

Sparsely Precomputing The Light Transport Matrix for Real-Time Rendering

*Fu-Chung Huang
Ravi Ramamoorthi*



Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-79

<http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-79.html>

May 14, 2010

Copyright © 2010, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Acknowledgement

This work was supported in part by NSF CAREER grant IIS-0924968 and ONR PECASE grant N00014-09-1-0741, as well as equipment and generous support from Intel, NVIDIA, Adobe, and Pixar.

Sparsely Precomputing The Light Transport Matrix for Real-Time Rendering

Fu-Chung Huang and Ravi Ramamoorthi

University of California at Berkeley, USA

Abstract

Precomputation-based methods have enabled real-time rendering with natural illumination, all-frequency shadows, and global illumination. However, a major bottleneck is the precomputation time, that can take hours to days. While the final real-time data structures are typically heavily compressed with clustered principal component analysis and/or wavelets, a full light transport matrix still needs to be precomputed for a synthetic scene, often by exhaustive sampling and raytracing. This is expensive and makes rapid prototyping of new scenes prohibitive. In this paper, we show that the precomputation can be made much more efficient by adaptive and sparse sampling of light transport. We first select a small subset of “dense vertices”, where we sample the angular dimensions more completely (but still adaptively). The remaining “sparse vertices” require only a few angular samples, isolating features of the light transport. They can then be interpolated from nearby dense vertices using locally low rank approximations. We demonstrate sparse sampling and precomputation $5\times$ faster than previous methods.

1. Introduction

Precomputation-based rendering, or precomputed radiance transfer (PRT), has enabled real-time image synthesis with natural lighting, intricate shadows, and global illumination effects [SKS02, NRH03]. However, the initial precomputation remains a bottleneck, since the full light transport matrix needs to first be precomputed, corresponding to each vertex and light source direction. To include interreflection effects, exhaustive sampling and ray or path tracing is typically required. For an all-frequency lighting cubemap resolution of $6 \times 32 \times 32$, the cost is essentially equivalent to rendering 6144 images, and can take hours to days. This precludes rapidly prototyping new scenes, and hinders adoption.

The final data structure for real-time rendering is highly compressed, such as with wavelets or clustered principal component analysis (CPCA) [SHHS03, LSSS04], but these methods first require the full light transport to be available. This paper investigates fast precomputation by adaptive and sparse sampling of light transport. While our technique is simple and broadly applicable to almost any PRT system, we focus on all-frequency relighting of static geometry, including interreflection effects, where the precomputation is by explicit sampling and ray tracing. This includes both the diffuse geometry relighting of Ng et al. [NRH03], and its extension to glossy materials with BRDF in-out factoriza-

tion [LSSS04, WTL04]—in the latter case, our method is simply applied to each view-independent transport term.

We leverage key recent insights about the structure of light transport. CPCA [SHHS03] is based on assuming that locally, the response of vertices is similar, and of low rank [MKS RB07]. Hence, we first compute a small subset of “dense” vertices, where almost all angular directions are sampled. The remaining “sparse” vertices require only a few light source directions to be computed, and can then be reconstructed with a low rank approximation from their neighbors. We also know that light transport is sparse [NRH03, PML*09]. We exploit this idea by focusing our sampling on those angular directions that correspond to features, and by using recent sparse L_1 minimization methods [KKL*07]. Our algorithm includes simple heuristics to choose the best candidate “dense” spatial vertices and angular features.

As shown in Fig. 1, we can produce accurate results with explicit precomputation of only about 11% of the light transport matrix, and with acceptable overheads (wall clock time is 20% of brute force precomputation). These results potentially enable new capabilities for rapid prototyping of scenes, or shots for lighting design.

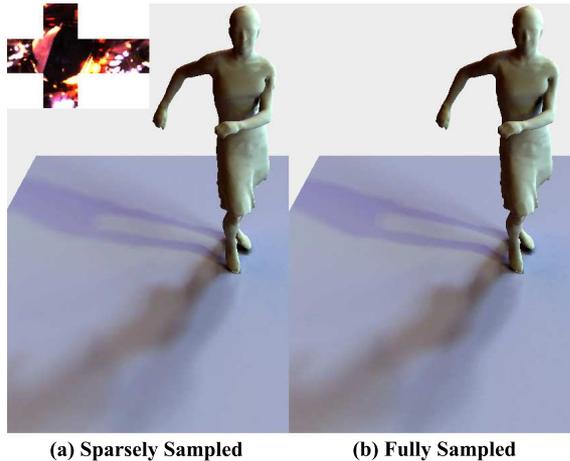


Figure 1: Our method (a) sparsely samples the light transport matrix for precomputation in PRT methods. The rendering results, including diffuse global illumination, are almost identical to fully sampled transport (b). In this example, involving static diffuse geometry, we used only 11% of the samples, and 20% of the precomputation time of full sampling. Relighting uses all-frequency $6 \times 32 \times 32$ cubemaps.

2. Related Work

Precomputation-Based Rendering has a long history, going back to early work by [NSD94, DAG95]. The seminal paper by Sloan et al. [SKS02], and the all-frequency extensions of Ng et al. [NRH03, NRH04] have led to increased interest and many applications. We refer to [Ram09] for a survey of many recent developments. All of these approaches have focused on real-time functionality, relying on brute force precomputation of light transport. [NRH03] introduced a hardware-accelerated rasterization method to precompute visibility, but only for direct lighting. The alternative ray or path-tracing approach is needed for global illumination, and can be very expensive.

Row-Column Sampling: In recent years, many other fields of rendering have recognized the need for sparsely and adaptively computing variants of light transport. For offline rendering, matrix row-column sampling [HPB07] has been proposed, and the Kernel Nystrom method has been developed for appearance acquisition [WDT*09]. These methods also relate to numerical matrix decompositions like Drineas et al. [DMM08]. We are inspired by these techniques, but consider a different application, to precomputation-based relighting. In that context, we demonstrate better methods to choose angular samples, reconstruct locally low rank approximations, and compute clustered principal components on sparse data. Moreover, we compute each element of the matrix separately, and so do not use full rows and columns.

Compressive Sensing: Another body of work is compressive sensing [CT06, CRT06], that has recently been used for appearance capture [PML*09, SD09]. Random patterns are projected, with sparse minimization. We cannot use these methods directly, since we must sample individual elements from the light transport matrix by ray or path tracing—the

cost of computing the projection of a random pattern will be proportional to the support of the pattern. However, we are inspired by the sparsity demonstrated, choosing our angular samples at locations identified as features. We also leverage the L_1 sparse minimization approach advocated by these methods, using the recent fast algorithm of [KKL*07].

Hierarchical and Sparse Sampling: Methods like irradiance caching and variants allow sparse sampling and interpolation of global illumination data over a scene [KG09]. Recent hierarchical approaches to light transport include Lehtinen et al. [LZT*08], Kontkanen et al. [KTHS06] and Hasan et al. [HPB06], which have also been applied for direct-to-indirect transfer for PRT. We differ in focusing on all-frequency relighting, where the light transport signal includes direct illumination, and varies more rapidly in both spatial and angular domains. Thus, the hierarchical and sparse sampling methods are not directly applicable (much as they are rarely used for direct lighting in global illumination). Our approach can effectively be seen as a hierarchical sparse precomputation for higher-frequency signals.

Adaptive Remeshing: A closely related work is Krivanek et al. [KPZ04], who adaptively subdivide a mesh for diffuse PRT, based on the change in the full transport operator. This work has similarities to adaptive image sampling [Guo98]. Our method is orthogonal to theirs, in using a fixed geometric mesh, but sparsely precomputing light transport. We also focus on all-frequency relighting, enable glossy materials, and integrate our method closely with CPCA. The approaches could be combined in future to allow sparse precomputation along with an adaptive mesh simplification.

3. Overview

Following Ng et al. [NRH03], light transport can be written,

$$\mathbf{B} = \mathbf{T}\mathbf{L}, \quad (1)$$

where \mathbf{B} is the outgoing radiance or image, \mathbf{L} is the incident lighting (a vector corresponding to different directions), and \mathbf{T} is the light transport matrix. While the form above may be used in many contexts, for concreteness we may consider relighting of diffuse geometry, where \mathbf{B} is a vector of vertex intensities, and each column of \mathbf{T} corresponds to the intensity of each vertex for a single lighting direction (each row of \mathbf{T} corresponds to the contribution from different incident lighting directions at a single vertex).

The extension to glossy materials using the in-out BRDF factorization [LSS04, WTL04] is straightforward, simply being a product of view-independent transport factors, weighted by view-dependent BRDF terms. In these methods, the BRDF ρ is factored into n terms as

$$\rho(\omega, \omega_o) = \sum_{i=1}^n f_i(\omega) g_i(\omega_o), \quad (2)$$

where ω is the incident direction, and ω_o is the outgoing direction. Then, instead of a single transport matrix $\mathbf{T}(\mathbf{x}, \omega)$, we have n transport matrices $\mathbf{T}_i(\mathbf{x}, \omega)$, each of which includes the view-independent incident BRDF factor $f_i(\omega)$. Our algorithm is run on each of the \mathbf{T}_i separately, and they can be

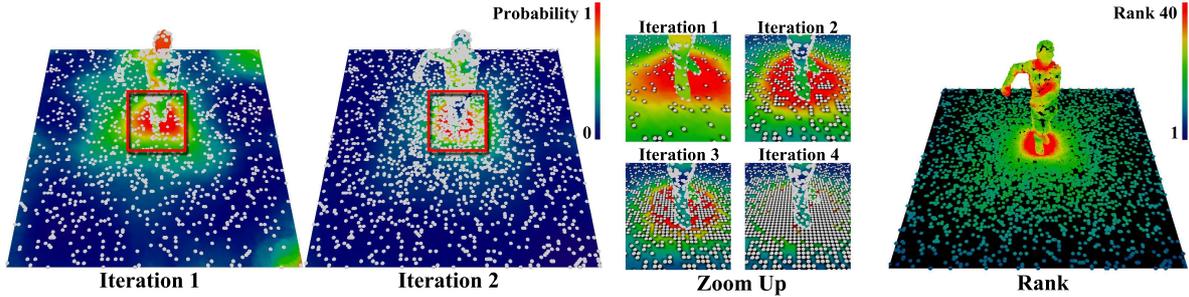


Figure 2: Spatial sampling via exploration. Several iterations are used to determine important regions in the scene. Uniform sampling is used for the first iteration (left), and probabilities are calculated (via local rank) for where to place dense vertices in the next iteration. The colors in the left two diagrams correspond to these probabilities after that iteration. The closeups show additional iterations that concentrate samples in the complex light transport (high rank) region near the feet. The local rank for each vertex is shown rightmost. It is lowest in near-convex regions, where light transport is simplest, and highest in complex shadowed regions such as the neck, beneath the feet, and waist.

combined at run-time with the view-dependent factors,

$$\mathbf{B}(\mathbf{x}, \omega_o) = \sum_i g_i(\omega_o) \sum_j \mathbf{T}_i(\mathbf{x}, \omega_j) \mathbf{L}(\omega_j), \quad (3)$$

where the inner summation is the standard matrix-vector multiplication of transport and illumination for relighting, and the outer summation considers the view-dependent BRDF factors. We may thus apply our method to each view-independent transport term \mathbf{T}_i separately, and so we simply consider the form of equation 1 in the rest of our exposition. Examples with glossy materials are in Figs. 17, 18, 19, 21.

For typical all-frequency relighting applications, \mathbf{L} can involve a $6 \times 32 \times 32$ cubemap, while \mathbf{B} may be evaluated at 10^4 – 10^5 vertices, so that the precomputed light transport matrix \mathbf{T} is of dimension approximately $10^5 \times 10^4$. While hardware rasterization is possible for direct lighting, global illumination requires explicit precomputation of each element of \mathbf{T} (or equivalently each image pixel or vertex for a given lighting direction). Our goal is to greatly speed up precomputation of the \mathbf{T} matrix, by sparsely sampling the spatial locations and angular directions.

Sparse sampling of the PRT matrix requires us to carefully choose certain rows (vertices) and columns (angular directions) to evaluate \mathbf{T} . We first define two kinds of scene vertices: *dense* and *sparse*. The light transport of dense vertices is sampled densely in their angular directions. The light transport of sparse vertices is interpolated from that of dense vertices, and they require very sparse sampling in the angular directions:

Phase 1—Sampling for dense vertices: Dense vertices are sampled in an iterative and adaptive way. At each iteration we decide where to put more dense vertices via local rank estimation, so that the local light transport information is sufficient for phase 2. For each dense vertex, we sample the angular domain in many (a “dense set” of) directions, but not exhaustively—we start with a sparse angular sampling, and use local variation to decide where to add more samples.

Phase 2—Interpolation for sparse vertices: Sparse vertices are interpolated from neighboring dense vertices. They

require very little (sparse) angular sampling. We select important angular directions or “features” to constrain the reconstruction, based on neighboring dense vertices. How to pick important directions and choose neighboring dense vertices will be discussed in later sections.

These operations correspond to a fine-grained row-column sampling strategy, in that we choose rows (vertices) carefully, and then columns (angular directions) adaptively for each row. Unlike previous work [HPB07, WDT*09] we have the freedom (and requirement) to sample each matrix element separately. Thus, ours is not a strict row-column method. In particular, even dense rows (vertices) do not require exhaustive evaluation across all columns (angular directions). Moreover, the same columns (directions) are not required to be used for different sparse rows (vertices). We also show how to integrate our method with CPCA.

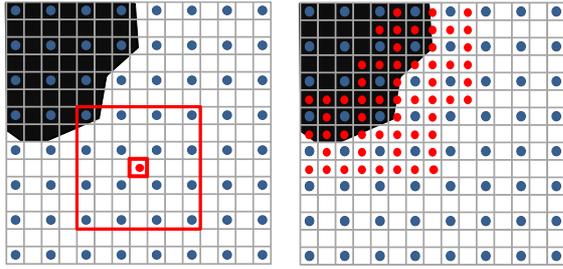
4. Sampling Dense Vertices

There are two problems for sampling dense vertices: where to sample vertices in the spatial domain, and how to reduce the cost of sampling them (in the angular domain). In terms of the \mathbf{T} matrix, this corresponds to which rows to sample densely, and how to reduce the number of columns to sample for each dense row.

The first problem is mainly for phase 2 interpolation, and the general idea is to choose more vertices for complex light transport regions; this requires us to determine the rank of that region. The second problem is simpler: since we already know certain directions are completely occluded or empty, it is wasteful to put samples there; this results in accelerated sampling for dense vertices.

4.1. Adaptively Sampling the Spatial Domain

We gather information about the scene by exploration in several iterations, as seen in Fig. 2. The initial iteration assumes an isotropic importance in vertices (spatial domain), and we randomly choose a small number of dense initial vertices (typically 5%). After each iteration, updates assign



• 1st pass regular grid samples • 2nd pass feature samples
Figure 3: Two pass angular sampling for dense vertices. **Left:** Samples (in blue) are taken at 1/4 of the resolution. For each unsampled direction, a window is used to check the variances. **Right:** If variance is nonzero, then new samples (in red) are drawn in the second pass. Samples will accumulate near high frequency edges, such as the boundary between blocked (black) and visible (white) regions. For the remaining unsampled directions, the value is directly obtained from their neighbors without explicit sampling.

new weights to other unsampled vertices, and we perform weighted random sampling to choose new vertices.

The weights for vertices are their rank, in terms of the local light transport matrix. As shown in Mahajan et al. [MKSRB07] and exploited in CPCA methods, while light transport is globally complex, it is of low rank in a local area. The density of sampling (and hence vertex weight) in each spatial region depends on this rank.

For each vertex, a local light transport matrix is built using radius-restricted k nearest-neighbors [AMN*98] from nearby vertices that have already been sampled densely. The rank of this matrix is used to determine the complexity of local light transport. The intuition is that for regions close to objects, their shadows and hence light transport tends to vary significantly, and is thus of high weight. Conversely, as seen in Fig. 2, the local rank will be low for regions far away from objects, where the cast shadows correspond to a small solid angle, and light transport otherwise varies smoothly.

4.2. Adaptively Sampling the Angular Domain

For each dense vertex, we can exhaustively sample angular directions, but some saving is obtained by adaptive angular sampling (Fig. 3)—our approach also bears some similarities to adaptive image sampling methods [Guo98].

As shown in Fig. 3(left), we first use a regular grid (on a cubemap), with 1/2 of the resolution in each direction (1/4 the total number of samples). For every unsampled direction, we then use a window to inspect the variance from sampled directions in that window (we use 7×7 windows as in Fig. 3(left)). This variance guides us in adding more samples, similar to adaptive pixel rendering methods (Fig. 3(right)). While we could use multiple iterations as in spatial sampling, we found a single iteration adequate.

Given some angular samples, the remaining directions need to be interpolated. We tried a number of radial basis

function scattered data interpolation schemes, but the reconstruction cost did not justify the savings. Instead, we apply simple heuristics, given the nature of the transport function. Light transport is a product of the BRDF and visibility for direct lighting. Since we assume a diffuse surface, the visibility term has the most significant variation. However, visibility is a binary function, so we can determine the value of a pixel by borrowing if all its neighbors are the same (unless there is an extremely high-frequency feature, which is a rare case (moreover, these high-frequency errors do not significantly affect the final image, since they will be integrated out against the lighting anyway.) In summary, our angular sampling algorithm for direct lighting is:

1. Sample the angular domain with a regular grid pattern, sending 1/4 samples. For every unsampled direction, calculate the variance from sampled directions within a small window.
2. If the variance for the unsampled direction is non zero, we mark it as a feature and sample it directly. Otherwise use the value from its neighbors.

The (diffuse) global illumination is generally much smoother and low-frequency, so sparsely sampling it is even simpler. We directly use the low-resolution grid, or 1/4 the total number of samples.

5. Interpolation for Sparse Vertices

Since light transport is locally low rank [SHHS03, MKSRB07], we can use dense vertices in a local neighborhood (i.e. a few rows) to interpolate and reconstruct the transport for the remaining vertices. This interpolation requires some examples (from local neighboring dense vertices) and constraints (newly sampled angular directions or columns at sparse vertices).

5.1. Reconstruction

More formally, each sparse vertex is considered a linear combination of its nearby dense vertices. In the following, let $\mathbf{T}(\mathbf{x})$ correspond to the transport row we seek to reconstruct,

$$\mathbf{T}(\mathbf{x}) \approx \sum_{k=1}^n \alpha_k \mathbf{T}(\mathbf{x}_k), \quad (4)$$

where \mathbf{x}_k are the local neighboring dense vertices, $\mathbf{T}(\mathbf{x}_k)$ are the corresponding dense transport functions, and α_k are unknown weights. To determine α_k , we need constraints, and therefore sample $\mathbf{T}(\mathbf{x})$ at some sparse set of directions (columns) $\mathbf{T}(\mathbf{x}, \omega_j)$,

$$\forall j: \mathbf{T}(\mathbf{x}, \omega_j) = \sum_{k=1}^n \alpha_k \mathbf{T}(\mathbf{x}_k, \omega_j). \quad (5)$$

Equation 5 is a system of linear equations for unknown α_k with constraints $\mathbf{T}(\mathbf{x}, \omega_j)$. Typically, the number of dense vertices n is small (usually 10–40), so it can be solved efficiently by least squares. Once the α_k are known, equation 4 is used to reconstruct the sparse vertex transport $\mathbf{T}(\mathbf{x})$.

While the least squares method solves equation 5 well, the solutions are sometimes not appropriate for reconstructing the transport. Since we only use a small number of angular samples, the least squares solution tends to over-fit the constraints, causing large positive/negative weights, with mostly non-zero values. Following [MKS RB07], if the local light transport is of low dimensionality, then a few samples should be sufficient to form the bases and to describe the interpolation, leading to a sparse weighting vector α .

Inspired by the compressive sensing literature, we have observed that better results can actually be obtained using sparse L_1 minimization (rather than least squares), using the fast algorithm in Kim et al. [KKL*07]. L_1 reconstruction preserves the sparse structure of light transport better, and leads to lower errors (results are shown later in Fig. 6).

In order for the reconstruction to be good with a sparse set of directions ω_j , it is crucial to sample useful angular directions, and also choose appropriate local neighbors \mathbf{x}_k .

5.2. Choosing Angular Direction Samples (Columns)

Choosing sparse angular direction samples is analogous to determining which columns to pick in row-column approaches. They need to be meaningful, and make the linear system in equation 5 well-posed. We describe a heuristic that considers two properties. First, what the variance among the dense vertices is in that angular direction (directions where all the dense vertices agree do not provide meaningful constraints). Second, for a given angular direction, in how many of the dense vertices it was a feature (the red dots in Fig. 3). We then choose the best angular directions, using a weight based on the product of these two metrics.

This method is illustrated in Fig. 4 for a single sparse vertex. Figure 4(a) shows the cubemaps for light transport at representative dense vertices. The angular features shown in Fig. 4(b) lie close to boundaries between blocked and visible regions. The bottom row shows the two properties we focus on. Figure 4(d) shows the variance for each angular direction among dense vertices, while Fig. 4(e) shows the union of features from (b). Our final selection considers a combination of these two aspects, and picks the most important angular directions in (f). Finally, Fig. 4(c) shows the high-fidelity reconstruction of light transport that we achieve.

Comparison: We briefly compare our angular sampling method to previous techniques for choosing the best columns. Resulting images, using reconstruction with the same number of columns, are shown in Fig. 5, and clearly indicate the benefit of our approach. Our method makes fundamentally different assumptions, and benefits from the fact that different columns (angular directions) can be selected for each row, to focus on local angular features.

The matrix row-column method [HPB07] uses weighted k-clustering to choose feature columns on a reduced row set (sampled from the entire scene). However, their method does not apply well to local regions, especially since there are many zero or low-energy columns in a local area, that are an issue for their error metric. Both [WDT*09] (that uses

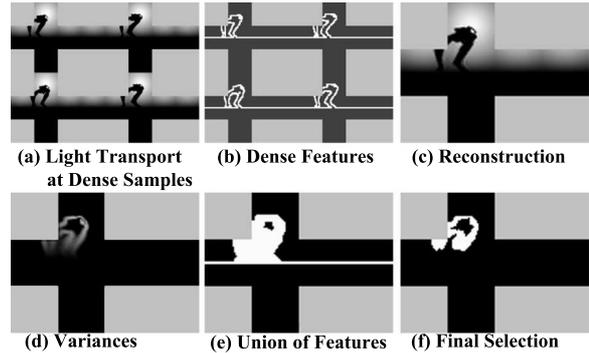


Figure 4: Picking angular features (columns) to reconstruct light transport for a sparse vertex v . (a) Light transport for representative neighboring dense vertices around v . (b) The angular features for each dense vertex, which lie close to boundaries between visible and blocked. (c) Final reconstruction result, using our method for picking angular features. The bottom row shows the weight given to each angular direction: (d) The variance for a particular direction among dense vertices. (e) The union of features from neighboring dense vertices. (f) The chosen angular directions.

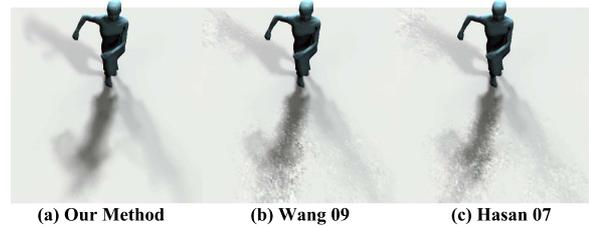


Figure 5: Comparison of our method (with the same number of angular features or columns) to Hasan et al. 2007 and Wang et al. 2009. All three methods use 25% of the vertices (rows) and 6% of the angular directions (columns).

k-means) and [HPB07] must choose the same columns for all sparse vertices in a region, and cannot easily adapt to local angular features. Performance is also important for us to minimize overhead, and clustering can be expensive, especially if needed for each group of sparse vertices.

5.3. Choosing Local Neighbors

The local dense neighbors chosen at a sparse vertex affect the angular features, and hence accuracy of the reconstruction. We follow a very simple heuristic, wherein dense vertices within some spatial radius of the sparse vertex are used. In general, if the radius defining local neighbors is too large, then we could include too many feature directions that are not important at the sparse vertex under consideration. A radius that is too small could result in an insufficient number of feature directions and an ill-posed system.

Figure 6 shows the effects of increasing the radius (this is for the ground plane in Fig. 1 where the total extent is 4 units, from -2 to $+2$). The error initially decreases as expected, but then increases. As we increase the radius (and hence also

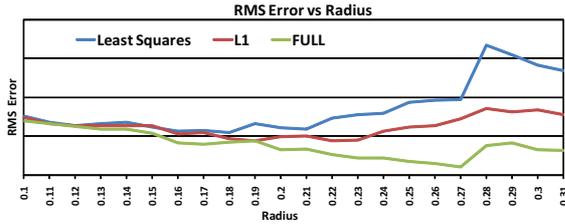


Figure 6: As expected, the error first decreases, and then increases with the radius used for choosing local dense neighbors. Sparse L_1 minimization performs better than least squares. The green “Full” curve is a control, with dense angular sampling. Note that for many radii, the error of the L_1 reconstruction with a sparse set of angular directions (red) is very close to using all the directions (green).

the number of bases), we have more features to select from—in the limit every angular direction is a feature, and we have no good way to choose the best directions or columns. In addition, the ratio between the number of constraints and the number of bases, or coefficients α_k (unknowns), also gets smaller, so the problem changes from a well-posed system to an ill-conditioned or ill-posed system.

As Fig. 6 shows, if we use all angular samples as constraints (the green “Full” line), we obtain the best reconstruction. This is stable and the error decreases consistently with increasing radius/number of bases as expected. If we use a smaller set of angular samples with least squares, the error decreases initially as we add more neighbors. However, at an inflection point, the system starts to over-fit the constraints and error increases. Finally, the L_1 solver maintains the sparsity of the solution, so it achieve a much lower error, often comparable to the “Full” line.

6. Compression and Clustered PCA

After precomputation, PRT algorithms compress the light transport matrix for relighting. Our method could be used to efficiently reconstruct light transport, followed by any compression scheme. However, even greater benefits can be obtained by more tightly integrating the sparse precomputation with compression, and we describe one approach to directly compute a CPCA representation. Note that while Sloan et al. [SHHS03] targeted low-frequency spherical harmonics, we use CPCA for all-frequency relighting from high-resolution cubemaps, as in Mahajan et al. [MKSRB07].

The original paper [SHHS03] proposed incrementally adding bases, besides the LBG iterations for clustering. Computing the projected distance to each cluster scales linearly with the number of samples, and thus is the bottleneck of the system. Since the light transport for sparse vertices is a linear combination of that for their local dense neighbors, it is a great saving to simply compute CPCA on dense vertices, and then utilize that information to assign sparse vertices to clusters. Our modification to CPCA has two passes: the dense pass on dense vertices, and the re-clustering pass on all vertices (see Fig. 7).

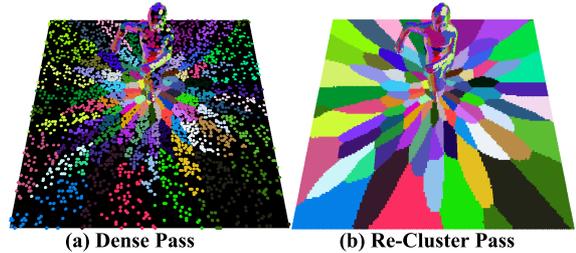


Figure 7: Integrating CPCA with sparse precomputation. (a): The dense pass does standard CPCA on dense vertices (b): The re-clustering pass assigns sparse vertices to the nearest cluster and then reclusters. Final cluster assignment is smooth, and the desired CPCA representation is directly and efficiently computed.

The dense pass works after the sampling for dense vertices. Standard CPCA is performed on the dense vertices only, by incrementally adding bases to avoid local minima, as described in Sloan et al. [SHHS03]. Since dense vertices are only about 25% of the original data set, computation times are significantly reduced (Fig. 11).

Once we obtain dense clustering information, sparse vertices are simply assigned their nearest neighboring cluster. The final re-clustering pass is done after all sparse vertices are assigned. This pass does not increment bases but only performs LBG iterations.

Note the premise of CPCA is that vertices in a local cluster have transport that is a linear combinations of a few bases. In fact, this is the same premise made for sparse vertices, that their transport is a linear combination of that for dense neighbors. Therefore, performing CPCA for dense vertices in the standard way, followed by a final re-clustering step, is adequate and in effect directly computes the compressed representation.

7. Results

While the methods in this paper are simple, and can be integrated into a variety of PRT systems, we focus most of the discussion in this section on relighting diffuse geometry. As noted in the overview, the extension to glossy materials is simple, and we also show a number of examples of view-dependent reflectance using four terms of the BRDF in-out factorization. Our baseline method precomputes transport of both direct and indirect lighting, using a raytracer for the former and a path tracer for the latter. Each element of the $\mathbf{T}(\mathbf{x}, \omega)$ matrix needs to be separately evaluated in software. We extend the precomputation phase with the algorithms described earlier, where only some elements of \mathbf{T} are sampled, and the full matrix is then reconstructed using our technique.

Scenes and Parameters: For experiments with diffuse geometry, we considered three scenes: the dancer in Fig. 1, the horse in Fig. 15 and bunny in Fig. 16. We will report later in the section on results for glossy scenes, shown in Figs. 17, 18 and 19. All scenes have a model with 10k–35k vertices on a ground plane with 30k vertices, and produce all-frequency shadow effects in contact regions and the ground plane. In

Scene	Mesh	Num. Vert.	Dense Vert.	Radius	Num. Feat.	Num. Clusters	Num Basis
Dancer	Model	9,971	2500	0.2	350	50	24
	Ground	29,241	6000	0.2	350	100	24
Horse	Model	8,431	2100	0.2	350	50	24
	Ground	29,241	6000	0.2	350	120	24
Bunny	Model	35,103	8850	0.2	350	70	24
	Ground	29,241	6000	0.2	350	120	24

Figure 8: Scenes used in the paper for diffuse geometry re-lighting, and algorithm parameters.

Scene	Mesh	Dense Vertices			Sparse Vertices			Total
		Spatial Sampling Rate	Angular Sampling Rate	Total Sampling Rate	Spatial Sampling Rate	Angular Sampling Rate	Total Sampling Rate	
Dancer	Model	25%	31.06%	7.76%	75%	5.7%	4.275%	12.04%
	Ground	20.52%	30.78%	6.32%	79.48%	5.7%	4.53%	10.85%
Horse	Model	24.9%	31.26%	7.78%	75.1%	5.7%	4.28%	12.06%
	Ground	20.52%	31.32%	6.42%	79.48%	5.7%	4.53%	10.95%
Bunny	Model	25.2%	29.87%	7.53%	74.8%	5.7%	4.26%	11.79%
	Ground	20.52%	29.31%	6.01%	79.48%	5.7%	4.53%	10.54%

Figure 9: The number of sparse samples used. Approximately 25% of the vertices are chosen as dense, but only a third of their angular directions are sampled. Only about 6% of the directions for the remaining sparse vertices are sampled, for a net sampling rate of only 10–12%.

Scene	Mesh	Sparsely Sampling				Full Sampling		Comparison	
		Dense Time	Sparse Time	Dense+Sparse	Total Time	Sampling	Total Time	Time Ratio	L2 Error
Dancer	Model	16m 32s	8m	24m 32s	1h15m 18s	2h13m 35s	5h40m 19s	20.95%	7.43%
	Ground	22m 40s	28m 6s	50m 46s		3h26m 44s			1.72%
Horse	Model	10m 29s	6m 26s	16m 55s	54m 9s	1h46m 7s	4h2m 22s	22.35%	10.2%
	Ground	15m 55s	21m 18s	37m 14s		2h16m 15s			1.61%
Bunny	Model	1h32m 18s	56m	2h28m 18s	3h25m 55s	9h52m 47s	13h7m 57s	26.14%	2.86%
	Ground	26m 31s	31m 6s	57m 37s		3h15m 10s			0.79%

Figure 10: Timings for various aspects of the precomputation. Our method requires only 20–25% of the cost of dense sampling.

addition, we consider diffuse shading and global illumination. All timings were run on a Dell T7400 workstation with a 2.5GHz processor and using a single thread. Since the focus of this paper is on precomputation, we do not directly report on relighting performance, that is comparable with that for fully sampled precomputations.

There are a few parameters for our algorithm, reported in Fig. 8. In particular, approximately 25% of vertices are used as “dense” vertices. Neighbors for sparse vertex reconstruction are chosen within a range of 0.2 units (the extent of the ground plane is from -2 to $+2$ units). In all cases, we use 350 angular directions or features for constraining sparse vertices, which is about 6% of the full set of 6144 directions. The number of clusters for CPCA is chosen in the range of 50–120, with 24 basis functions in each cluster.

Scene	Model	Our Method		Standard CPCA		Speed Up
		Time	Total	Time	Total	
Dancer	Model	32s	3m37s	6m47s	45m31s	12.6x
	Ground	3m5s		38m44s		
Horse	Model	27s	4m6s	5m40s	51m48s	12.6x
	Ground	3m39s		46m8s		
Bunny	Model	2m37s	6m20s	38m3s	1h20m57s	12.8x
	Ground	3m43s		42m54s		

Figure 11: Time for our method compared with standard CPCA. For all scenes, our method enables a 12 \times speed up.

Sampling Rate: Figure 9 lists the number of total samples of the transport matrix we use for reconstruction. Our results require an average of only 11% of the transport matrix, leading to an order of magnitude savings for precomputation. We choose 25% of vertices as “dense” vertices, but need only about a third of the angular samples even in those cases. For the remaining 75% sparse vertices, only about 6% of the angular samples suffice.

Precomputation Time: Ultimately, we care about how much the wall clock precomputation time has been reduced. Timings are shown in Fig. 10, and compared to brute force reconstruction of the transport matrix. Our method requires only 20–25% of the running time. The above results do not include the cost of CPCA compression. Timings are shown in Fig. 11 and indicate that integrating our sparse sampling approach with CPCA, as in Sec. 6, results in a substantial speedup of 12 \times on all scenes.

Details of time spent in various stages are given in Fig. 12. In general, ray or path tracing dominates the cost with about 90% of the total time. The overhead for reconstruction is only about 10% and mostly involves the L_1 solver for sparse vertex interpolation and the rank estimation. Note that although we compute only 10–11% of total samples, it takes up to 20% of the time for a full brute force precomputation. This result indicates that while we require many fewer rays or paths, our samples focus on the difficult regions where they are somewhat more expensive—a similar observation is made for ray tracing environment maps by Ben-Artzi et al. [BARA06]. However, the additional cost is manageable, and our method yields a precomputation speedup of 4 \times –5 \times .

Dense Vertex Distributions: Figure 13 shows the distribution of dense vertices, and ranks, on the horse and bunny scenes (data for the dancer have already been presented in Fig. 2). As can be seen, many more dense vertices are used in difficult regions, such as the contact between figure and ground. As expected, fewer dense vertices are needed in smoother areas like the ground plane, especially the near-convex parts far from the occluding model.

Quality of Results: We evaluate the error quantitatively in the rightmost column of Fig. 10. In all three examples, the ground plane has negligible error. However, there can be significant RMS L_2 errors in small parts of the more complex models, as shown in Fig. 14. We believe this is largely unrelated to our method. The bunny model is finely tessellated from a range scan, so that errors are evenly distributed. In

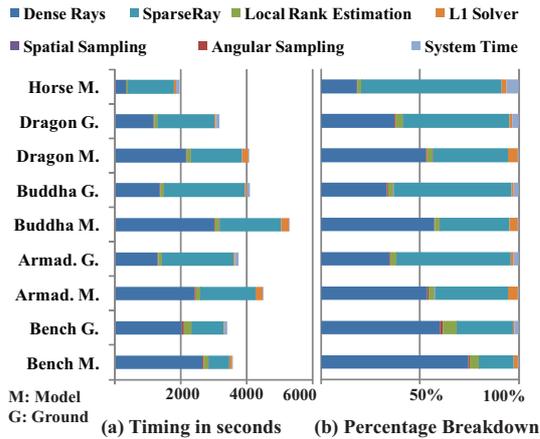


Figure 12: Detailed timing breakdown. (a) Each stage measured in wall clock time. (b) The percentage breakdown.

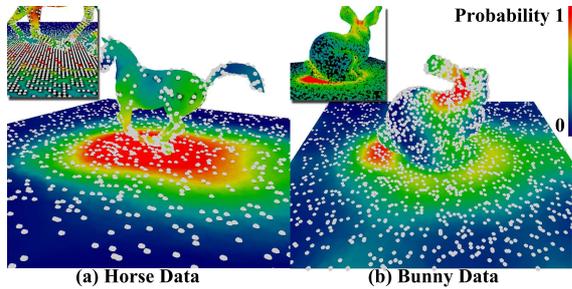


Figure 13: Distribution of dense vertices (white dots). Insets show a closeup of the contact region for the horse, and the rank image for the bunny (compare with Fig. 2 for the dancer). Colors indicate the probability for adding dense vertices, clearly showing more complex light transport near contacts and highly shadowed areas.

contrast, the horse and dancer are hand-designed, and the vertices are biased towards features of interest to the artist, at the cost of fewer vertices in some locations of complex light transport. In practice, these are very small regions, and the visual errors are not very noticeable. In the future, mesh refinement algorithms, building on [KPZ04], could be combined with our approach to add additional samples in those regions, based on our prediction of high local rank.

Our results in Figs. 1, 15 and 16 show little visual difference between relighting with sparsely sampled and fully sampled transport matrices, indicating we obtain a high quality approximation. The accompanying video also indicates there are no temporal artifacts.

Glossy Materials: While the results so far have focused on relighting diffuse geometry, the BRDF in-out factorization and equation 3 allow us to easily extend our method to view-dependent reflectance and glossy highlights. We use 4 terms in the BRDF factorization, showing examples in Figs. 17, 18 and 19. It can be seen that the images closely match the fully sampled versions.

Figure 20 shows timing numbers for the glossy scenes,

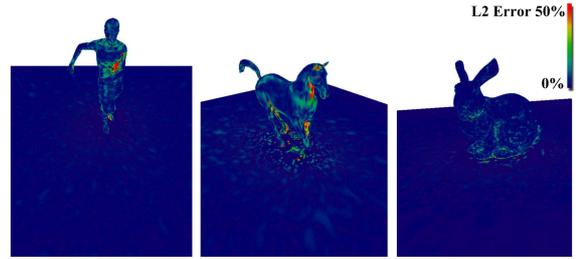


Figure 14: L_2 error comparison across different scenes.

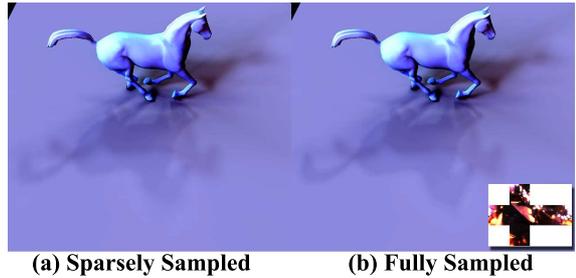


Figure 15: Comparison of sparsely and fully sampled transport for the horse. Our method reconstructs an accurate approximation.

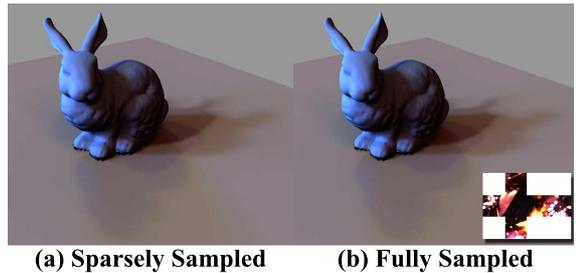


Figure 16: We compare sparse and full light transport reconstruction for the bunny scene. Our method has little visual difference with the ground truth, and accurately reproduces complex shadows.

analogous to our earlier results for diffuse scenes. Since we simply apply our algorithm to each view-independent transport term, the speed-ups are about $5\times$, and almost identical to those for relighting diffuse geometry.

More Complex Scene: Finally, Fig. 21 shows the bench scene from [OBARG06], which includes several objects, some of which have fairly complicated light transport. The street lamps also include glossy reflectance. As such, this example is a stress test for our method. As with any sparse reconstruction approach, slight blurring can result with our method, and some of the highest-frequency shadows, such as from the seams of the bench, are blurred over.

The light transport and shadows are more complex in this scene than our other examples. Hence, we do require somewhat more samples, as indicated in Fig. 20. Our method still produces very similar results to the fully sampled version, and results in a nearly $4\times$ speedup in precomputation time.

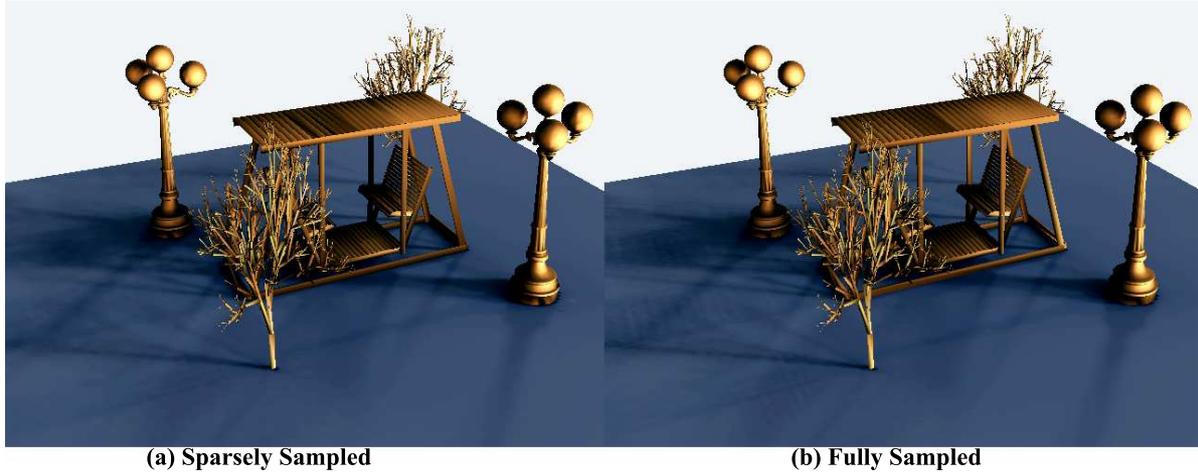


Figure 21: The bench scene includes several objects with difficult shadows and glossy reflectance. While we do slightly overblur some of the bench shadows, our method still produces an accurate result, comparable to the fully sampled version.

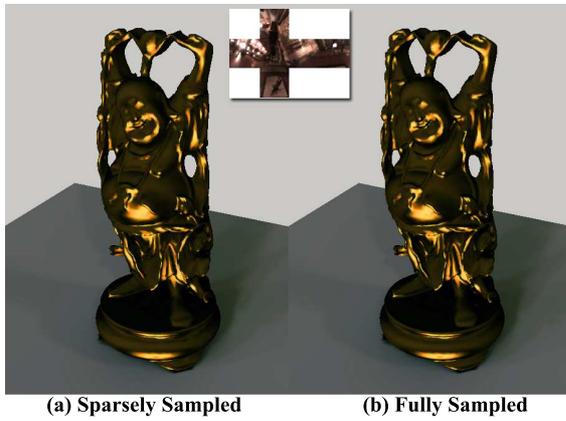


Figure 17: Buddha with Ashikhmin-Shirley Anisotropic BRDF (4 terms BRDF in-out factorization).

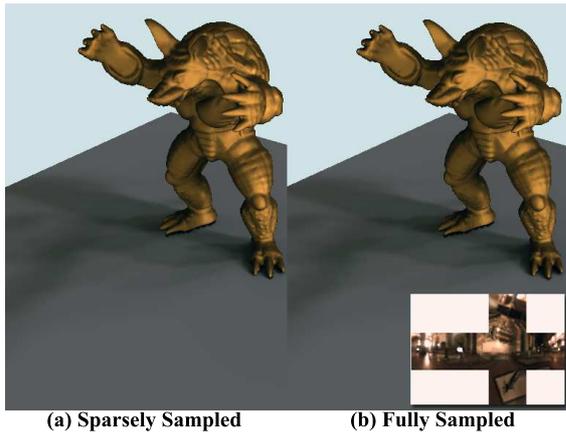


Figure 18: Armadillo with Ashikhmin-Shirley Anisotropic BRDF (4 terms BRDF in-out factorization).

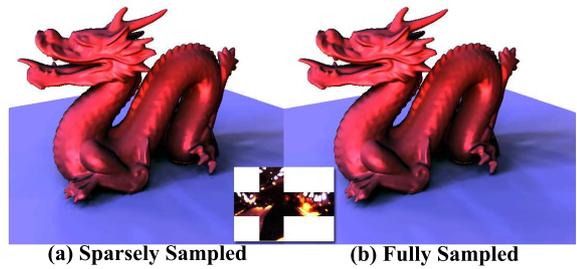


Figure 19: Dragon with Phong BRDF.

Scene	Mesh	Num. Vert.	Sampling Rate				Total	Timing		
			Dense Spatial	Dense Angular	Dense Total	Sparse Total		Sparsely	Fully	Speed Up
Armadillo	Model	25002	25.0%	31.1%	7.8%	4.3%	11.43%	2h	10h	4.75x
	Ground	29241	20.5%	31.0%	6.4%	4.5%		17m	7m	
Buddha	Model	24975	25.0%	29.7%	7.4%	4.3%	11.11%	2h	13h	5.12x
	Ground	29241	20.5%	29.6%	6.0%	4.5%		36m	22m	
Dragon	Model	25474	24.5%	30.3%	7.4%	4.3%	11.16%	2h	10h	5.22x
	Ground	29241	20.5%	29.8%	6.1%	4.5%		1m	31m	
Bench Scene	Model	19780	40.4%	39.9%	16.2%	3.4%	18.54%	1h	7h	3.67x
	Ground	29241	41.0%	35.4%	14.5%	3.3%		56m	7m	

Figure 20: Detailed results and timing for glossy objects, including the more complex bench scene.

8. Conclusions and Future Work

Most research in precomputation-based rendering has focused on adding real-time functionality. However, the precomputation itself is often the bottleneck, especially when considering global illumination. In this paper, we have taken a significant step towards addressing this drawback to PRT methods. We adaptively and sparsely precompute the light transport matrix, choosing well-placed “dense” vertices, and solving a simple linear system to reconstruct “sparse” vertices from a few angular samples. We also show how the method can be integrated with common compression schemes like clustered PCA. We demonstrate visually nearly identical results with an order of magnitude fewer samples, and a wall clock speedup in precomputation of $4\times-5\times$. This

has the potential to enable new approaches to rapid prototyping of scenes for lighting design or gaming environments.

In the future, we would like to leverage recent advances in GPU-based global illumination, to develop a hardware-accelerated precomputation pipeline. We envisage a system where the precomputation phase is interactive or takes only a few seconds, greatly reducing the barrier to PRT methods. We would also like to gain a deeper theoretical understanding into what samples of the transport matrix enable the best reconstructions, and how these vary for different types of scenes. Finally, estimation of variants of the transport matrix is also a challenge in offline many-light rendering and appearance acquisition, and we believe the insights in this paper hold promise for those domains as well.

Acknowledgements

This work was supported in part by NSF CAREER grant IIS-0924968 and ONR PECASE grant N00014-09-1-0741, as well as equipment and generous support from Intel, NVIDIA, Adobe, and Pixar.

References

- [AMN*98] ARYA S., MOUNT D., NETANYAHU N., SILVERMAN R., WU A.: An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM* 45, 6 (1998), 891–923.
- [BARA06] BEN-ARTZI A., RAMAMOORTHI R., AGRAWALA M.: Efficient shadows from sampled environment maps. *Journal of Graphics Tools* 11, 1 (2006), 13–36.
- [CRT06] CANDES E., ROMBERG J., TAO T.: Stable signal recovery from incomplete and inaccurate measurements. *Communications of Pure and Applied Mathematics* 59, 8 (2006), 1207–1223.
- [CT06] CANDES E., TAO T.: Near optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory* 52, 12 (2006), 5406–5425.
- [DAG95] DORSEY J., ARVO J., GREENBERG D.: Interactive design of complex time-dependent lighting. *IEEE Computer Graphics and Applications* 15, 2 (1995), 26–36.
- [DMM08] DRINEAS P., MAHONEY M., MUTHUKRISHNAN S.: Relative-error CUR matrix decompositions. *SIAM J. Matrix Anal. Appl.* 30, 2 (2008), 844–881.
- [Guo98] GUO B.: Progressive radiance evaluation using directional coherence maps. In *SIGGRAPH 98* (1998), pp. 255–266.
- [HPB06] HASAN M., PELLACINI F., BALA K.: Direct to indirect transfer for cinematic relighting. *ACM Transactions on Graphics (SIGGRAPH 06)* 25, 3 (2006), 1089–1097.
- [HPB07] HASAN M., PELLACINI F., BALA K.: Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics (Proc. SIGGRAPH 07)* 26, 3 (2007), Article 26.
- [KG09] KRIVANEK J., GAUTRON P.: *Practical Global Illumination with Irradiance Caching*. Morgan and Claypool, 2009.
- [KKL*07] KIM S., KOH K., LUSTIG M., BOYD S., GORINEVSKY D.: An interior-point method for large-scale L_1 regularized least squares. *IEEE Journal on Selected Topics in Signal Processing* 1, 4 (2007), 606–617.
- [KPZ04] KRIVANEK J., PATTANAIK S., ZARA J.: Adaptive mesh subdivision for precomputed radiance transfer. In *SCCG 04: Proceedings of the 20th spring conference on Computer graphics* (2004), pp. 106–111.
- [KTHS06] KONTKANEN J., TURQUIN E., HOLZSCHUCH N., SILLION F.: Wavelet radiance transport for real-time indirect lighting. In *EuroGraphics Symposium on Rendering 06* (2006), pp. 161–172.
- [LSSS04] LIU X., SLOAN P., SHUM H., SNYDER J.: All-frequency precomputed radiance transfer for glossy objects. In *EuroGraphics Symposium on Rendering 04* (2004), pp. 337–344.
- [LZT*08] LEHTINEN J., ZWICKER M., TURQUIN E., KONTKANEN J., DURAND F., SILLION F., AILA T.: A meshless hierarchical representation for light transport. *ACM Transactions on Graphics (Proc. SIGGRAPH 08)* 27, 3 (2008), Article 37, 1–9.
- [MKSRB07] MAHAJAN D., KEMELMACHER-SHLIZERMAN I., RAMAMOORTHI R., BELHUMEUR P.: A theory of locally low dimensional light transport. *ACM Transactions on Graphics (Proc. SIGGRAPH 07)* 26, 3 (2007), 62.
- [NRH03] NG R., RAMAMOORTHI R., HANRAHAN P.: All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics (Proc. SIGGRAPH 03)* 22, 3 (2003), 376–381.
- [NRH04] NG R., RAMAMOORTHI R., HANRAHAN P.: Triple product wavelet integrals for all-frequency relighting. *ACM Transactions on Graphics (Proc. SIGGRAPH 04)* 23, 3 (2004), 475–485.
- [NSD94] NIMEROFF J., SIMONCELLI E., DORSEY J.: Efficient re-rendering of naturally illuminated environments. In *EuroGraphics Workshop on Rendering 94* (1994), pp. 359–373.
- [OBARG06] OVERBECK R., BEN-ARTZI A., RAMAMOORTHI R., GRINSPUN E.: Exploiting temporal coherence for incremental all-frequency relighting. In *EuroGraphics Symposium on Rendering* (2006), pp. 151–160.
- [PML*09] PEERS P., MAHAJAN D., LAMOND B., GHOSH A., MATUSIK W., RAMAMOORTHI R., DEBEVEC P.: Compressive light transport sensing. *ACM Transactions on Graphics* 28, 1 (2009), Article 3, pages 1–18.
- [Ram09] RAMAMOORTHI R.: *Precomputation-Based Rendering*. NOW Publishers Inc, 2009.
- [SD09] SEN P., DARABI S.: Compressive Dual Photography. *Computer Graphics Forum (EUROGRAPHICS 09)* 28, 2 (2009), 609 – 618.
- [SHHS03] SLOAN P., HALL J., HART J., SNYDER J.: Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics (Proc. SIGGRAPH 03)* 22, 3 (2003), 382–391.
- [SKS02] SLOAN P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics (SIGGRAPH 02)* 21, 3 (2002), 527–536.
- [WDT*09] WANG J., DONG Y., TONG X., LIN Z., GUO B.: Kernel nystrom method for light transport. *ACM Transactions on Graphics (Proc. SIGGRAPH 09)* 28, 3 (2009).
- [WTL04] WANG R., TRAN J., LUEBKE D.: All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *EuroGraphics Symposium on Rendering* (2004), pp. 345–354.