ambient occlusion. Trees are approximated by bounding spheres containing randomly distributed small occluders. Fragments are shaded based on the average probability of the fragment being occluded given its position within the bounding sphere. Though simple to compute, this method, as mentioned by the authors, only considers local information, and results can differ widely from more physically accurate approaches. Luft et al. [9] were able to capture ambient occlusion in rendering foliage through the addition of a preprocessing step in which overall tree geometry was simplified to an implicit surface, i.e., a density field, using Marching Cubes [10]. The ambient coefficient in a standard, local illumination model was then modified by a transfer function that was exponentially decreasing in the field density. They also realigned leaf normal vectors to match the implicit surface in order to reduce lighting noise.

The overall technique presented in this study is similar, in spirit, to both precomputed radiance transfer of Sloan et al. [11] and photon mapping of Jensen et al. [12], in that a preprocessing step is used to compute and store lighting information within the scene itself. Comparatively, the LB lighting preprocessing step is very fast. Hašan et al. [13] introduced the direct-to-indirect transfer for cinematic relighting, which requires a long preprocessing step and does not provide full anisotropic scattering.

2.2 Lattice-Boltzmann Lighting

Our goal, cinematic relighting, requires fast global illumination of forest ecosystems. Geist et al. [4] describe a lighting technique that was adapted in [2] to provide static forest illumination. The original lighting technique offered a new solution to the standard volume radiative transfer equation for modeling light in a participating medium:

$$(\boldsymbol{\omega} \cdot \nabla + \sigma_t)\, L(\boldsymbol{x}, \boldsymbol{\omega}) = \sigma_s \int p(\boldsymbol{\omega}, \boldsymbol{\omega'}) L(\boldsymbol{x}, \boldsymbol{\omega'}) d\boldsymbol{\omega'} + Q(\boldsymbol{x}, \boldsymbol{\omega}) \tag{1}$$

where L denotes radiance, $\boldsymbol{\omega}$ is spherical direction, $p(\boldsymbol{\omega}, \boldsymbol{\omega'})$ is the phase function, σ_s is the scattering coefficient of the medium, σ_a is the absorption coefficient of the medium, $\sigma_t = \sigma_s + \sigma_a$, and $Q(\boldsymbol{x}, \boldsymbol{\omega})$ is the emissive field in the volume [14]. The solution, which is applicable to simulating photon transport through participating media such as clouds, smoke, or haze, was based on a lattice-Boltzmann (LB) method. At each iteration, photons scatter from a lattice site to its neighboring sites. Neighboring sites are defined by a standard approach, due to d'Humières, Lallemand, and Frisch [15], that uses 24 points, equidistant from the origin in 4D space, projected onto 3D. The points are:

$$(\pm 1, 0, 0, \pm 1)\ (0, \pm 1, \pm 1, 0)\ (0, \pm 1, 0, \pm 1)$$
$$(\pm 1, 0, \pm 1, 0)\ (0, 0, \pm 1, \pm 1)\ (\pm 1, \pm 1, 0, 0)$$

and projection is truncation of the fourth component, which yields 18 directions. Axial directions then receive double weights, thus ensuring isotropic flow. A final direction, a rest direction that points from a lattice site to itself, is added to facilitate the representation of several phenomena, including energy absorption and energy transmission. This gives 19 directions, $\boldsymbol{c}_m$, $m \in \{0, 1, ..., 18\}$. Each lattice site contains a per-node photon density, $f_m(\boldsymbol{r}, t)$, which is the density arriving at lattice site $\boldsymbol{r} \in \Re^3$ at time t in direction $\boldsymbol{c}_m$. Given a lattice spacing λ and a time step τ, the simulation amounts to a synchronous update:

$$f_m(\boldsymbol{r} + \lambda \boldsymbol{c}_m, t + \tau) - f_m(\boldsymbol{r}, t) = \Omega_m \cdot f(\boldsymbol{r}, t) \tag{2}$$

where Ω_m denotes row m of a 19×19 matrix, Ω, that describes scattering, absorption, and wavelength shift at each site.

For any LB method, the choice of Ω is not unique. In [4], for the case of isotropic scattering, Ω was chosen as follows:
For row 0:

$$\Omega_{0j} = \begin{cases} -1 & j = 0 \\ \sigma_a & j > 0 \end{cases} \tag{3}$$

For the axial rows, $i = 1, ..., 6$:

$$\Omega_{ij} = \begin{cases} 1/12 & j = 0 \\ \sigma_s/12 & j > 0, j \neq i \\ -\sigma_t + \sigma_s/12, & j = i \end{cases} \tag{4}$$

For the non-axial rows, $i = 7, ..., 18$:

$$\Omega_{ij} = \begin{cases} 1/24 & j = 0 \\ \sigma_s/24 & j > 0, j \neq i \\ -\sigma_t + \sigma_s/24, & j = i \end{cases} \tag{5}$$

Entry i, j controls scattering from direction $\boldsymbol{c_j}$ into direction $\boldsymbol{c_i}$, and directional density f_0 holds the absorption/emission component. Note that the axial rows, $i = 1, ..., 6$, are weighted twice as much as the non-axial rows, $i = 7, ..., 18$. The entries of Ω were then multiplied by the density of the medium at each lattice site, so that a zero density yielded a pass-through in (2), and a density of 1 yielded a full scattering.

In terms of total site photon density, $\rho = \sum_{i=0}^{18} f_i(\boldsymbol{r}, t)$, the synchronous update (2) was shown, in the limit $(\lambda, \tau \to 0)$, to yield a diffusion equation

$$\frac{\partial \rho}{\partial t} = \left(\frac{\lambda^2}{\tau}\right) \left[\frac{(2/\sigma_t) - 1}{4(1 + \sigma_a)}\right] \nabla^2_{\boldsymbol{r}} \rho \tag{6}$$

Previous approaches ([16, 17]) have demonstrated that multiple photon scattering events invariably lead to diffusion processes.

In [2], this model was extended to provide global illumination effects for forests scenes. Wavelength-dependent, anisotropic scattering was achieved through modifying Ω. Each σ_s that appears in entry $\Omega_{i,j}$ was multiplied by a normalized phase function:

$$pn_{i,j}(g) = \frac{p_{i,j}(g)}{\left(\sum_{i=1}^{6} 2p_{i,j}(g) + \sum_{i=7}^{18} p_{i,j}(g)\right)/24} \tag{7}$$

where $p_{i,j}(g)$ is a discrete version of the Henyey-Greenstein phase function [18],

$$p_{i,j}(g) = \frac{1 - g^2}{(1 - 2g\boldsymbol{n_i} \cdot \boldsymbol{n_j} + g^2)^{3/2}} \tag{8}$$

Here $\boldsymbol{n_i}$ is the normalized direction, $\boldsymbol{c_i}$. Parameter $g \in [-1, 1]$ controls scattering direction. Value $g > 0$ provides forward scattering, $g < 0$ provides backward scattering, and $g = 0$ yields isotropic scattering. For each primary wavelength, g was computed from leaf characteristics (transmission and reflectance) of real plants measured by Knapp and Carter [19], to provide accurate scattering.

3 Cinematic Relighting

Our goal is to allow cinematic relighting, that is real-time rendering, of complex forest scenes with dynamic, user-controllable light sources, while capturing global illumination effects such as scattering, transmission, and absorption. Our target scenes consist of one infinite point light (the sun) and ambient light due to reflection and scattering of the sky, ground, and forest. (Supporting multiple

infinite point lights is a trivial extension.) Due to the large memory requirements of plant models, individual plants in a forest are rotated and translated instances of a small set of shared plant models. Thus, a typical scene may contain hundreds of individual plants, but less than ten unique plant models.

First, we detail a new technique that utilizes the previously described static LB lighting model to allow for dynamic, global illumination, based on the current, run-time sun position and intensity. Then, we introduce a coarse-grained illumination model that accounts for global scene effects, such as plant-plant occlusion, that the fine-grained illumination model does not account for by itself.

3.1 Dynamic Lighting

In the technique of [2], plant instances that reference the same plant model, but have different orientations relative to a given sun direction, require individual lighting solutions, as a lighting solution is only valid for one orientation. Storing a solution per plant instance would quickly exhaust the available memory of today's hardware. Computing a solution per frame for each plant instance would exceed the real-time capability of today's hardware. We now describe a technique that removes these restrictions, thus enabling dynamic global illumination for scenes containing many plant models while allowing both translated and rotated instances. The approach is conceptually simple. A set of base lighting solutions is precomputed per plant model, not per instance. These base lighting solutions are combined, at run-time, based on each instance's orientation relative to the current sun direction, thus allowing dynamic lighting updates to complex scenes containing thousands of plants in real-time.

First, for each plant model (not instance) we precompute 19 base lighting solutions $\{B_j | j = 0, 1, ..., 18\}$ by using (2) with boundary conditions based on direction index j. For solution B_j with $j > 0$, all boundary nodes have fixed densities $f_i(\boldsymbol{r}, t) = \delta_{ij}$ (Kronecker delta), all i. For solution B_0, the ambient solution, all boundary nodes have $f_i(\boldsymbol{r}, t) = 1$, all i.

At run-time, we employ a shader that computes the lighting for each fragment by combining multiple, weighted base solutions based on the sun direction, transformed into each instance's local coordinate space. The selection of the base solutions and weights is geometrically straightforward. Consider a convex polyhedron, consisting of 32 faces (triangles), formed by adjacent triples of unit length directions, $\boldsymbol{c}_i / \|\boldsymbol{c}_i\|, i = 1, ..., 18$. A ray from the origin in the sun direction will intersect one triangle. If the vertices of that triangle have associated directions $\boldsymbol{c}_{i_0}$, $\boldsymbol{c}_{i_1}$, and $\boldsymbol{c}_{i_2}$, then we use base solutions B_{i_0}, B_{i_1}, and B_{i_2}, with weights determined by the barycentric coordinates of the intersection point. The weights sum to one, thus conserving total energy.

At run-time, for each fragment, we sample the three base solutions for that fragment's instance and linearly combine them using the weights. Ambient light is added by sampling from the ambient base solution B_0, which has scaling controllable by the user. As described, dynamic lighting requires at most 32 ray-triangle intersections per plant instance and four 3D texture lookups per fragment.

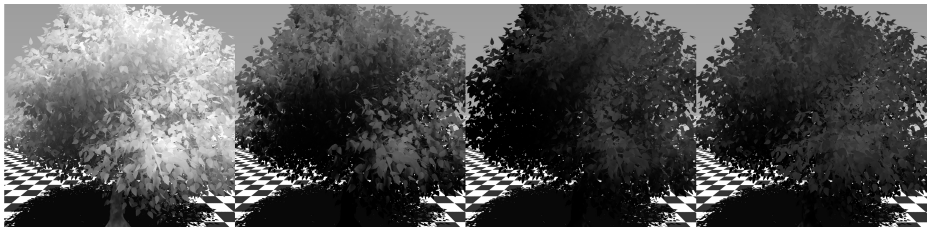


Fig. 1. Final LB lighting solution and three base LB lighting solutions.

Figure 1 illustrates the dynamic lighting technique. The left image visualizes the final LB lighting, which is computed by combining the three base LB lighting solutions that appear in the remaining three images. (Ambient light is also used, but not shown.) The base LB lighting solutions are selected and weights are computed based on the plant instance's orientation relative to the current sun position.

3.2 Hierarchical Lighting Model

Our LB lighting method, as described to this point, computes indirect illumination effects under the assumption that global illumination of the plant is not occluded by other plants. (Occlusion of direct illumination is handled by shadow rays, as described in the next section.) This can lead to too much light energy at the boundary nodes, resulting in too much indirect illumination for occluded plants. We now describe an extension that accounts for such inter-object (plant-plant) occlusions.

We combine our fine-grained, local LB lighting solution with a coarse-grained, global LB lighting solution. A coarse-grained, global LB lighting grid can be imposed upon an instanced forest system. Each node in the coarse-grained grid has a density factor estimated from the tree instances that intersect the node. Solution of this coarse-grained grid over the entire forest, using iterations of (2), simply provides scale factors that weight the indirect illumination of a fragment computed from the fine-grained, local LB lighting solution.

As with the fine-grained, local LB solution, a coarse-grained, global LB solution can be precomputed for a scene. The same technique described in Section 3.1 can be applied to the coarse-grained, global LB lighting solution to allow dynamic updates in real-time as the sun traverses the sky.

4 Implementation

The preprocessing steps of sections 3.1 and 3.2 are implemented in CUDA. We precompute the fine-grained LB lighting separately for the three primary wavelengths for each lattice direction. We only precompute lighting intensity (luminance) for the coarse-grained LB lighting.

Visible fragments are generated through ray-tracing with CUDA, though rasterization is a viable alternative. A perspective grid (see Hunt and Mark [20]) is used to accelerate primary ray/instance intersection testing. For the scenes provided here, we compute local diffuse lighting of the form $k_d(\boldsymbol{n}\cdot\boldsymbol{l})$, where $\boldsymbol{l}$ is sun direction, $\boldsymbol{n}$ is the surface normal, and k_d is the combined sun color and sampled texture color. Specular lighting is of the form $k_s(\boldsymbol{v}\cdot\boldsymbol{r})^s$, where $\boldsymbol{v}$ is the viewer position vector, $\boldsymbol{r}$ is the sun reflection vector, s controls highlight dissipation, and k_s is the combined sun color and leaf/wood/water specular color. Our LB lighting, such as that shown in the left image of Figure 1, is modulated by texture color and added to the local, direct illumination, to produce final fragment color.

4.1 Shadow Rays

To accelerate shadow rays (the dynamic, global illumination technique does not provide high frequency shadows), we employ a sun-aligned grid, which is similar in spirit to the perspective grids of Hunt and Mark [20] and is a two-dimensional data structure, consisting of multiple tiles arranged as a grid on a plane whose normal is parallel to the sun direction. Each tile contains a list of those model instances that intersect that tile in sun-space, which is an orthographic projection of the scene translated so that the sun is located at infinity on the z-axis and the direction of sunlight is the negative z-axis. For each plant instance in our scene, the associated model's axis-aligned bounding box (AABB) is projected into world-space, followed by a projection into sun-space. The sun-aligned grid is recomputed as the sun moves.

Once a shadow ray intersects an instance's AABB, we transform the shadow ray to model space and test for occlusion with the model's geometry by traversing a uniform grid. We found that traversing uniform grids, as described by Lagae and Dutré [21], offered better performance than multiple kd-tree traversal algorithms when traversing aggregate models such as trees.

4.2 Compression

The LB lighting model produces volumetric lighting data, which quickly consumes large amounts of memory per plant model. Each node value requires 3 floats (1 for each primary wavelength). For each plant model, 19 base solutions on a 128^3 lattice requires $(2M\times19\times3\times4)$ 456MB of LB lighting data. Our implementation supports compression of this data through the use of Haar wavelets. Several authors have demonstrated that Haar wavelets can significantly compress volumetric data while supporting fast, random access to individual voxels. As noted by Westermann [22], other wavelet transforms may provide better compression rates, but, the simplicity of Haar wavelets results in faster reconstruction times, which is vital to our application.

To avoid visible discontinuities that result from nearest-neighbor sampling, we use linear filtering when sampling from the LB lighting. Thus, we need fast, random access to each 2^3 block of texels. We compress sub-blocks of size 3^3, but

include the forward neighbors of all nodes in a sub-block. Our final sub-block size is thus 4^3, for which 37 of 64 values stored are redundant. Although this reduces our compression rate, the redundancy significantly improves rendering times by reducing the amount of work required to access those forward neighbors of a texel needed when applying linear filtering. We compress by decimating coefficients below a user supplied threshold. As in the work of Bajaj et al. [23], we do not store data that will not be accessed.

5 Results

All results are reported for lattices of size 128^3. Execution time for the dynamic LB lighting preprocessing step, described in Section 3.1, (computing 19 base solutions, three wavelengths each, on a GTX 280) averages 3 minutes 20 seconds for multiple plants, which includes computation of σ_s and σ_a from the model data. Each wavelength for each base solution averages 2.926 seconds. The number of iterations required to achieve convergence to steady-state is approximately twice the longest edge dimension of the grid.

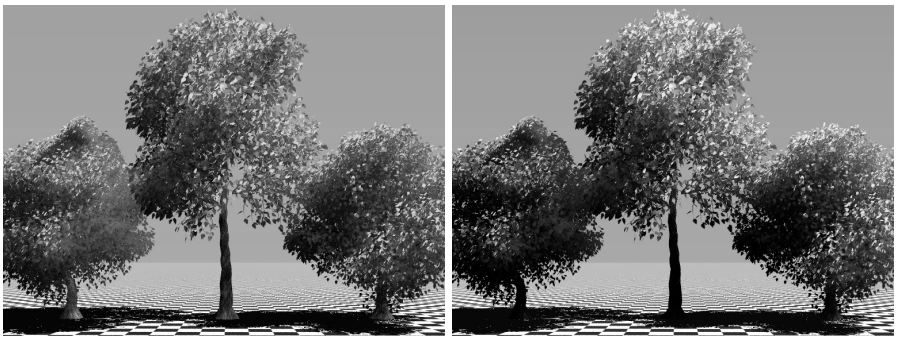
The effect of applying the hierarchical method described in Section 3.2 is shown in Figure 2. In this scene, the sun is located towards the upper right of the image. The two smaller tree instances both reference the same tree model and thus have the same precomputed LB lighting solution, although each accounts for instance translation and rotation. The image on the left combines direct illumination with the fine-grained indirect illumination from LB lighting, while the image on the right combines direct illumination with the fine-grained indirect illumination from LB lighting and the coarse-grained indirect illumination. Note that in the image on the left, the tree on the left receives too much light for the position of the sun, while in the image on the right, the same tree is appropriately darkened.

Figure 3 shows multiple frames captured from two scenes as a user moves the sun from left to right. The top scene, consisting of a single Catalpa tree, contains 316,727 triangles. The bottom scene, a canopy view of multiple plants, contains 109 million triangles. Execution times for relighting each scene in Figure 3 at a resolution of 512×512 pixels with 1 ray per pixel and 4 rays per pixel are shown in Table 1, for one, two, and all four GPUs of a Tesla S1070. Relighting time includes the time required to compute indirect illumination by sampling and combining four base solutions (including decompressing) and the time to compute direct shadows with shadow rays. Table 1 shows the average execution time for 24 sun positions, ranging 90 degrees about the zenith.

Without compression, the precomputed LB lighting solution for the canopy scene of Figure 3 would require 2.28 GB (456 MB per model, 5 models). With compression, the total LB lighting for the scene is 366 MB. The root mean square error between renders of the canopy scene with compression and without compression is 0.00843. Finally, a high fidelity scene rendered with these techniques is shown in Figure 4.

Table 1. Execution times for relighting, Tesla S1070

Catalapa			Canopy		
Number of GPUs	1 ray/pixel	4 rays/pixel	Number of GPUs	1 ray/pixel	4 rays/pixel
1	0.072 s	0.258 s	1	0.259 s	0.858 s
2	0.040 s	0.150 s	2	0.132 s	0.432 s
4	0.023 s	0.096 s	4	0.073 s	0.247 s



Local LB only Local and global LB

Fig. 2. Rendering comparison: Local LB vs. local and global LB.

6 Conclusion

Incorporating global illumination effects, such as leaf transparency and inter-object reflection, in real-time rendering of large, forest ecosystems is a challenging problem. One approach to achieving such effects, suggested herein, is through extending a lattice-Boltzmann lighting model that approximates indirect illumination to allow for dynamic sampling at run-time.

There are several drawbacks to the original LB lighting technique [2, 4] that have been addressed in this work. First, we use a preprocessing step that allows dynamic, run-time updates of positions of lights at an infinite distance, such as the sun. This also allows plant instances that reference the same plant model to share the same LB lighting solution, regardless of each instance's orientation. Second, we employ a two-stage hierarchy to capture occlusion of global, indirect illumination. Third, we have shown that compressing the volumetric, LB lighting data using Haar wavelets is viable, in that relatively high compression rates can be achieved while maintaining final image quality and rendering performance.

Cinematic relighting is achieved by combining local, direct illumination at any visible fragment point with an indirect value obtained by interpolating values from a set of preprocessed LB lighting solutions. Shadowing, a vital component of any direct illumination technique, is computed by shadow rays that traverse a sun-aligned grid. Near real-time performance is obtained by mapping the ray-

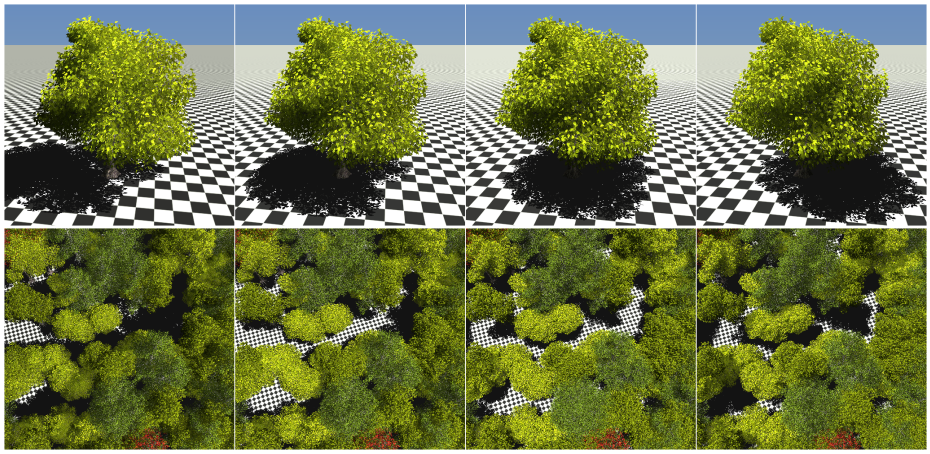


Fig. 3. Relighting scenes. Top: Catalapa. Bottom: Canopy

Fig. 4. River scene.

tracing engine, as well as the LB lighting model, to NVIDIA's Compute Unified Device Architecture and then distributing across multiple GPUs.

Future work will focus on speeding visibility computations necessary for cinematic relighting. It is worth investigating the techniques of Lacewell et al. [24], where occlusion is prefiltered, and how it may be extended or incorporated into our current technique.

Acknowledgements

This work was supported in part by an NVIDIA Fellowship grant, an NVIDIA equipment donation through the Professor Partnership Program, and by the U.S. National Science Foundation under Award 0722313.

References

1. Behrendt, S., Colditz, C., Franzke, O., Kopf, J., Deussen, O.: Realistic real-time rendering of landscapes using billboard clouds. Computer Graphics Forum **24** (2005) 507–516
2. Geist, R., Steele, J.: A lighting model for fast rendering of forest ecosystems. In: IEEE/EG Symposium on Interactive Ray Tracing 2008, IEEE/EG (2008) 99–106, and back cover
3. Dietrich, A., Colditz, C., Deussen, O., Slusallek, P.: Realistic and Interactive Visualization of High-Density Plant Ecosystems . In: Eurographics Workshop on Natural Phenomena, Dublin, Ireland (2005) 73–81
4. Geist, R., Rasche, K., Westall, J., Schalkoff, R.: Lattice-boltzmann lighting. In: Rendering Techniques 2004 (Proc. Eurographics Symposium on Rendering), Norrkőping, Sweden (2004) 355 – 362,423
5. NVIDIA Corp.: Nvidia cuda programming guide, version 2.1. http://www.nvidia.com/object/cuda_get.html (2008)
6. Wang, L., Wang, W., Dorsey, J., Yang, X., Guo, B., Shum, H.Y.: Real-time rendering of plant leaves. ACM Trans. Graph. **24** (2005) 712–719
7. Reeves, W.T., Blau, R.: Approximate and probabilistic algorithms for shading and rendering structured particle systems. In: SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (1985) 313–322
8. Hegeman, K., Premože, S., Ashikhmin, M., Drettakis, G.: Approximate ambient occlusion for trees. In: I3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games, New York, NY, USA, ACM (2006) 87–92
9. Luft, T., Balzer, M., Deussen, O.: Expressive illumination of foliage based on implicit surfaces. In: Natural Phenomena 2007 (Proc. of the Eurographics Workshop on Natural Phenomena), Prague, Czech Republic (2007) 71 – 78
10. Lorensen, W., Cline, H.: Marching cubes: A high resolution 3d surface construction algorithm. In: Proc. SIGGRAPH '87. (1987) 163–169
11. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques. (2002) 527–536

12. Jensen, H.W.: Realistic Image Synthesis Using Photon Mapping. A.K. Peters, Natick, MA (2001)
13. Hašan, M., Pellacini, F., Bala, K.: Direct-to-indirect transfer for cinematic relighting. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, New York, NY, USA, ACM (2006) 1089–1097
14. Arvo, J.: Transfer equations in global illumination. In: Global Illumination, SIGGRAPH '93 Course Notes. (1993)
15. d'Humières, D., Lallemand, P., Frisch, U.: Lattice gas models for 3d hydrodynamics. Europhysics Letters **2** (1986) 291–297
16. Jensen, H., Marschner, S., Levoy, M., Hanrahan, P.: A practical model for subsurface light transport. In: Proceedings of SIGGRAPH 2001. (2001) 511–518
17. Stam, J.: Multiple scattering as a diffusion process. In: Proc. 6^{th} Eurographics Workshop on Rendering, Dublin, Ireland (1995) 51–58
18. Henyey, G., Greenstein, J.: Diffuse radiation in the galaxy. Astrophysical Journal **88** (1940) 70–73
19. Knapp, A., Carter, G.: Variability in leaf optical properties among 26 species from a broad range of habitats. American Journal of Botany **85** (1998) 940–946
20. Hunt, W., Mark, W.R.: Ray-specialized acceleration structures for ray tracing. In: IEEE/EG Symposium on Interactive Ray Tracing 2008, IEEE/EG (2008) 3–10
21. Lagae, A., Dutré, P.: Compact, fast and robust grids for ray tracing. Computer Graphics Forum (Proceedings of the 19th Eurographics Symposium on Rendering) **27** (2008) 1235–1244
22. Westermann, R.: A multiresolution framework for volume rendering. In: Symposium on Volume Visualization, ACM Press (1994) 51–58
23. Bajaj, C., Ihm, I., Park, S.: 3d rgb image compression for interactive applications. ACM Trans. Graph. **20** (2001) 10–38
24. Lacewell, D., Burley, B., Boulos, S., Shirley, P.: Raytracing prefiltered occlusion for aggregate geometry. (2008) 19–26