Precomputed Visibility Cuts for Interactive Relighting with Dynamic BRDFs

Oskar Akerlund
UMass Amherst
akerlund@cs.umass.edu

Mattias Unger
UMass Amherst
mattias@cs.umass.edu

Rui Wang
UMass Amherst
ruiwang@cs.umass.edu

Abstract

This paper presents a novel PRT-based method that uses precomputed visibility cuts for interactive relighting with all-frequency environment maps and arbitrary dynamic BRDFs. Our method is inspired by the recent Lightcuts approach [25] and we parameterize distant environment lighting onto uniformly distributed sample points over the sphere. Using a binary tree structure of the points, we precompute and approximate each vertex’s visibility function into clusters that we call the precomputed visibility cuts. These cuts are iteratively selected with bounded approximation error and confined cluster size. At run-time, a GPU-based relighting algorithm quickly computes the view-dependent shading color by accessing a dynamically built light tree, the precomputed visibility cuts, and a direct sampling of an arbitrary BRDF using each visibility cluster’s average direction and the dynamic view direction. Compared to existing PRT techniques, our method guarantees uniform sampling of the lighting, requires no precomputed BRDF data, and can be easily extended to handle one-bounce glossy indirect transfer effects in real-time.

1. Introduction

Interactive relighting with large-scale detailed environment maps, complex shadows and arbitrary BRDFs presents a great challenge in real-time rendering. Conventional systems that use ray-based techniques (ray tracing, photon mapping etc.) can take minutes to hours to simulate one frame, preventing real-time manipulation of lighting, viewpoint or material properties on the fly.

Recent advances in Precomputed Radiance Transfer (PRT) [20, 14] have enabled real-time photorealistic relighting effects by assuming static scenes. However, several key challenges remain. First, existing PRT techniques are often limited to low-frequency surface reflection effects, due to the use of either low-frequency illumination basis [20, 21] or low-frequency BRDF basis [12, 27]. Successive research efforts address these limitations by using fast wavelet triple

product integrals [15, 13] or novel basis functions [6, 23]; nonetheless, online changing of dynamic BRDFs remains a challenging problem, as these methods either bake a known BRDF into precomputation or require expensive preprocessing of BRDFs using basis projection techniques. In addition, many current systems parameterize the lighting onto distant environment maps, making them difficult to apply in scenarios that require complex local lighting effects, such as in computer games or interior design.

In this paper, we introduce a novel PRT-based method that uses precomputed visibility cuts for interactive relighting with all-frequency environment maps and arbitrary BRDFs. Our method is inspired by the recently proposed Lightcuts [25] approach, which approximates arbitrary illumination with sufficiently many point lights. Their key idea is to group coherent lights into piecewise constant clusters, thus reducing the rendering cost while bounding the maximum approximation error. This method allows different illumination sources (environment lights, indirect lights etc.) to be incorporated in a single unified framework.

We take use of this idea in PRT. Specifically, we parameterize distant environment lighting onto a set of uniformly distributed sample points over the unit sphere. We build a binary tree structure of the illumination samples, then precompute and approximate each vertex’s visibility function into clusters that we call the *precomputed visibility cut*. These clusters represent a piecewise constant approximation of the full visibility function and are chosen with bounded maximum error. In addition, we constrain the size of each cluster to be small enough such that the BRDF variation within each cluster can be ignored. This allows us to factor the BRDF term out of the illumination integral and replace it instead with a single sample dynamically evaluated using each cluster’s representative direction. At run-time, the relighting results can then be quickly computed using a dynamically built light tree, the precomputed visibility cuts, and a direct sampling of the BRDF within each cluster.

Our system provides the following key features:

- **Uniform illumination sampling.** Our method works with an arbitrary, unparameterized point set that represents uniformly distributed illumination samples. By

Figure 1. Realistic and dynamic relighting effects captured with our system.

doing so, we completely avoid uneven sampling of environment maps that is inevitable with traditional parameterized schemes such as cubemaps [14], octahedron maps [26] or spherical wavelets [13].

- **Fully dynamic BRDFs.** Our system does not require any precomputed BRDF data, as BRDFs are being evaluated directly on the fly inside a fragment shader. To avoid undersampling artifacts, we constrain the size of each visibility cluster to be small enough such that the BRDF variation within each cluster can be ignored. We provide the theoretical upper bound of the approximation error introduced by this simplification.
- **Incorporation of direct-to-indirect transfer.** We can easily integrate precomputed direct-to-indirect transfer [7] into our system by representing local direct illumination using uniform samples around the scene surface. This enables accurate simulation of indirect lighting effects with fully dynamic glossy BRDFs.
- **GPU-based rendering.** We accelerate our rendering algorithms using GPU shaders that access non-uniform precomputed data on the fly. This provides significant speedup over an equivalent CPU implementation. We demonstrate realistic relighting effects simulated at 15 ~ 30 fps under fully dynamic lighting, viewpoint and BRDFs. Figure 1 shows several examples captured in real-time using our system.

2. Related Work

Illumination from environment maps. Realistic image synthesis often uses large-scale detailed environment maps [5] to illuminate a scene. Standard Monte Carlo ray tracing performs poorly for such large illumination sources. Wavelet importance sampling [4] brings significant speedup but still relies on an offline renderer. Various importance sampling schemes [9, 1] partition environment maps into pre-integrated directional lights to reduce rendering costs; however, these partitioning schemes are purely

light-dependent, and are typically very slow to compute thus have to be performed offline.

Many environment mapping techniques approximate illumination and BRDFs using a pre-selected basis set, such as spherical harmonics [16, 17, 8] and wavelets [26]. These methods simplify the illumination computation as inner products of low dimensional vectors, but require expensive preprocessing of the BRDF and ignore shadowing effects.

Precomputed radiance transfer (PRT). PRT was first introduced in [20, 14] as a general technique for rendering real-time global illumination effects under large-scale environment lighting. PRT projects precomputed illumination data onto a basis set such as spherical harmonics (SH) or wavelets, then linearly combines the results to synthesize new images under novel lighting. Subsequent work extended PRT toward the incorporation of all-frequency view-dependent effects [12, 27, 15, 13, 6], local lighting effects [31, 7, 10], and dynamic scene models [21, 18].

Despite recent advances, several key challenges remain in PRT. First, existing methods often address complex glossy BRDFs by bandlimiting high frequency reflections using a low-order projection basis [12, 28]. The quality can be improved using fast triple product wavelet integrals [15], but at the cost of very large preprocessed BRDF data. Online filtering of environment maps [6] provides an alternative, but at the cost of reduced quality for general BRDFs and expensive run-time filtering for truly dynamic lighting. Second, online changing of dynamic BRDFs remains a greater challenge. Several recent work [3, 2, 22] build interactive BRDF editing systems, but they either require fixed lighting and viewpoint, or require the incorporation of BRDF basis into precomputation, resulting in very large datasets. Xu et al [30] use spherical piecewise constant basis to permit fast online switching of BRDFs, but still require presampled BRDF data. Finally, many existing PRT systems assume distant (purely directional) environment lighting, which has limited their usage in local lighting scenarios. A recent work by Hasan et al [7, 10] pre-

sented a precomputed direct-to-indirect transfer method for simulating interreflection effects under local lighting. They use unstructured points as illumination samples and apply 2D wavelets to compress the precomputed datasets. Their method requires fixing viewpoint and BRDFs at precomputation. A related work by Kontkanen et al [10] uses a 4D wavelet basis but requires parameterized models.

In contrast to existing systems, we permit fully dynamic material editing with no preprocessed BRDF data; furthermore, our algorithm can naturally integrate precomputed direct-to-indirect transfer effects without either fixing the BRDF and viewpoint or requiring parameterized models.

3. Algorithms and Implementation

In this section we describe our algorithms in detail. We first describe the algorithm for environment lighting with only direct illumination effects, then extend the same algorithm to local lighting with precomputed direct-to-indirect transfer effects.

3.1. Environment Lighting

To simplify the discussion, we start with diffuse BRDFs. In this case, the vertex shading color B under direct illumination $L(\omega)$ is computed by:

$$B = k_d \int_{\Omega} L(\omega) V(\omega) \cos \theta d\omega = k_d \int_{\Omega} L(\omega) \tilde{V}(\omega) d\omega \quad (1)$$

where k_d is the diffuse BRDF, $\cos \theta$ is the incident cosine (clamped to 0), and V is the binary visibility. We combine the latter two into a single term $\tilde{V}$, representing the cosine-weighted visibility. Typically $\tilde{V}$ has large coherent regions that can be clustered and approximated using a single average value per cluster, allowing us to re-write Eq. 1 as:

$$B \approx k_d \sum_k v_k \int_{\Omega_k} L(\omega) d\omega = k_d \sum_k v_k l_k \quad (2)$$

where Ω_k represents the domain of cluster k , l_k is the integral of the lighting within Ω_k , and v_k is the clustered visibility value, representing the average of $\tilde{V}$ within Ω_k :

$$v_k = \frac{1}{|\Omega_k|} \int_{\Omega_k} \tilde{V}(\omega) d\omega$$

Since function $\tilde{V}$ is assumed to be coherent within each cluster, we could as well take an arbitrary sample $\tilde{V}(\omega_k)$ from the cluster to be v_k ; however, computing v_k as an average minimizes the approximation error in a least square sense.

Illumination samples. To account for lighting effects at all frequencies, we choose to represent lighting L using a

Figure 2. Our algorithm works with unparameterized illumination sample points.

Figure 3. Precomputing visibility cuts.

large number of directional samples ($32 \sim 64$ thousand). This sample set represents a discretization of the entire domain Ω and can be carefully precomputed, such as using a point repulsion algorithm [24], to guarantee uniform distribution in a statistical sense. By doing so, we can completely avoid uneven sampling of environment maps that is inevitable with traditional parameterized schemes such as cubemaps or octahedron maps. Figure 2(a) shows a smaller set of directional samples we generated on a unit sphere.

Precomputed visibility cuts. Given the illumination samples, we can then precompute the per-vertex visibility function $\tilde{V}$ by utilizing a ray tracer to cast shadow rays from every vertex to all sampled directions. To cluster the visibility samples, we build a spatial hierarchical structure – a binary tree – from the samples. The tree building process is similar to the Lightcuts [25] and is only dependent on the sample points’ spatial locations. We then use the tree to compute a clustering of visibility values: the leaf nodes represent individual samples, and each interior node represents the cluster average of all children samples underneath it in the tree. In addition to the cluster average, each node also stores the L^2 approximation error associated with the clustering. Note that the same tree structure is used globally for every vertex, only that the value stored at each tree node varies from vertex to vertex.

Next, we find a cut through the tree, consisting of a col-

lection of tree nodes such that every path from the root to the leaf passes through exactly one node from the cut. We call this a *precomputed visibility cut*. Each cut represents a unique partitioning of $\tilde{V}$ into clusters, and each different cut results in a different approximation error.

The cut is computed using a progressive refinement algorithm similar to [25]. We start from the root node of the tree, and repeatedly replace a node in the current cut with its two children nodes, until the L^2 approximation error associated with each node falls below a predefined threshold ε . In other words, the refinement step terminates when the following criteria is satisfied for every cut node k :

$$\frac{1}{|\Omega_k|} \int_{\Omega_k} |v_k - \tilde{V}(\omega)|^2 d\omega \leq \varepsilon^2 \quad (3)$$

We typically pick $\varepsilon = 2 \sim 3\%$, which gives no noticeable artifacts in all our experiments. The selected cut is then stored as a sparse vector containing each cut node's index into the tree as well as the cluster average v_k .

Relighting. Our relighting algorithm is straightforward and similar to other PRT systems. We first sample dynamic environment lighting onto the illumination samples used in precomputation, then apply the pre-built binary tree structure to quickly group the lighting into clusters that we call the *light tree*. Next, we compute the shading color of each vertex (Eq. 2) as a sparse vector inner product of the pre-computed visibility cuts with the light tree. This computation takes use of the index and visibility value stored at each cut node. We use modern graphics hardware to accelerate this step, which is discussed in detail in Section 3.3.

Incorporating glossy BRDFs. To incorporate general BRDFs into Eq. 2, we make an assumption that the size of each cluster Ω_k is small enough such that the BRDF variation within the cluster is very smooth and can be ignored. This makes it possible to factor the BRDF term out of the lighting integral, and replace it instead with a single sample evaluated using each cluster's representative direction. Thus the view-dependent shading color is computed by:

$$B(\omega_o) = \sum_k v_k l_k f_r(\omega_k, \omega_o) \quad (4)$$

where f_r is the BRDF, and ω_k is the cluster Ω_k 's representative direction, which can be computed either as a simple average of all the sampled directions underneath that cluster, or weighted by the intensity of the run-time lighting, thus favoring the results toward more important regions of the lighting.

The assumption about small BRDF variations requires us to confine the size of each visibility cluster in precomputation. Thus we add a second criteria in the cut selection algorithm to limit the solid angle represented by each cut

node below a predefined threshold:

$$|\Omega_k| \leq \varepsilon_\Omega \quad (5)$$

Thus the progressive refinement step terminates only when both Eq. 3 and 5 are satisfied. Our typical value for ε_Ω is $\frac{1}{256}$ of the entire domain 4π , which is sufficiently accurate for many commonly seen BRDFs.

The relighting algorithm is only slightly modified, by adding a BRDF term that is to be evaluated directly on the fly using ω_k and the dynamic view direction ω_o . The key advantage of this approach is that it permits real-time BRDF manipulation without any preprocessed BRDF data. This feature is crucial for applications such as architectural or interior design. The main limitation is that highly reflective BRDFs can contain rapid changes that violate our assumption. To address this problem, we could send multiple BRDF samples per cluster at run-time to better account for high frequency changes. Alternatively, we can use the error upper bound derived in the following to guide precomputation more carefully. Because BRDF evaluation is costly to compute, the use of modern GPU to accelerate rendering is critical for maintaining interactive relighting speed.

Error bounds. The error in relighting can be bound using the predefined error threshold we applied in precomputation. In the diffuse case, the L^2 error in relighting contributed by each cluster is:

$$\begin{aligned} E_{\Omega_k} &= k_d^2 |v_k l_k - \int_{\Omega_k} L(\omega) \tilde{V}(\omega) d\omega|^2 \\ &= k_d^2 \left| \int_{\Omega_k} L(\omega) (v_k - \tilde{V}(\omega)) d\omega \right|^2 \\ &\leq k_d^2 \int_{\Omega_k} L^2(\omega) d\omega \cdot \int_{\Omega_k} |v_k - \tilde{V}(\omega)|^2 d\omega \quad (6) \\ &\leq k_d^2 \int_{\Omega_k} L^2(\omega) d\omega \cdot |\Omega_k| \varepsilon^2 \quad (7) \end{aligned}$$

where step (6) applies the Cauchy-Schwarz inequality, and step (7) applies the termination criteria in Eq. 3. Summing up the error for all clusters, the total *normalized* L^2 error (the ratio of E_{Ω_k} to the light energy $\int_{\Omega_k} L^2$) of the entire domain Ω is bound by:

$$\text{normalized } E_\Omega \leq k_d^2 |\Omega| \varepsilon^2 = 2\pi k_d^2 \varepsilon^2$$

where 2π comes from the fact that only half of Ω contributes nonzero error. The error bound for general BRDFs is more complicated. However, if we assume the BRDF variation within each Ω_k is small and bound by ε_ρ , such that

$$|\rho(\omega, \omega_o) - \rho_k(\omega_k, \omega_o)| = |\rho(\omega, \omega_o) - \rho_k| \leq \varepsilon_\rho \quad (8)$$

is true for all ω 's in that cluster, we can prove (see Appendix A) that the L^2 error in relighting is bound by:

$$E_{\Omega_k} \leq \int_{\Omega_k} L^2(\omega) d\omega \cdot |\Omega_k| (\rho_k^2 \varepsilon^2 + \varepsilon_\rho^2 + 2\rho_k \varepsilon_\rho)$$

Comparison with wavelet basis projection. Conceptually our approach is analogous to nonlinear wavelet approximation methods [14, 7] in that they both adaptively pick the best set of approximators to represent per-vertex visibility/transport functions. The primary difference is that we use an arbitrary point set to sample the illumination, and each subdomain is guaranteed to be small enough such that the BRDF term can be decoupled from the lighting integral. In contrast, wavelets usually require a regular parameterized domain, thus are very difficult to provide uniform sampling of the illumination or work with unparameterized domains. In addition, the lowest level wavelet bases, which are often of the highest importance, have wide support and violate our BRDF assumption, thus they do not allow the decoupling of BRDFs from the lighting integral.

3.2. Indirect Transfer under Local Lighting

Because our algorithm works with arbitrary illumination samples that may be unparameterized, we can easily adapt it to incorporate one bounce of precomputed direct to indirect transfer [7], permitting complex indirect lighting effects under local illumination scenarios. We make two assumptions here: 1) we assume that direct illumination can be quickly computed using an existing GPU technique such as shadow mapping; 2) we only consider one bounce of diffuse to glossy indirect transfer, ignoring multiple inter-reflection passes or transfer paths that start from a glossy surface. Simulating multiple indirect transfer with dynamic BRDFs is itself an open research problem [2, 22].

Illumination samples. Similar to environment lighting, we choose to represent local lighting using a large number of sample points x_i , which are now placed locally around the scene. These sample points are precomputed with uniform distribution. Figure 2(b) shows an example.

Formulation. We still use diffuse BRDF as a starting case, as the general case can be accounted for by adding a BRDF term during rendering. In this case, the indirect component of the vertex shading color is computed by:

$$B(x_o) = k_d \int_A L_d(x_i) \tilde{V}(x_i \rightarrow x_o) dA(x_i) \quad (9)$$

where the integration is over the entire surface of the scene; L_d is the diffuse radiance resulting from direct illumination and is computed dynamically on the fly; and

$$\tilde{V}(x_i \rightarrow x_o) = V(x_i \rightarrow x_o) \frac{\cos \theta_i \cdot \cos \theta_o}{|x_i - x_o|^2}$$

is the precomputed visibility term weighted by the differential form factor between an illumination sample x_i and a shading point x_o .

Visibility cuts. As before, $\tilde{V}$ contains large coherent regions that can be exploited by clustering. To do so, we build a binary tree structure of the illumination sample points x_i ; the construction considers similarity between sample points in both spatial location and normal. We then sample visibility functions at every vertex, and use the same tree structure to quickly compute a cut through the tree, imposing the two termination criteria described previously. The cut represents a piecewise constant approximation of the visibility function, and is stored as a sparse vector. Error analysis can be similarly derived as before.

Relighting. At run-time, we first compute the diffuse direct shading colors L_d at all illumination samples. This step takes use of any existing GPU technique such as shadow mapping or shadow volume. Local area lights can be included using shadow fields [31] at a higher computation and storage cost. Following this step, we cluster L_d values into a light tree using the same binary tree structure built in pre-computation, then use a GPU algorithm to compute the per-vertex indirect shading color. As before, this step accesses the precomputed visibility cuts, the light tree, and evaluates BRDFs directly on the fly. Finally, a hardware rendering pass computes the per-pixel direct shading color, sums it up with the per-vertex indirect shading color, and displays the final global illumination result onto the screen.

3.3. Implementation Details

Precomputing visibility cuts. We represent the illumination using 32,000 $\sim$ 64,000 uniformly distributed sample points, and build a binary tree from them. For environment lighting, these are points sampled on a unit sphere representing directional lights; for local lighting, these are samples placed locally over the scene surface. The per-vertex visibility function is first evaluated using a ray tracer, then clustered using the binary tree structure (an example is shown in Figure 3). During this process, each tree node stores the cluster average v_k , the aggregated solid angle $|\Omega_k|$ subtended by the cluster, and also the L^2 error associated with the approximation. The integral differential constant $d\omega$ is also included in this computation.

Next, we find a visibility cut through the tree by using the iterative refinement algorithm and the termination criteria described in Section 3.1. The selected cut nodes are then stored as a sparse vector using 16-bit integer indices and 16-bit floating point values. Cut nodes with zero values are discarded and never stored. Since the BRDF is not baked into precomputation, our datasets are reasonably compact. Standard spatial compression schemes such as CPCA [19] can apply to further reduce the data size.

GPU-based relighting algorithm. Our relighting algorithm is analogous to wavelet-based PRT that computes

Figure 4. Diagram of our GPU relighting pipeline. All data are stored as 16-bit floating point textures.

sparse vector inner products on the fly. Because evaluating BRDF is very costly, we found it is critical to leverage modern graphics hardware to accelerating rendering and achieve interactive performance. Previous work has already shown GPU-based implementation for wavelet-based PRT relighting [28]. This work takes a step further by utilizing several advanced features available in the latest nVIDIA G80 graphics hardware. Our early implementation in [28] required a fixed number (128) of data elements at each vertex such that they can be laid out in uniform 16×8 blocks and later downsampled to compute the relighting results. The major improvement in this work is that we represent the per-vertex precomputed visibility cuts on the GPU using their native 1D layout, and use dynamic fragment shader loops to compute the relighting results without reduction. This approach permits non-uniform data formats, thus providing significant improvements in accuracy and flexibility.

Figure 4 shows a high-level diagram of our relighting pipeline. All data are stored as 16-bit floating point textures on card. Shading vertices and the dynamic light tree are both stored as square 2D textures, thus indexing into these data can be performed by a texture access using 2D coordinates. We pack precomputed visibility cuts data into a large texture such that each cut is stored continuously in a single row, and no cut is split into two rows. This scheme can leave 'holes' at the end of each row, and we use a simple heuristics to optimize the packing and reduce fragmentation. Due to the limit in texture size (4096 in each dimension), a single 2D texture is insufficient to store the entire precomputed dataset. We therefore choose to use the floating point 3D texture format to store the data. Because the data are at non-uniform lengths, we also need a 'pointer' texture that keeps track of the starting and ending elements of each vertex's data within the entire visibility texture.

To relight, we first update the light tree textures, then rasterize a quad representing the domain of all vertices to be shaded. Next, we bind a fragment shader that accesses

the data pointer texture to find out the starting and ending elements of the visibility cut, then loops over these elements and sums up the triple products of the cluster average v_k , the clustered lighting l_k , and the BRDF sample f_r . The BRDF is evaluated directly using the average cluster direction ω_k and the view direction ω_o . As a single fragment loop cannot exceed 256 iterations, we break up the computation into blocks of 256 iterations each. The final shading results B are then copied into a vertex buffer to be display as vertex colors.

4. Results and Discussion

This section presents our results. Our programs are benchmarked on an Intel Core 2 Duo 2.0GHz computer with 2GB memory and an nVIDIA 8800 GTX graphics card. The precomputation algorithm utilizes a standard ray tracer that runs on both CPU cores. During relighting, the CPU computes and builds the light tree, and the remaining algorithm is performed entirely on the GPU.

We test our algorithms with several standard 3D models and a scanned horse statue model. The BRDFs used in our experiments include diffuse, Phong, anisotropic Ward [29], and several Lafortune models [11].

Precomputation. The precomputation results are summarized in Table 1. For all test scenes, the precomputation generates reasonable storage footprints, an important factor for maintaining fast relighting speed. The average number of clusters (or the *cut size*) directly relates to the relighting performance, and we show a false colored visualization of the per vertex cut size in Figure 5. The left image shows the cut selection result by applying only the approximation criteria (Eq. 3) but not the solid angle criteria (Eq. 5); the right images shows the result when both criteria are applied. This forces cuts to be further subdivided even at large coherent regions of the visibility, thus guaranteeing that our BRDF

			Environment lighting (Direct)				Local lighting (Indirect)			
	Vertices	Triangles	Samples	Time	Storage	Cut size	Samples	Time	Storage	Cut size
Horse	60,893	120,716	65,536	50 min	77 MB	320	-	-	-	-
Bird	61,555	122,408	32,768	35 min	70 MB	284	32,768	25 min	50 MB	204
Buddha	87,365	174,126	32,768	35 min	120 MB	340	32,768	22 min	140 MB	400
Bunny	72,027	144,046	32,768	28 min	89 MB	306	-	-	-	-
Plank	61,807	122,836	-	-	-	-	65,536	50 min	120 MB	485

Table 1. Precomputation benchmark for five test scenes. For each model, we list the number of illumination samples, the precomputation time and storage size, and the average number of clusters (cut size) per vertex. The average cut size relates linearly to the relighting cost.

Figure 5. Visualization of the cut size before and after applying the solid angle constraint.

assumption will be met. Because we never store cut nodes with zero visibility, vertices of high occlusion tend to have significantly smaller cut sizes.

Relighting. The relighting performance is summarized in Table 2. For all test scenes, we maintain interactive relighting speed under arbitrary illumination, viewpoint, and material changes. Upon lighting change, the CPU performs the illumination sampling and builds the light tree. It is possible to move this computation to the GPU too. Figure 6 shows two examples of the Buddha model rendered with different environment lighting and dynamic BRDFs.

For local lighting, we bind a GPU shadow mapper to evaluate the direct illumination at all sample points; the results are then downloaded to the CPU to build the light tree. Figure 7 compares renderings with only direct lighting vs. global illumination including one bounce of indirect transfer. Note how indirect lighting adds realistic convincing effects to the final rendering. Figure 10 shows dynamic manipulation of indirect lighting effects produced by our system. Note the change in glossy interreflection due to the viewpoint and material changes. These highly glossy interreflection effects are very difficult to handle by previous basis projection methods.

	Sampling Light (CPU)		Relighting (GPU)	
	Env. map	Local	Env. map	Local
Horse	70 fps	-	19 fps	-
Bird	125 fps	110 fps	35 fps	34 fps
Buddha	125 fps	100 fps	15 fps	13 fps
Bunny	125 fps	-	27 fps	-
Plank	-	120 fps	-	14 fps

Table 2. Relighting benchmark for both environment lighting and local lighting, itemized by the illumination sampling speed and the GPU-based relighting speed.

Dynamic BRDF editing. Our system enables fully dynamic BRDF editing. Figure 11 shows two examples featuring online editing of the Phong exponent parameter, and the Ward anisotropic parameters. Combined with large-scale environment lighting and complex shadows, these effects provide the user with realistic real-time feedbacks of arbitrary material design under final quality rendering.

Accuracy. The relighting accuracy is primarily determined by two sources: the visibility approximation error, and the error resulting from our smooth BRDF assumption. The former is bound by our pre-selected error threshold ϵ , which is typically picked as $2 \sim 3\%$. This is an error upper bound that is guaranteed by our cut selection algorithm. Changing this threshold can be used to vary the average cut size, thus providing a tradeoff between accuracy and performance. Figure 9 shows our experiments with varying ϵ 's together with a comparison of the rendering quality.

The BRDF associated error can be adjusted by varying the predefined solid angle threshold ϵ_Ω , which has a default value of $\frac{4\pi}{256}$ in our design. This means that the solid angle subtended by each cluster is guaranteed to be less than $\frac{1}{256}$ of the entire domain. A smaller threshold allows for more accurate simulation of highly glossy BRDFs; however, it can also dramatically increase the average cut size thus reducing the relighting speed. An accurate analysis of this

Figure 6. The Buddha model rendered with dynamic lighting and BRDFs.

error remains our future work. Figure 8 provides a qualitative analysis by varying the glossiness parameter of a Phong BRDF applied to a precomputed horse model. Note how the quality of glossy highlights degrades as the BRDF becomes sharper. The error is manifested by banding artifacts in the high frequency reflections.

5. Conclusion

In conclusion, we have presented a novel method based on precomputed visibility cuts for interactive relighting under all-frequency environment maps and arbitrary dynamic BRDFs. Our approach does not require a parameterized illumination domain, therefore it works with arbitrary lighting scenarios and guarantees uniform sampling in a statistical sense. In addition, it requires no precomputed BRDF data, thus allowing for fully dynamic BRDF editing on the fly. Finally, we have shown that this approach can be easily extended to support one bounce of glossy indirect transfer.

In future work, we expect to extend the system to support per-pixel rendering, thus permitting complex effects such as bump mapping and spatially varying BRDFs. The biggest challenge is how to interpolate sparse vectors in a fragment shader. In addition, we plan to explore the use of this algorithm in more complex lighting effects such as translucency, by representing illumination inside the object using unstructured points.

Acknowledgement The authors would like to thank John Tran and Ewen Cheslack-Postava for providing the point sampling and tree building algorithms, Peter Shirley for the initial ray tracer code used in this paper, Paul Debevec for

(a) Direct lighting only (b) Direct + 1-bounce indirect

Figure 7. Direct vs. global illumination.

his light probe images, and the Stanford Graphics group for the 3D models.

Appendix A

Proof of relighting error bound with general BRDFs:

$$\begin{aligned}
 E_{\Omega_k} &= |v_k l_k \rho_k - \int_{\Omega_k} L(\omega) \tilde{V}(\omega) \rho(\omega, \omega_o) d\omega|^2 \\
 &= | \int_{\Omega_k} L(\omega) (v_k \rho_k - \tilde{V}(\omega) \rho(\omega, \omega_o)) d\omega |^2 \\
 &\leq \int_{\Omega_k} L^2(\omega) d\omega \cdot \int_{\Omega_k} |v_k \rho_k - \tilde{V}(\omega) \rho(\omega, \omega_o)|^2 d\omega
 \end{aligned}$$

The second term in the above equation can be bound by

$$\begin{aligned}
 \int |v_k \rho_k - \tilde{V}(\omega) \rho(\omega)|^2 &= \\
 \int |(v_k - \tilde{V}(\omega))\rho_k + \tilde{V}(\omega)(\rho_k - \rho(\omega))|^2 &\leq \\
 |\Omega_k|(\rho_k^2 \epsilon^2 + \epsilon_p^2 + 2\rho_k \epsilon_p) &
 \end{aligned}$$

where the last step is derived by expanding the quadratic terms, then applying the approximation criteria (Eq. 3), the BRDF bound in Eq. 8, and using the fact that $\tilde{V}(\omega) \in [0, 1]$. Note that due to the energy conservation law of a physical BRDF, the term $|\Omega_k| \rho_k$ converges when summed up across the entire domain Ω .

References

- [1] S. Agarwal, R. Ramamoorthi, S. Belongie, and H. W. Jensen. Structured importance sampling of environment maps. In *Proceedings of SIGGRAPH '03*, pages 605–612, 2003.

Figure 8. Due to our assumption that BRDFs remain smooth within each cluster, artifacts can begin to occur as we increase the glossiness of a Ward anisotropic BRDF. To improve, we could use a smaller solid angle threshold ε_Ω during precomputation, but at a higher storage and relighting cost.

Figure 9. Average cut size and the rendering quality resulting from varying the visibility approximation threshold ε . In order to highlight the effect of changing ε , the solid angle threshold ε_Ω is not applied here. Note the differences in the shadow boundaries and glossy highlights.

- [2] A. Ben-Artzi, K. Egan, F. Durand, and R. Ramamoorthi. A precomputed polynomial representation for interactive brdf editing with global illumination. *ACM Transactions on Graphics (to appear)*.
- [3] A. Ben-Artzi, R. Overbeck, and R. Ramamoorthi. Real-time BRDF editing in complex lighting. *ACM Transactions on Graphics*, 25(3):945–954, 2006.
- [4] P. Clarberg, W. Jarosz, T. Akenine-Möller, and H. W. Jensen. Wavelet importance sampling: efficiently evaluating products of complex functions. *ACM Transactions on Graphics*, 24(3):1166–1175, 2005.
- [5] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH '97*, pages 369–378, 1997.
- [6] P. Green, J. Kautz, W. Matusik, and F. Durand. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *ACM Symposium on Interactive 3D graphics*, pages 7–14, 2006.
- [7] M. Hašan, F. Pellacini, and K. Bala. Direct-to-indirect transfer for cinematic relighting. *ACM Transactions on Graphics*, 25(3):1089–1097, 2006.
- [8] J. Kautz, P.-P. Sloan, and J. Snyder. Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics. In *Proceedings of the 13th Eurographics Symposium on Rendering*, pages 291–296, 2002.
- [9] T. Kollig and A. Keller. Efficient illumination by high dynamic range images. In *Proceedings of the 14th Eurographics symposium on Rendering*, pages 45–50, 2003.
- [10] J. Kontkanen, E. Turquin, N. Holzschuch, and F. Sillion. Wavelet radiance transport for interactive indirect lighting. In *Proceedings of Eurographics Symposium on Rendering*, 2006.
- [11] E. P. Lafortune, S.-C. Foo, K. E. Torrance, and D. P. Greenberg. Non-linear approximation of reflectance functions. In *Proceedings of SIGGRAPH '97*, pages 117–126, 1997.
- [12] X. Liu, P.-P. Sloan, H.-Y. Shum, and J. Snyder. All-frequency precomputed radiance transfer for glossy objects. In *Proceedings of the 15th Eurographics Symposium on Rendering*, pages 337–344, 2004.
- [13] W.-C. Ma, C.-T. Hsiao, K.-Y. Lee, Y.-Y. Chuang, and B.-Y. Chen. Real-time triple product relighting using spherical

Figure 10. Dynamic indirect lighting effects.

(a) Ward $ax=0.1, ay=0.3$

(b) Ward $ax=0.3, ay=0.1$

(c) Phong $exp=10$

(d) Phong $exp=50$

Figure 11. Dynamic BRDF editing of Ward anisotropic BRDF and Phong BRDF. Note the differences in anisotropy and specularity.

local-frame parameterization. In *Proceedings of the 14th Pacific Graphics*, pages 682–692, 2006.

- [14] R. Ng, R. Ramamoorthi, and P. Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics*, 22(3):376–381, 2003.
- [15] R. Ng, R. Ramamoorthi, and P. Hanrahan. Triple product wavelet integrals for all-frequency relighting. *ACM Transactions on Graphics*, 23(3):477–487, 2004.
- [16] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of SIGGRAPH 2001*, pages 497–500, 2001.
- [17] R. Ramamoorthi and P. Hanrahan. Frequency space environment map rendering. In *ACM Transactions on Graphics*, volume 21, pages 517–526, 2002.
- [18] Z. Ren, R. Wang, J. Snyder, K. Zhou, X. Liu, B. Sun, P.-P. Sloan, H. Bao, Q. Peng, and B. Guo. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Transactions on Graphics*, 25(3):977–986, 2006.
- [19] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics*, pages 382–391, 2003.
- [20] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics*, volume 21, pages 527–536, 2002.
- [21] P.-P. Sloan, B. Luna, and J. Snyder. Local, deformable precomputed radiance transfer. *ACM Transactions on Graphics*, 24(3):1216–1224, 2005.
- [22] X. Sun, K. Zhou, Y. Chen, S. Lin, J. Shi, and B. Guo. Interactive relighting with dynamic brdfs. In *Proceedings of SIGGRAPH 2007 (to appear)*, 2007.
- [23] Y.-T. Tsai and Z.-C. Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clus-

tered tensor approximation. *ACM Transactions on Graphics*, 25(3):967–976, 2006.

- [24] G. Turk. Re-tiling polygonal surfaces. In *Proceedings of SIGGRAPH '92*, pages 55–64, 1992.
- [25] B. Walter, S. Fernandez, A. Arbree, K. Bala, M. Donikian, and D. P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Trans. Graph.*, 24(3):1098–1107, 2005.
- [26] R. Wang, R. Ng, D. Luebke, and G. Humphreys. Efficient wavelet rotation for environment map rendering. In *Proceedings of the 17th Eurographics Symposium on Rendering*, 2006.
- [27] R. Wang, J. Tran, and D. Luebke. All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Proceedings of the 15th Eurographics Symposium on Rendering*, pages 345–354, 2004.
- [28] R. Wang, J. Tran, and D. Luebke. All-frequency relighting of glossy objects. *ACM Transactions on Graphics*, 25(2):293–318, 2006.
- [29] G. J. Ward. Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH '92*, pages 265–272, 1992.
- [30] K. Xu, Y.-T. Jia, H. Fu, S.-M. Hu, and C.-L. Tai. Spherical piecewise constant basis functions for all-frequency precomputed radiance transfer. *IEEE Transaction on Visualization and Computer Graphics (under revision)*.
- [31] K. Zhou, Y. Hu, S. Lin, B. Guo, and H.-Y. Shum. Precomputed shadow fields for dynamic scenes. *ACM Transactions on Graphics*, 24(3):1196–1201, 2005.