Light Meshes—Original Approach to Produce Soft Shadows in Ray Tracing

Victor A. Debelov, Igor M. Sevastyanov

Institute of Computational Mathematics and Mathematical Geophysics SB RAS, Russia, Novosibirsk
(debelov,sevastyanov)@oapmg.sscc.ru

Abstract. The given paper is devoted to the application of the original approach of Light Meshes to the recursive ray tracing algorithm. It is a sort of a simulation of a solution for the global illumination problem. It consumes not so much extra computational time and, nevertheless, allows: a) to speedup a process of shadow calculations, b) to build soft shadows, c) to simulate main effects of diffuse interreflections, d) to omit the ambient term. In order to obtain the goals listed above we introduce the notion of space light meshes. A light mesh is a point set (mesh points) in 3D space of a scene. At each mesh point we compute illuminance, which characterizes some kind of "light field intensity" in the scene. Then important part of resulting illumination of a scene object point (object point) is obtained via interpolation of stored values of nearby mesh points. Our experiments show that the suggested approach reduces computational time with respect to ray tracing algorithm if a scene contains a lot of light sources and/or an image has a large resolution.

1 Introduction

In the given work we consider the problem of photorealistic images synthesis of 3D scenes using a light-backwards recursive ray tracing algorithm (e.g. [1]) in the following context. Scenes are characterized as follows.

– Without participating media.
– Contain any number of point light sources.
– Consist of opaque one-sided surfaces. Each surface has diffuse emissive, diffuse and specular reflectance properties. We assume a quite simple model, i.e. each object point y is assigned: the diffuse reflection coefficient $k_d(y)$; the specular reflection coefficient $k_s(y)$; the diffuse emission $Emiss(y)$.
– A position of a camera and a final image resolution are given too.

Here we suggest an approach that should improve known drawbacks of ray tracing:

– Absence of soft shadows.
– Necessity to assign some value as the ambient term while calculations of an object point illumination.

– Poor accuracy of a diffuse interreflection calculation, even an absence of color bleeding (e.g. see [2] about the scene "Constructive Wood").

It is known that all these drawbacks are solved (at least theoretically) by heavy global illumination techniques. Since the ray tracing method is still the most popular and widespread used, we developed the modification of it in order to give a certain solution to posed problems.

1.1 Scalar Light Field

The irradiance in a scene is a function of five variables, three for the position and two for the direction. In the Stanford project "Light Field Rendering" light field is represented as 4D functions. In order to improve the accuracy of a diffuse interreflection calculation Ward and Heckbert [3] introduced the notion of the irradiance gradients; which is direction-dependent. They keep two vectors and a diffuse irradiance value for points on scene surfaces. In our approach we decided to reduce the amount of information up to a single value in each space point. Let's assume that light induces intensity in each point of a space of a scene similar to electrical or pressure (potential) fields. We suggest considering "a light scalar field" which is characterized by a single value of light field intensity $I(x)$, $x \in R^3$. This value does not depend on directions. Probably by comparing of values in two close points we can obtain directional information. Basing on ideas of the bidirectional Monte Carlo ray tracing algorithm, especially on illumination maps [4], photon maps [5] that are fulfilled in 2 steps: *forward)* rays (photons) are cast from light sources and stored in a certain structure in a scene surfaces; *backward)* path tracing from a camera to gather incoming intensities. As a rule a proper data structures are created taking into account a consideration of scene geometry. In our case we decide to accumulate light energy in every point of scene space. We do 2 steps too but while first step we issue backward rays from the space point to the point light sources in order to accumulate the direct illumination. In the paper [6] a scene space is divided into small volumes. Photons are traced from light sources and stored into volumes together with information on directions. Stored information is used while backward phase. In a paper [7] authors suggest to compute irradiance in space points with directional information also. In our algorithm we save computer memory because we do not keep information on directions at all, nevertheless we obtain quite realistic simulation of scene images.

1.2 Direct Light Field

Let's define the scalar field DirectLight(x) that accounts for direct illumination of arbitrary point $x \in R^3$:

$$\mathrm{DirectLight}(x) = \sum_{i=1}^{Nl} V_i(x) \cdot E_i \cdot \frac{1}{\varphi_1(r_i(x))} \tag{1}$$

where:

Nl is a number of point light sources;
$V_i(x)$ is the visibility. It takes 1, if ith point light source sees point x, otherwise 0.
E_i is the intensity (color) of ith point light source;
$r_i(x)$ is the distance from the point to ith point light source;
$\varphi_1(t)$ in the given work has the classical view $Const + t$.

Thus we accumulate direct illumination that reaches the point x from all directions. Then a direct illumination of an object point y – a point of a scene surface S – is defined via formula:

$$I(y) = \tilde{I}(y) \cdot \sum_{i=1}^{Nl} \varphi_2(n_y, lp_i - y) \cdot E_i \cdot \frac{1}{\varphi_1(r_i(y))} \tag{2}$$

where:

n_y is the normal vector in y,
lp_i is the position of ith light source,
$\varphi_2(n, v) = \begin{bmatrix} 0 & \text{if } \cos\angle(n,v) < 0, \\ \cos\angle(n,v) & \text{otherwise,} \end{bmatrix}$

$$\tilde{I}(y) = \frac{1}{\mu(D_y)} \int_{x \in D_y} \mathrm{DirectLight}(x)\, dV \tag{3}$$

$D_y \subset R^3$ is a certain neighborhood of the point y, an interpolation set. Really the shape of the neighborhood should be selected accordingly to the reflectance properties at the point y and geometric properties of a surface. But in the limits of the given work we do not consider it.
$\mu(D)$ is the measure (volume) of the set D. In other words we simply integrate and average out intensities of all points belonging to selected neighborhood.

1.3 Indirect Light Field

The diffuse illumination of a space point $x \in R^3$ is defined as:

$$\mathrm{IndirectLight}(x) = \int_{\Omega} L(x, \omega)\, d\omega \tag{4}$$

where: $L(x, \omega)$ is the incoming indirect radiance to the point x in the direction ω from an object point y. $L(x, \omega) = \tilde{I}(y)$ if y exists on a ray; otherwise 0. Ω is the unit sphere of directions. Instead of integration we calculate a sum using a fixed number of samples in a uniformly weighted, stratified Monte Carlo sampling analogously to [3]. As a rule a scene space point does not belong to a scene surface; so we integrate along a whole sphere of directions. If a ray hits a certain scene surface in the point y then the incoming diffuse intensity is calculated using formula:

$$\tilde{I}_d(y) = \frac{1}{\mu(D_y)} \int_{x \in D_y} \mathrm{IndirectLight}(x) \varphi_5(x, y)\, dV \tag{5}$$

$\varphi_5(x, y) = 1$ in the given work.

1.4 Full Intensity of Object Point

Let's suppose that we have computed both direct and indirect fields in each point of a scene space. We suggests the following simulation of a full local intensity of the object point y:

$$I_{\text{full}}(y) = I(y) + k_d(y)\tilde{I}_d(y) \tag{6}$$

where $k_d(y)$ is the coefficient of the diffuse reflection. It is a sum of the direct part and the indirect part. The former represents the irradiance accumulated from all "almost" visible point light sources and the latter represents the term obtained as a diffuse interreflection accumulated in surrounding points of a scene space.

Remark 1. If a scene surface emits the light in the point y then the expression (3) is slightly modified:

$$\tilde{I}(y) = \text{Emiss}(y) + \frac{1}{\mu(D_y)} \int_{x \in D_y} \text{DirectLight}(x)\, dV \tag{7}$$

1.5 Light Meshes

Obviously that continuous functions DirectLight(x) and IndirectLight(x) are represented as samples given in finite point sets. The corresponding point sets DLM and ILM we call light meshes although they may be arbitrary point sets without any regular structure. In a work [3] the meshless caching scheme was used when points are selected from consideration of geometric properties – the location of the computed indirect irradiance values is determined by the proximity and curvature of the surfaces, and does not fall on a regular grid. In the presented work we use almost a similar fashion but essentially we select the light mesh arbitrary, i.e. without geometric considerations. We wish the light mesh lives, i.e. senses scene geometry and adapts itself to it. It is our main goal (for future), and in the given paper we will try to show the feasibility of the approach that it allows reaching goals declared in the beginning of the paper.

Remark 2. Light meshes DLM and ILM may differ as a set of points.

Remark 3. In the tests given in the paper we used regular point sets.

Remark 4. It should be stressed that here we describe rather the simulation of illumination but the approximation of the global illumination solution. Although our approach was found during the study of global illumination its main formulations were derived basing on intuition and experiments too.

1.6 Recursive Gathering of Intensity

In Fig. 1 we show a standard path which used in a backward recursive ray tracing algorithm: the initial ray R1, the reflected ray R2 in the point P, the next reflected ray R3 in the point Q. Although light meshes DLM and ILM are quite dense in the space, only a few of their points are used for calculations – those points that belong to neighborhoods D_P and D_Q. We have the following algorithm that is similar to one described in [1].

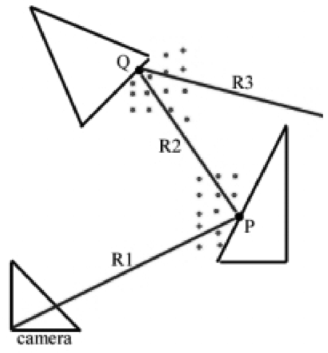

Fig. 1. The path traced from a camera

1. Initially all points of DLM and ILM are assigned the unknown value.
2. Analogously to [1] we build the path from a camera: camera $\rightarrow P \rightarrow Q \ldots$
3. Gather light energy from the end of the path.
4. At the point P we have intensity $I(Q)$ incoming along the ray R2.
5. Define the set DLM(P) – points of DLM belonging to D_P.
6. Calculate values for all points of DLM(P) with the unknown value using formula (1).
7. Calculate the direct illumination of P using formulas (3) and (2).
8. Define the set ILM(P) – points of ILM belonging to D_P.
9. Calculate values for all points of ILM(P) with the unknown value using formulas (4) and (5).
10. Calculate the full local intensity of P using formula (6).
11. Calculate the intensity outcoming from P to a camera using the expression:

$$I_{\text{out}}(P) = I(Q) + k_s(P)I(Q) \tag{8}$$

12. Repeat the procedure for all pixels of the image.

1.7 The Ambient Term

A user assigns the ambient term arbitrary and it is an artificial trick. The original method [8] allows defining it via the radiosity algorithm that estimates the diffuse interreflections between scene faces. Our light meshes carry out the similar work. Thus the formulas of the light meshes algorithm have no ambient terms.

2 Interpolation

During experiments with different interpolation schemes we selected two of them: Nearest and Spherical which gives acceptable images. The former scheme means that discrete neighborhoods DLM(P) and ILM(P) of the object point P include a single nearest mesh point. The left image in Fig. 2 illustrates this interpolation mode. Obviously that it gives images of bad quality but renders scenes very quickly because the only "layer" of mesh points nearby scene surfaces is computed. One can see the light meshes clear.

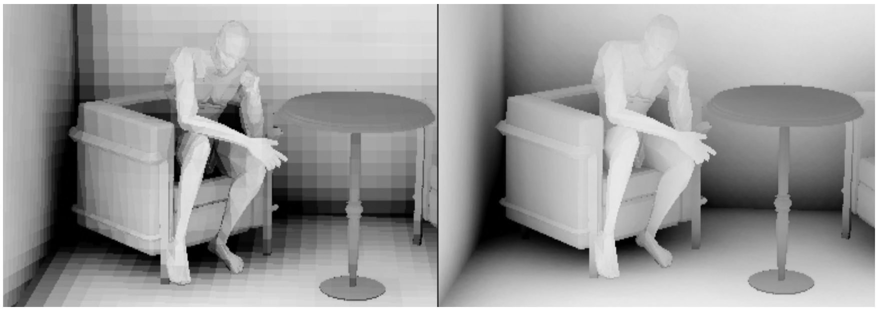

Fig. 2. Nearest interpolation mode (left). Spherical interpolation mode (right)

Spherical interpolation. The neighborhood D_P of the object point P is a full sphere of the definite radius. The right image in Fig. 2 is obtained using spherical interpolation over the same coarse meshes as the left image. The images in Fig. 3 illustrates that the method works quite well if a scene contains: specular surfaces (left image), diffuse surfaces (right image).

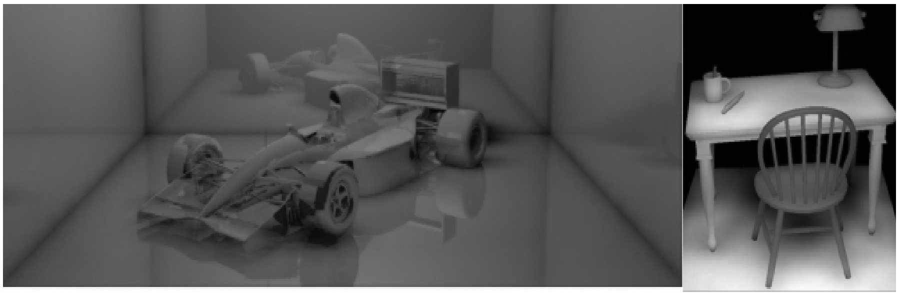

Fig. 3. Scene with specular surfaces (left). Diffuse surfaces (right)

3 Soft Shadows

It is known that soft shadows increase realism of synthesized images. Ray tracing is the method working with scenes illuminated by point light sources, and thus from the theoretical point of view it can produce only sharp shadows. Many specialists tried to enhance realism by developing techniques for simulations of soft shadows, it could be found in the Internet discussion. Main efforts concern two goals: a) to reduce a computational cost of shadows [9], b) to simulate soft shadows from point light sources. As one can see from formulas above we never use shadow rays from object points. It is implicitly done while calculating values of light meshes and does not depend on the depth of the recursion but on the number of actually used mesh points.

The cost of ray tracing depends on the number of light sources and consequently on the number of shadow rays. For example using of photon maps reduces this cost [10] but nevertheless shadow tests are done. In the paper [11] an inexpensive algorithm for soft shadows was introduced. It supposed the simulation of point light sources by spherical bulbs. Note that it works improper if an object casting shadows is too small. In order to create the effect of a soft shadow we calculate illuminations in DLM mesh points and then interpolate the values of nearby mesh points onto rendered surfaces of a scene. And it works! The soft shadows obtained looks quite realistic, namely: a shadow diffuses not only outside of the theoretical contour of a sharp shadow but inside it too, as well. Thus one could see the effect of an enlarged area of a point light source. It is important that the presented algorithm provides plausible results when too small scene objects (leaves) practically do not cast noticeable shadows, see example in Fig. 4. By the proper control of the shape and size of the interpolation neighborhood $D(y)$ and the density of light meshes a user can vary shadows from soft to sharp. A generation of soft shadows does not make the tracer slower, it allows computing faster. The difference becomes noticeable while increasing a number of light sources and/or scene objects and image resolution. Let's consider an example of a simple scene, consisting of 23542 triangles and 100 light sources. A conventional tracer calculates an image of 600×600 pixels for 277 seconds. A light mesh tracer takes 40 seconds. Experiments with scenes consisting of 100 light sources and 20000–50000 triangles show that a light mesh tracer is from 6 to 10 times faster than a conventional tracer.

4 Conclusion

We hope that we has shown the feasibility of the light meshes approach, and that the technique of light meshes extends the available arsenal of techniques enhancing the ray tracing algorithm. The indirect light mesh provides color interaction only between those surfaces that "see" each other. So the ray tracing problem of diffuse color interreflections is not solved yet by our algorithm, for example the mentioned scene "Constructive wood" is rendered wrong. Probably the reason of this relict drawback results from the independence of the direct and indirect meshes. We are going to continue investigations of this question. The second direction of our interests is an adaptive meshing and interactive control. The usage of light meshes allows step-by-step improvements of the image similar to the progressive radiosity algorithm [2]. After adding of new points to a mesh the values assigned to existing mesh points are not recalculated. As the adaptation

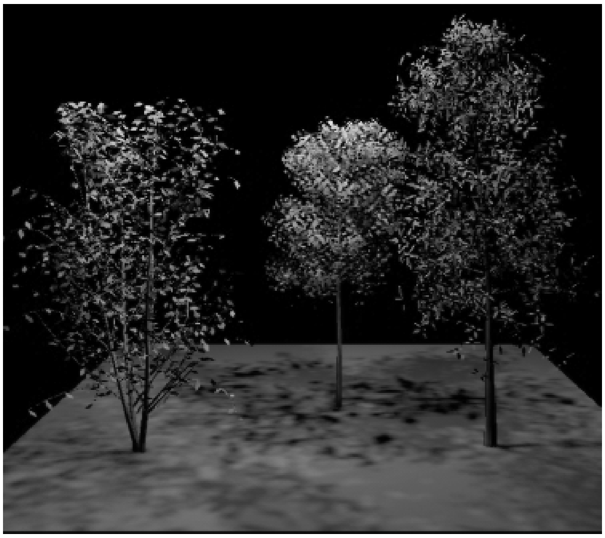

Fig. 4. Soft shadows of the forest leaves. Three point light sources are placed just above trees

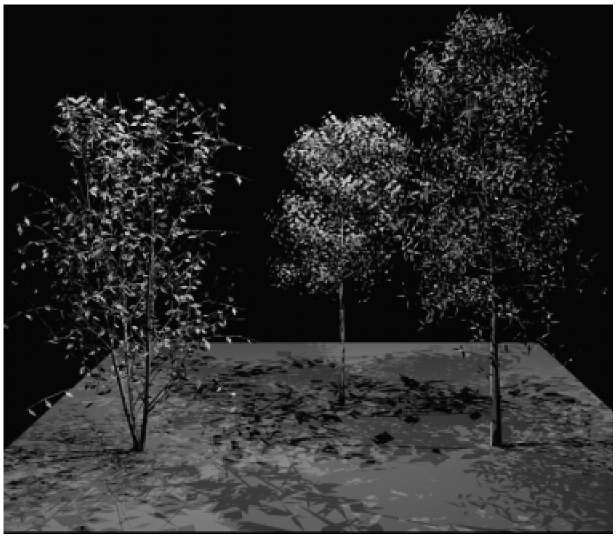

Fig. 5. Conventional ray tracing, sharp shadows

of a mesh we mean the process of increasing its density in certain regions of a scene. For example, if the interpolation sphere contains the neighboring points that differ in values more than some threshold then new points are added to the mesh. A user can control the process of image refinement interactively. Beginning from the coarse mesh and the "nearest" mode of interpolation he could: a) switch the interpolation mode; b) change the radius of an interpolation sphere; c) use more dense meshes in the next step; d) select regions of a scene where the meshes must be more dense; etc. And the final remark concerns possibilities to a specific parallelization of the presented algorithm. Calculations of all unknown values (steps 6 and 9 of the algorithm) can be done in parallel. Obviously we do not consider all questions concerning suggested space light meshes, we left beyond the paper such problems as fighting with artifacts on images, adaptive interpolation schemes, mathematics of light meshes.

5 Acknowledgements

The presented work was supported by Russian Foundation for Basic Research grants 99-0100577, 0101-06197.

References

1. Whitted, T.: An Improved Illumination Model for Shaded Display. Commun. ACM. Vol. 23, No. 6, (1980) 343–349.
2. Cohen, M.F., Wallace, J.R.: Radiosity and Realistic Image Synthesis. – Academic Press, New York (1993).
3. Ward, G.J., Heckbert P.S.: Irradiance Gradients. Proc. of the Third Eurographics Workshop on Rendering, Bristol, UK (1992) 85–98.
4. Arvo, J.: Backward Ray Tracing. Developments in Ray Tracing, SIGGRAPH'86 Course Notes Developments in Ray Tracing, 12 (1986).
5. Jensen, H. Wann, Christensen, N.J.: Photon Maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects. Computers and Graphics. 19, 2(1995) 215–224.
6. Chiu, K., Zimmerman, K., Shirley, P.: The Light Volume: an Aid to Rendering Complex Environments. Seventh Eurographics Workshop on Rendering, Porto, Portugal, 1996.
7. Greger, G., Shirley, P., Hubbard, P.M., Greenberg, D.P.: The irradiance volume. IEEE Computer Graphics and Applications, 18(2), 1998, 32–43.
8. Castro, L., Neumann, L., Sbert, M.: Extended Ambient Term. J. of Graphics Tools. 5, 4(2000), 1–7.
9. Ghali, S., Fiume, E., Seidel, H.P.: Shadow Computation: a Unified Perspective. Proc. EUROGRAPHICS'2000 CD. shadows.pdf.
10. Jensen, H. Wann, Christensen, N.J.: Efficiently Rendering Shadows using the Photon Map. Proc. of Compugraphics'95. Alvor (1995) 285–291.
11. Parker, S., Shirley, P., Smits, B.: Single Sample Soft Shadows. Tech. Rep. UUCS-98019, Computer Science Department, University of Utah (1998).