Photorealistic Lighting with Offset Radiance Transfer Mapping

Ben Sunshine-Hill* UCLA

Petros Faloutsos† UCLA

Abstract

We propose a precomputation-based approach for the real-time rendering of scenes that include a number of complex illumination phenomena, such as radiosity and subsurface scattering, and allows interactive modification of camera and lighting parameters. At the heart of our approach lies a novel parameterization of the rendering equation that is inherently supported by the modern GPU. During the pre-computation phase, we build a set of *offset transfer maps* based on the proposed parameterization, which approximate the complete radiance transfer function for the scene. The rendering phase is then reduced to a set of texture-blending and mapping operations that execute in real-time on the GPU. In contrast to the current state-of-the-art, which employs environment maps to produce global illumination, our approach uses arbitrary first-order lighting to compute a final lighting solution, and fully supports point and spot lights. To discretize the transfer maps, we develop an efficient method for generating and sampling C^0-continuous probability density functions from unordered data points.

We believe that the contributions of this paper offer a significantly different approach to precomputed radiance transfer from those previously proposed.

CR Categories: I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism; I.3.6 [Computing Methodologies]: Computer Graphics—Methodology and Techniques

Keywords: global illumination, gpu, radiosity, interactive

1 Introduction

Over the last few years, the increasing popularity of applications such as video games and virtual reality simulations has given rise to the need for interactive simulations of complex lighting effects. Where once low-poly wireframe renderings were the state of the art, modern real-time rendering is now expected to reproduce effects such as radiosity and subsurface scattering.

The rise of programmable Graphics Processing Units (GPUs) has brought significant graphical power to the personal computer. Programming complex mathematical models on a GPU can be challenging. Nevertheless, there is a flurry of research aimed at harnessing the power of the GPU for both general purpose and graphics related problems. Some attempts to implement advanced lighting effects on the GPU are very promising, such as the photon mapping simulation by Purcell et al. [2003]. However, the current state of the art is far from achieving real-time performance.

*e-mail: bsunshin@cs.ucla.edu

†e-mail: pfal@cs.ucla.edu

Copyright © 2006 by the Association for Computing Machinery, Inc.
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions Dept, ACM Inc., fax +1 (212) 869-0481 or e-mail permissions@acm.org.
I3D 2006, Redwood City, California, 14–17 March 2006.
© 2006 ACM 1-59593-295-X/06/0003 $5.00

Precomputed Radiance Transfer (PRT) techniques have been proposed which partially solve this problem. A precomputation phase determines a set of lighting parameters which may be applied at runtime. These techniques, however, have an important limitation: they require a scene's lighting to be expressed as an environment map over some external manifold, which can only provide a heuristic approximation of lighting from local sources such as point or spot lights. Thus, arbitrary local illumination is not generally supported.

In this paper, we propose a novel precomputation-based approach for the photorealistic rendering of complex scenes with diffuse transfer functions that efficiently leverages the strength of the modern GPU. Unlike current PRT techniques, our approach supports arbitrary local lighting with advanced illumination effects and interactively modified lighting and camera paramters, and performs in real-time. A key novelty of our approach is a new parameterization of the rendering equation which allows us to compute a texture-based representation of a scene's radiance transfer. This parameterization is based on grouping the light interactions of points into textures according to different distance offsets (vectors) in texture space. Each offset corresponds to a single texture. The resultant textures are a key element of our approach and are called *offset transfer maps*. This unique form of our parameterization reduces the rendering process to a sequence of texture operations which are inherently supported by the GPU, and which can be executed in constant-time with respect to the number of light sources. The rendering process may be parallelized across multiple GPUs with very low parallelization overhead.

Before we provide an overview of our approach, we need to define the concept of a *radiance transfer function.* Consider a point, x, in a scene. The incident light at this point depends on direct, or *first-order*, illumination from light sources, and indirect illumination which is transmitted to point x from all other points, x', in the scene. A *radiance transfer function*, $RTF(x,x')$, defines the fraction of the first-order illumination of x' that is transmitted to x, and depends only on a scene's geometry and reflectivity parameters.

We can now define offset transfer maps in more detail. Given a texture mapping $x \in \mathbb{R}^3 \leftrightarrow t \in \mathbb{R}^2$, an *offset transfer map with offset* Δt is a texture map whose value at any location t is the value of the radiance transfer function evaluated at $(t, t+\Delta t)$. It tells us the fraction of the incident light at the surface location corresponding to $t+\Delta t$ which is transmitted to the surface location corresponding to t. An offset transfer map therefore represents a "slice" of the radiance transfer function corresponding to points which in texture (parametric) space have distance Δt. By computing an appropriate set of offset transfer maps we can represent a radiance transfer function as a set of textures.

We can now give an overview of our approach, which consists of two phases.

Pre-computation phase. Given a scene to be rendered, we begin by discretizing the solution of the radiance transfer function. Rather than attempting to compute and store the entire solution, we discretize it into a finite set of offset transfer maps. We develop a novel probabilistic sampling approach that allows us to construct such a set of transfer maps while maintaining the quality of the results. As with other PRT techniques, the precomputed data is dependent on scene geometry and must be recomputed whenever that geometry changes. However, unlike conventional PRT techniques,

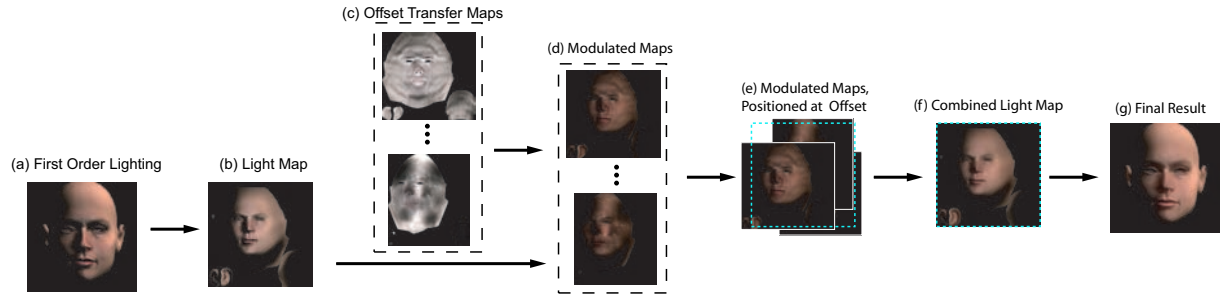


Figure 1: Rendering with precomputed offset transfer maps.

it is independent of both camera position and lighting, and can thus be used under a variety of circumstances, including local lighting, such as point and spot lights. Figures 3-6 show several effects produced by our approach.

Rendering phase. At runtime, first-order illumination (Figure 1a) is captured to a lightmap (1b). For each offset transfer map (1c), the lightmap is multiplicatively modulated by the transfer map (1d). The result of the modulation is offset by the offset associated with the transfer map (1e) and additively blended to the final lightmap (1f). Intuitively speaking, each modulated light map holds the percentage of first order lighting that each point receives, directly or indirectly, from a single point which is as far away in texture space as the offset of the associated offset transfer map. The final lightmap is then textured back onto the object (1g).

The contributions of this paper are:

Offset form of the rendering equation. At the heart of our approach lies a novel parameterization of the rendering equation in texture space which is inherently supported by a modern GPU. Our formulation allows a photorealistic lighting solution to be formed from first-order lighting, rather than an environment map.

Offset computation. To compute an appropriate set of offset for our transfer maps, we develop a method for efficiently generating and sampling C^0-continuous probability density functions from unordered data points.

Local lighting and complex effects. Our approach allows the combination of full support for local light sources with advanced illumination effects such as radiosity, subsurface scattering, and photon mapping.

Performance improvement methods. To further improve the performance of our approach, we implement a set of acceleration techniques. Among them, of independent interest is a method for efficiently packing axis-aligned texture regions into texture atlases which allows axial subdivision. To our knowledge, this is one of the first papers to solve this special case of texture atlas building.

2 Related Work

2.1 Precomputation for Interactive Lighting

Sloan et al. [2002] introduced a global illumination technique which uses spherical harmonics to sample an environment map. The technique allows self-shadowing and interreflections, and results are realistic, efficient, and not restricted to diffuse surfaces. The technique, however, is not intended for local illumination (wherein the light source is close to geometry) and has difficulty reproducing high-frequency lighting components. Ng et al. [2004] modified this technique to use wavelet lighting, extending the technique's suitability to high-frequency components as well as generalizing the light transport equation such that local lights are supported; this support, however, is based on a reparameterization of the transport equation that is specific to the light source. Sloan et al. [2003] extended PRT with a bi-scale lighting approximation to more effectively capture lighting effects from both large-scale and small-scale geometry.

Daubert et al. [2003] proposed precomputing and storing per-texel global visibility information, in order to efficiently perform a light-gathering step at runtime. This allows the computation of global illumination solutions in rigid scenes at near-interactive speed. Since visibility, rather than light transfer, data is precomputed, however, multiple scatterings must be performed in separate passes, and only fully opaque materials may be represented. Additionally, since spatial coherence is not exploited, a GPU-based implementation of this method would require several dependent texture reads per texel.

Recently, Kristensen et al. [2005] used clustered PCA to classify a large grid of potential local light sources for the purpose of performing PRT with local omnidirectional lights. Sloan et al. [2005] used zonal harmonics to perform low-frequency PRT for local features on arbitrarily deformable models.

2.2 Subsurface Scattering

The dipole approximation for subsurface light transport was proposed by Jensen et al. [2001]. A technique for simulating subsurface scattering by operating on a dynamically generated first-order light map was described by Green [2004], which performs a dynamically-sized Gaussian blur on the lightmap and uses it to modulate the scene. Our approach may in some respects be considered as a generalization of their technique, inasmuch as Gaussian convolution kernels are a subset of the effects possible with the approach we present. Their technique, however, is ineffective for objects with complex geometry such as ears, and people using this technique have had to augment it with standard ray-tracing for such situations [Borshukov and Lewis 2003].

2.3 Geometric and Statistical Methods

Lawson [1977] described the use of the Delaunay triangulation of irregularly spaced points to approximate a surface. Sibson [1981] suggested the Delaunay triangulation as a systematic triangulation that was near-optimal for reconstructing the surface function. Guibas and Stolfi [1985] described divide-and-conquer algorithms to compute the Delaunay triangulation which create a triangulation in $O(n \log n)$ and insert a new point into a triangulation in $O(n)$. As the Delaunay triangulation is the dual of the Voronoi polygonalization, these algorithms are also useful for computing the Voronoi polygonalization. Rocchini and Cignoni [2001] described a method for generating random points in a tetrahedron, which is useful for the sampling method given in Appendix A.

2.4 Texture Atlas Building

Much information has been published on fitting texture regions into a minimal rectangular space. This is an NP-complete problem [Fowler et al. 1981], an approximation to which has been described by Lévy et al. [2002]. Our technique, however, only requires the fitting of axis-aligned rectangular regions, and we may split the regions at will; we thus propose an alternative algorithm which is simpler and more efficient, described in Section 5.2.

3 Offset Form of the Rendering Equation

At the core of our approach is the use of texture maps for photo-realistic illumination. In this section, we derive the mathematical basis for applying texture blending to radiance transfer equations such as the radiosity equation. The goal of our derivation is an equation that will allow us to express a lighting solution as a function of first-order lighting in a form that may be approximated by 2D GPU texture maps. This will allow us to perform parallelized gather operations on first-order lighting data without dependent texture reads. In these equations, we assume that objects in the scene are comprised of the set $M \subset \mathbb{R}^3$ of all points on their surfaces.

We start with a simple transfer equation, that of second-order lighting $B_f(x)$ in a fully diffuse environment based on the first-order illumination $B_o(x)$ of the scene. That is, given light rays that have come from an emissive patch or other light source and hit exactly one object, we find the illumination caused by those rays as they hit objects for the second time. (Later, we will generalize the method to multiple scatterings.) If $F(x,x')$ is defined as the geometric form factor between two points, and $\rho(x)$ is the diffuse reflectivity of a point, second order illumination is given by

$$B_f(x) = \int_M \rho(x) B_o(x') F(x,x') dx'. \tag{1}$$

We convert to a slightly idiosyncratic discrete representation: instead of defining a set of n patches $p_i \subset \mathbb{R}^3$ as is usual, we define a division of M into n patches which we refer to by the set of their centroids $P \subset M$. This will allow us to relate the patches geometrically using a multidimensional texture mapping. We define $A(p)$ as the area of the patch having centroid p, and other functions similarly, and our equation becomes

$$B_f(x) = \sum_{x' \in P} \rho(x) B_o(x') F(x,x') A(x'). \tag{2}$$

We now define a series of n integer-valued points $T \subset \mathbb{I}^2$, and a mapping $TM : P \to T$ and its inverse $ITM : T \to P$, which we will refer to as a "texture mapping" and which represents a discretized GPU texture mapping. Mirroring the form of a GPU texture mapping, we assume that all elements are laid out in a regular square grid with edge length 1, and form centroids of patches with unit area in that grid. That is, for any integers u and v, if u and v are between 0 and $\sqrt{n}$, $\langle u,v \rangle \in T$. We define new functions $TA(t) = A(ITM(t))$, $TB_f(t) = B_f(ITM(t))$, $T\rho(t) = \rho(ITM(t))$, $TB_o(t) = B_o(ITM(t))$, and $TF(t,t') = F(ITM(t), ITM(t'))$ and rewrite the equation in terms of these:

$$TB_f(t) = \sum_{t' \in T} T\rho(t) TB_o(t') TF(t,t') TA(t'). \tag{3}$$

Because of the texture mapping invariants given above, for any two points t and t' in T, their spatial relation may be expressed in terms of an offset Δt as $t' = t + \Delta t$, where $\Delta t \in \mathbb{I}^2$. Defining $TB_o(t) = 0$ for $t \notin T$, we may therefore formulate the equation in terms of the offset Δt:

$$TB_f(t) = \sum_{\Delta t \in \mathbb{I}^2} T\rho(t) TB_o(t+\Delta t) TF(t,t+\Delta t) TA(t+\Delta t). \tag{4}$$

If we needed to express the radiance transfer in terms of every texel and every offset, the texture data would be far too large to be processed in anything approaching interactive speeds. Since texture mappings tend to map mostly-smooth areas of a mesh into a single contiguous region, however, we can assume that all terms in the equation, with the exception of $TA(t+\Delta t)$ (see below), have a certain degree of coherence with respect to Δt. We therefore do not need to sample every Δt in order to reasonably approximate the solution; instead, we sample from a set $O \subset \mathbb{I}^2$, and treat them as representative of the areas around them. The value in Equation 4 of the addend for each offset from our limited set of offsets is assumed to be close to the average of the results of all integer-valued offsets that are closest to it. Therefore, given the Voronoi graph generated by the set of all offsets, the Voronoi region of an offset within the Voronoi graph generated by the set of all offsets represents the area for which a particular offset is most representative. (In Section 3.1 we present a method of choosing a set of offsets such that this is the case.) For each offset, we multiply its contribution to the summation by the area of the Voronoi region it describes. Defining $|V(\Delta t, O)|$ as being the size of the Voronoi region containing Δt in the Voronoi graph described by all elements of O, our approximation is

$$TB_f(t) = \sum_{\Delta t \in O} T\rho(t) TB_o(t+\Delta t) TF(t,t+\Delta t) TA(t+\Delta t) |V(\Delta t, O)|. \tag{5}$$

Note that for all points on the convex hull of O, $|V(\Delta t, O)|$ is infinite. In order to prevent numeric instability, we require that all offsets Δt forming the hull are large enough that $\forall t, TA(t) = 0 \vee TA(t+\Delta t) = 0$, and that the result from that offset is therefore always 0.

This formulation is ideal for parallelization by the GPU. We view a "texture" as a discretization of a 2D function. In the case of the above equation, we have a texture for TB_o and several textures, each representing a discretization of

$$T\rho(t) TF(t,t+\Delta t) TA(t+\Delta t) |V(\Delta t, O)| \tag{6}$$

over t for a particular $\Delta t \in O$. This "transfer map", associated with the offset which generated it, is invariant of the light received by the patches, and depends solely on the reflectivity and geometry of the surfaces and on the choice of patches and mappings. It can, therefore, be computed and stored before it needs to be displayed. We also have a texture that represents a discretization of $TB_o(t)$, produced each frame by the standard diffuse light model. A simple fragment program allows us to combine these and produce a discretization of $TB_f(t)$, which can then be sampled to render the original mesh M. By varying the discretization of $TB_o(t)$ at runtime, we can easily produce an approximation of second-order reflections for any conditions of first-order lighting for the mesh M. Because of the unique offset form, the modulation of all points on the lightmap is parallelized for each offset transfer map. This allows large performance gains over other forms of parallelized gather operations.

We are, however, not limited to second-order reflections. Consider a scene that is lit such that only one patch receives a unit amount of first-order radiance, and all other patches receive no first-order radiance; a physical analogue would be a darkened mesh with a laser illuminating one point. The illumination transfers of all orders which result from this illumination may be computed through conventional global illumination techniques. One such solution for each point receiving first-order radiance may be computed, and the result is a discretization of the 6-dimensional function $R(x,x')$

which expresses the amount by which the point x is illuminated, by second- through infinite-order reflections, by a unit amount of first-order radiance received by the point x'. Defining $TR(t,t')$ similarly to those functions defined earlier, we may therefore replace the reflectivity and form-factor terms with this generalized transfer model, and express a full illumination solution as

$$TB_f(t) = \sum_{\Delta t \in O} TB_o(t+\Delta t)TR(t,t+\Delta t)TA(t+\Delta t)|V(\Delta t,O)|. \quad (7)$$

The determination of $R(x,x')$ to implement different effects is discussed in Section 4.

Our early results with this technique showed severe artifacts with most meshes, consisting of certain polygons in the mesh which were unnaturally bright or dark. We traced these artifacts to a naive method of computing $TA(t+\Delta t)|V(\Delta t,O)|$ in Equation 7. Our method was to simply sample the area at the given point, and multiply by the area of the Voronoi region. This led to instability in meshes where $TA(t)$ varied widely, as a polygon with an unusually small or large value for $TA(t)$ could have its influence magnified if it were one of only a few points being gathered and thus had a relatively large Voronoi region. Our solution is to sample at all points within the Voronoi region, not just one, and replace $TA(t+\Delta t)|V(\Delta t,O)|$ with a function $TVA(t+\Delta t)$ such that

$$TVA(t+\Delta t) = \sum_{v \in V(\Delta t,O)} A(t+v) \quad (8)$$

Our final formulation of the offset form of the rendering equation is therefore

$$TB_f(t) = \sum_{\Delta t \in O} TB_o(t+\Delta t)TR(t,t+\Delta t)TVA(t,\Delta t) \quad (9)$$

and the general form of a transfer map, a specialization of which was given in Equation 6, is

$$TR(t,t+\Delta t)TVA(t,\Delta t) \quad (10)$$

for some Δt. This formulation eliminates the aliasing artifacts mentioned earlier, and also gives the resultant renderings a smoother, more continuous look. Reformulating other terms of the equation such as $T\rho(t)$ after this fashion did not appreciably improve the result, as these terms are more coherent across the texture.

The offset form of the rendering equation, Equation 9, is the central contribution of this paper. It provides a parameterization of the rendering equation that can be efficiently parallelized by the SIMD architecture of modern GPUs. Furthermore, it supports local illumination and is runs in constant time with respect to the number of light sources, since solutions are determined by the first-order lighting of the scene. By computing an appropriate $TR(t,t+\Delta t)$ our formulation can support a range of complex illumination effects. Section 4 shows specifically how we can implement some of these effects. The effectiveness of our method depends on choosing an appropriate set of offsets, which is discussed in the next section.

3.1 Determining the Offsets

In order to ensure that results are both efficient and realistic, it is important to pick a set of offsets carefully; early implementations used a jittered grid, resulting in aliasing artifacts even with hundreds of transfer maps. Finding a "good" set of offsets is well-trodden ground in the realm of Monte Carlo methods, as importance sampling [Hammersley and Handscomb 1965]. The variance of a Monte Carlo method is generally minimized when the probability distribution roughly corresponds to the integrand being evaluated, and that is the approach we take. There is a wrinkle, however: whereas conventional Monte Carlo techniques generally evaluate functions whose results are single scalars, the set of offsets must suffice to produce good results for all patches on the mesh and therefore should be evaluated for all patches. We have used the maximum transference as the probability weighting with good results, but this may be unsuitable for certain texture mappings. In particular, using the average transference as the metric reduces aliasing in meshes with only a single contiguous texture region and relatively simple topology. For more complex scenes consisting of multiple objects, using the maximum transferrence as the heuristic prevents distant objects from being ignored by the importance sampling.

In order to determine our final set of offsets, we build up an importance distribution and sample randomly from it by using the importance as a probability metric. We seed the distribution by sampling transferrences over a regular grid, then iteratively refine by sampling according to the current distribution and adding the computed transference at the resultant sample as a data point within the distribution. For the generation of the continuous importance distribution from the unordered data points, see Appendix B.

4 Modeling Illumination Phenomena

An important aspect of our formulation, shown in Equation 9, is that it supports a number of complex illumination effects. We can reproduce a desired effect by using an appropriate $R(x,x')$ and $TR(t,t')$ for Equation 9. In this section, we present two popular effects and methods for computing $TR(t,t')$ for each.

4.1 Radiosity

Conventional radiosity solving techniques can be easily adapted to produce the offset form of radiosity. The solution of $TR(t,t')$ for a particular defined t' is simply the radiosity solution where t' has emissivity 1 and all other patches have emissivity 0. By setting the emissivity of the mesh progressively for each point, $TR(t,t')$ can be built up. The performance of such a solver may be greatly improved by reusing certain data structures built during the solving process. Since the resultant data is constant as long as the mesh geometry and reflectivity are constant, an arbitrarily long time may be spent refining the solution without affecting the rendering speed.

Photon mapping provides an alternative method of generating $TR(t,t')$. All patches may be made emissive, and individual photons "tagged" with the patch from which they originated. Once the photons have been propagated, $TR(t,t')$ may be estimated by taking into account only those photons which originated from the source Voronoi region.

4.2 Subsurface Scattering

Anisotropic subsurface scattering may be easily added to the renderings. Since conventional subsurface scattering techniques approximate single- and multiple-scattering with single-scattering terms [Jensen et al. 2001], they are quite simple to add to the modified radiosity equation. In our implementation, we have used Jensen's dipole approximation as well as a simple exponential falloff, and have found both to give good results. This is even the case with complex geometries such as ears and noses which have commonly proven problematic for image-based subsurface scattering techniques [Borshukov and Lewis 2003]. Because intensity for subsurface contributions may be sampled at an arbitrary distance on the texture from the destination point, points that are close together on the mesh but distant on the texture are properly lit.

5 Implementation

We have implemented this technique on a consumer-level PC using OpenGL as the rendering API. Using modern graphics optimizations, we have achieved realtime speeds for photorealistic renderings. Our implementation suite consists of three programs, each representing a computation phase which we will describe in turn.

5.1 Computing Offsets and Building Maps

The first program reads in scene geometry as an X file. It is responsible for computing a set of offsets and generating the transfer maps for those offsets. First, the probability distribution is seeded with a regular sparse grid of about 100 offsets by computing a radiance transfer map at each offset and adding a data point to the distribution with magnitude equal to the maximum magnitude of the radiance transfer for the map. The Delaunay triangulation is computed using Guibas and Stolfi's divide-and-conquer algorithm. Next, the distribution is iteratively refined by generating random offsets from the distribution, computing the maximum radiance transfer at those offsets, and adding the value as a new data point to the distribution. The performance of this process may be improved by generating a batch of offsets with the same iteration of the distribution before computing the corresponding maximum radiance transfers, since generating n values from the same distribution has lower time complexity than generating n values, one each from a different distribution. Once a certain number of refinements has occurred (we have found 1000 iterations to be more than sufficient), the distribution is considered to have converged. The final set of offsets is then generated, and the Voronoi area for each is computed, again using Guibas and Stolfi's divide-and-conquer algorithm. The Voronoi regions are clipped to avoid numeric instability; we have found that clipping to the square from $(-1,-1)$ to $(1,1)$ works well. Transfer maps are computed using the method given in Section 3. The program crops the radiance transfer maps to minimal bounding boxes, including axial slicing. The images are then written to disk as separate OpenEXR files [Kainz 2004], and an index file which describes the images and their offsets is written.

5.2 Binning Maps Into Texture Atlases

GPUs must flush their processing pipelines when texture units are changed, introducing a delay on completing the rendering of a frame. If each offset transfer map were to be loaded as a separate texture, this would become a considerable runtime penalty. By binning multiple offsets into single textures, texture unit changes are minimized, and rendering speed may be increased by a factor of 5 or more. The second program performs this binning. Each map is split into vertical slices, each slice having a width which is a power of two. Bins are then preferentially filled in a manner that avoids breaking textures vertically: if some unbinned textures have the same width as an empty bin area and a lesser height, the best-fitting one is chosen and the remaining empty area is returned to the pool of areas. If all unbinned textures of the proper width are taller than the space available, the tallest one is chosen and split. If no unbinned textures of the appropriate width are available, the area is subdivided. Subdividing into binary widths converts the difficult problem of two-dimensional bin packing to the relatively simple problem of one-dimensional bin packing, and the best-fit/largest-split approach to packing minimizes the number of vertical splits that must be made. This method is trivially optimal with respect to minimizing number of bins needed, and empirically we have found it to perform well with respect to minimizing the number of splits (see Section 6.1). Pseudocode for this algorithm is available at `http://www.cs.ucla.edu/~bsunshin/binalgorithm.pdf`.

5.3 Rendering

The third program is responsible for the actual realtime rendering of images. It loads in the original mesh, as well as the generated texture atlases. The set of transfers for each atlas is encoded into an OpenGL display list. It then initializes an OpenGL window and renders in a loop. Arbitrary fragment shaders may be used to provide first-order lighting, or the first-order lightmap may be simulated by painting from a small toolbox; we have tested the method successfully with spot, point, directional, ambient, and environmental lighting. The mesh may be rotated, and the first-order and final meshes are displayed side-by-side.

6 Results

Our method supports arbitrary lights and advanced illumination phenomena for complex meshes in real-time. In Section 4 we describe specifically how we can implement radiosity and subsurface scattering. In this section we demonstrate the effectiveness of our implementation of these methods.

Our tests were performed with a neutral face model with slightly exaggerated subsurface scattering (in order to more clearly demonstrate the effect). This mesh contains discontinuities and severe distortions in its texture mapping, as well as widely varying curvature (including extremely concave regions) and form factors, and thus is sufficient to demonstrate the results of a technique within a complex geometric scene. For additional images and animations we refer the reader to the accompanying video.

Radiosity. Figure 6a shows the effects of global illumination for a face mesh illuminated by a directional light positioned above the head. Although the lower part of the nose is not directly illuminated, it receives light that is reflected from other parts of the face.

Subsurface scattering. Translucent objects such as marble and skin allow incident light to penetrate their surface up to some depth. The light scatters within such objects and often exits at locations other than the original entry point. Because of these phenomena, translucent objects appear smoother and when backlit their perimeter exhibits a warm glow. Figure 6b shows a backlit face mesh demonstrating how our implementation captures this complex phenomenon. Figure 3 shows a face mesh illuminated by two local lights, a blue light at the front and a red light at the back. The effect of subsurface scattering is most visible at the nose contour where some amount of red light scatters through the skin and reaches the blue side.

The combination of radiosity and subsurface scattering produces highly photorealistic results. Figure 5 shows the face mesh illuminated by a local point-light source. Figure 4 shows a comparison between direct illumination (first order) and global illumination from a directional light source. The combination of radiosity and subsurface scattering results in a smoother and more saturated face mesh.

6.1 Performance details.

We have tested our method on an 2.53 GHz Intel Pentium 4 computer with an NVIDIA GeForce 6800. The face mesh consists of 3686 vertices and 6918 faces. Groups of between 50 and 400 transfer maps were generated, in sizes ranging from 256x256 to 512x512 pixels. Transfer maps were split and rejoined into 1024x1024 texture atlases. Our binning algorithm split each map into an average of 7-9 pieces. With as few as 100 transfer maps, offset aliasing artifacts (in which artificial edges appear in the middle of the lightmap) were invisible and the result was fully photorealistic. We believe the dominant factor in determining framerate to be the total size of the texture data: each texel in the texture data corresponds

Bin size (pixels)	# Offsets	# Bins	Map-gen Time (sec)	FPS
256x256	50	3	22	30.08
256x256	100	6	28	15.04
256x256	200	10	42	8.59
256x256	400	19	57	1.96
384x384	100	13	36	7.52
384x384	200	24	107	1.43
512x512	100	22	197	1.72
512x512	200	43	226	1.01

Table 1: Implementation details and timing results.

to a blended framebuffer write, and framebuffer writing is the bottleneck of the pipeline in this system under normal circumstances. Framerates varied from 1.01 frames per second to 30.08 frames per second, and will likely be even higher with the next generation of GPUs and with multi-GPU systems. Our timing results are given in Table 1.

The GPU we have used does not automatically support filtering of full-precision floating-point textures. This leads to magnification artifacts such as those visible in Figure 6a. If desired, it is possible to eliminate such artifacts by performing filtering in the final pixel shader. Since the system does not perform significant computation in that pass, overhead would be minimal.

7 Discussion

Our approach to radiance transfer is significantly different from those currently used in GPU applications, and it is one with the potential for many augmentations. We present several possible modifications to the technique that may increase its generality and usefulness.

Currently, the vertices passed to the transfer program are such that the first-order lightmap is not transformed in any way before being modulated and added to the full lightmap. If the technique were to perform arbitrary affine scalings, rotations, and shearings on the lightmap for each transfer map, certain effects such as radiosity would be less dependent on a well-chosen texture mapping. Since GPUs always interpolate texture coordinates, such transformations would be essentially "free" with respect to rendering speed. Such an improvement would require a careful reformulation of the underlying equations in order to ensure that points do not receive uneven amounts of lighting, as well as efficient algorithms for determining an optimal set of transformations.

Likewise, the use of a set of global offsets, rather than local offsets within discrete regions of the texture mapping, is a simplifying assumption which in future work may be relaxed. Using local rather than global offsets may improve the quality of renderings on texture mappings with extremely divergent scaling factors. Like the use of arbitrary transformations, this change would require a reformulation of the offset form of the rendering equation in order to normalize lighting in boundary areas.

Although the axial splitting used in our implementation is sufficient for removing many areas of the transfer maps that do not significantly contribute to the final product, more advanced methods are possible. Texture atlases tend to contain separate texture areas that are not axially separated, in order to maximize the used area of the texture. If a method of identifying and isolating these areas were developed, this wasted space could be reclaimed.

Limitations. Our approach is based on a diffuse form of the rendering equation, and as such assumes that all surfaces are diffuse. In certain circumstances, however, this restriction may be relaxed: as long as the first and last patches that a light beam reflects from are diffuse, the 'middle' reflections may be off of surfaces with arbitrary reflectivity functions. Therefore, if all surfaces behave as fully diffuse when participating in first and last reflections, the requirement that they be diffuse otherwise may be lifted. For instance, we have had some success adding a specular lighting term to the computed final lighting map in the final rendering. Since such specular terms are not factored into the radiance transfers, lighting situations with large specular contributions may appear darker than normal. The technique also depends on a texture mapping encompassing all objects which are to participate in interreflections.

Rendering with a very low number of transfer maps (50 or less) may result in sampling artifacts; these artifacts rapidly become invisible at higher numbers of transfer maps. Small discontinuities in lighting may appear across texture seams with lower numbers of transfer maps.

7.1 Multi-GPU Parallelization

This technique could be easily parallelized across an arbitrary number of GPUs with very low overhead. Since the accumulation of radiance components is additive, each GPU may be assigned a different subset of the texture bins, and the result from each GPU additively blended to produce the final lightmap. The only overhead of this approach compared to a single GPU would be the time spent rendering first-order lighting on each GPU and the time spent transferring the lightmaps to the display GPU and accumulating them. For a very large number of GPUs where the accumulation overhead is significant, standard parallel summation techniques could be used to reduce the time to $O(\log n)$ in the number of GPUs.

8 Conclusion

We have proposed a precomputation-based approach for the photorealistic rendering of complex scenes that efficiently leverages the strength of the GPU. At the heart of our technique lies a novel parameterization of the rendering equation which discretizes the radiance transfer function into texture space. Our approach supports complex illumination effects, as well as arbitrary local lighting which may be interactively modified in real-time. In addition, its performance is independent of the number of light sources, since lighting solutions are determined by the first-order lighting of the scene.

We believe that the contributions of this paper represent an expansion of the field of precomputed radiance transfer in a new direction.

9 Acknowledgements

The work in this paper was partially supported by NSF under contract CCF-0429983. The authors would like to thank the anonymous reviewers for helping to manifestly improve the clarity of this paper. We would also like to thank Intel Corp., Microsoft Corp., ATI Corp. and Alias Corp. for their generous support through equipment and software grants.

References

BORSHUKOV, G., AND LEWIS, J. P. 2003. Realistic human face rendering for "The Matrix Reloaded". In *Proceedings of SIGGRAPH 2003*, ACM Press, 1–1.

DAUBERT, K., KAUTZ, J., SEIDEL, H.-P., HEIDRICH, W., AND DISCHLER, J.-M. 2003. Efficient light transport using precomputed visibility. *IEEE Comput. Graph. Appl. 23*, 3, 28–37.

FOWLER, R. J., PATERSON, M., AND TANIMOTO, S. L. 1981. Optimal packing and covering in the plane are NP-complete. *Information Processing Letters 12*, 3, 133–137.

GREEN, S. 2004. Real-time approximations to subsurface scattering. In *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, R. Fernando, Ed. Pearson Higher Education, ch. 16, 272–275.

GUIBAS, L., AND STOLFI, J. 1985. Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams. *ACM Trans. Graph. 4*, 2, 74–123.

HAMMERSLEY, J. M., AND HANDSCOMB, D. C. 1965. *Monte Carlo Methods*. Chapman and Hall.

JENSEN, H. W., MARSCHNER, S. R., LEVOY, M., AND HANRAHAN, P. 2001. A practical model for subsurface light transport. In *Proceedings of SIGGRAPH 2001*, ACM Press, 511–518.

KAINZ, F. 2004. The OpenEXR image file format. In *GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics*, R. Fernando, Ed. Pearson Higher Education, ch. 26, 425–444.

KRISTENSEN, A. W., AKENINE-MÖLLER, T., AND JENSEN, H. W. 2005. Precomputed local radiance transfer for real-time lighting design. *ACM Trans. Graph. 24*, 3, 1208–1215.

LAWSON, C. L. 1977. Software for C^1 surface interpolation. In *Mathematical Software III*, J. R. Rice, Ed. Academic Press, 161–194.

LÉVY, B., PETITJEAN, S., RAY, N., AND MAILLOT, J. 2002. Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics 21*, 3, 362–371.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2004. Triple product wavelet integrals for all-frequency relighting. *ACM Transactions on Graphics 23*, 3, 477–487.

PURCELL, T. J., DONNER, C., CAMMARANO, M., JENSEN, H. W., AND HANRAHAN, P. 2003. Photon mapping on programmable graphics hardware. In *Proceedings of HWWS 2003*, Eurographics Association, 41–50.

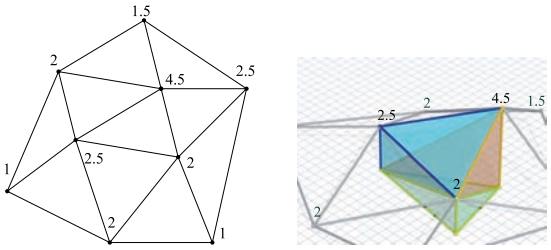
ROCCHINI, C., AND CIGNONI, P. 2001. Generating random points in a tetrahedron. Avaiable at `http://vcg.isti.cnr.it/publications/papers/rndtetra_a.pdf`.

SIBSON, R. 1981. A brief description of natural neighbor interpolation. In *Interpreting Multivariate Data*, V. Barnett, Ed., Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, ch. 2, 21–36.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *Proceedings of SIGGRAPH 2002*, ACM Press, 527–536.

SLOAN, P.-P., LIU, X., SHUM, H.-Y., AND SNYDER, J. 2003. Bi-scale radiance transfer. *ACM Trans. Graph. 22*, 3, 370–375.

SLOAN, P.-P., LUNA, B., AND SNYDER, J. 2005. Local, deformable precomputed radiance transfer. *ACM Trans. Graph. 24*, 3, 1216–1224.



(a) Delaunay triangulation of data points.

(b) Stacked tetrahedra representing one triangle in the distribution.

Figure 2: Building a continuous probability distribution from unordered data points.

A Importance Sampling from Unordered Data Points

In order to determine the set of offsets as described in Section 3.1, we have developed an efficient method of generating a C^0 continuous probability distribution from a set of unordered data points, and of sampling from this distribution.

For a particular collection of probability data points, the probability for an arbitrary sample point (which may or may not be in the set of data points) is interpolated between data points by finding the Delaunay triangulation of all points (Figure 2a), and performing a bilinear interpolation on the points of the triangle a particular target point is in. The generated function is thus C^0 continuous. The Delaunay triangulation is used because it has the desirable property that most points are in a Delaunay triangle determined by the three closest vertices.

For the purposes of sampling, each triangle within the Delaunay triangulation is partitioned into a stacking of three tetrahedra, each with three vertices determined by the three vertices of the triangle over the XZ plane, and the fourth determined by one each of the vertices with a Y height of that vertex's weighting (Figure 2b). The combined area of the three tetrahedra is equal to the linear average of the three weightings, multiplied by the area of the base triangle. To find a sample, it is sufficient to randomly select a tetrahedron based on a weighting factor of the volume of the tetrahedron, then sample a point uniformly from within that tetrahedron. A single sample from a triangulation with n triangles may be obtained in $O(n)$ with no precomputation, and m samples from the same distribution may be obtained in $O(m \log n)$ by performing binary searches for the random samplings within the progressively summed areas (as is common when sampling from discrete probability distributions). For our purposes, we have found no discernable loss of quality when distributions are only recalculated after every 20th added sample.

This interpolation method is efficient and numerically robust, and permits the distribution to be adaptively refined without a drop in performance (once a triangulation is established, a point may be added to it in $O(\log n)$). Although the Delaunay triangulation is commonly used to correlate unordered data points, to our knowledge this is the first time that these techniques have been used to define a continuous probability density function from such data.

Figure 3: Subsurface scattering in a mesh lit by two point lights.

(a) First order illumination.

(b) Global illumination with subsurface scattering.

Figure 4: Comparing first-order and global illumination.

Figure 5: Radiosity and subsurface scattering with one point light source.

(a) Radiosity.

(b) Subsurface scattering.

Figure 6: Complex illumination effects.

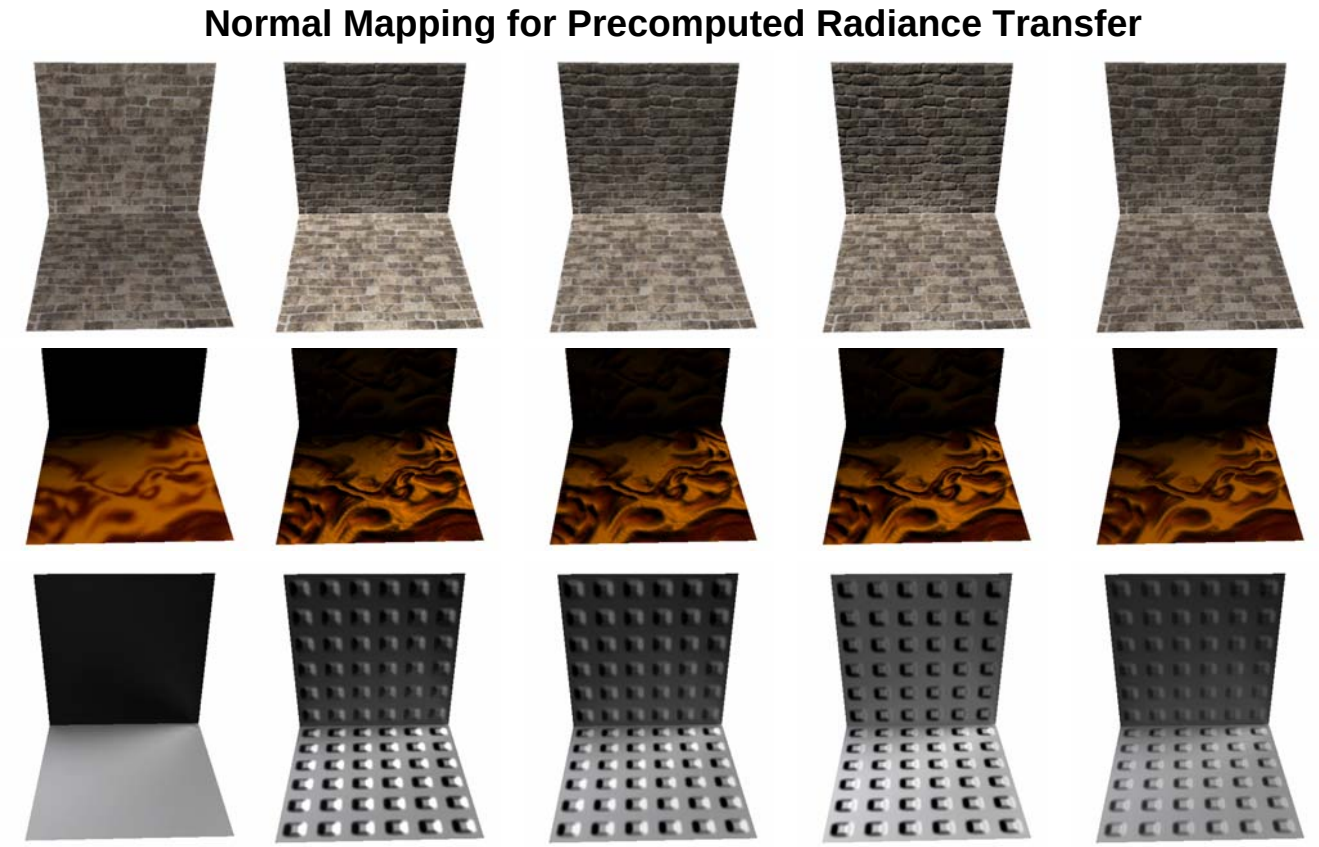


Figure 3: Simple scene, PRT, Gold Standard, Separable, Half-Life 2, Shifted Associated Legendre Polynomials