N° d'ordre : 4449 ANNÉE 2011

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : (Informatique)

Ecole doctorale (Matisse)

présentée par

Václav Gassenbauer

préparée à l'unité de recherche UMR6074 IRISA
Institut de Recherche en Informatique et Systèmes Alèatoires
IFSIC

Illumination Coherence for Ligh Transport Simulation

Thèse soutenue à Rennes
le 12 Décembre 2011
devant le jury composé de :

Mathias PAULIN
Professeur, Univ. Paul Sabatier / rapporteur

Nicolas HOLZSCHUCH
Dir. de Recherche, INRIA Rhône-Alpes / rapporteur

Jiří BITTNER
Chercheur, ČVUT in Prague / rapporteur

Jiří ŽÁRA
Professeur, ČVUT in Prague / examinateur

Pavel SLAVÍK
Professeur, ČVUT in Prague / examinateur

Rémi COZOT
Maître de Conférences, Univ. Rennes 1 / examinateur

Jaroslav KŘIVÁNEK
Maître de Conférences, MFF UK / directeur de thèse

Kadi BOUATOUCH
Professeur, Univ. Rennes 1 / co-directeur de thèse

Abstract

Simulation of light transport in a scene is an essential task in realistic image synthesis. However, an accurate simulation of light as it bounces in the scene is time consuming. It has been shown that a key to speeding up light transport simulation algorithms is to take advantage of the high degree of spatial, angular, and temporal coherence. In this thesis we make three contributions in this area.

First, we propose *spatial directional radiance caching* (SDRC) for accelerating the light transport simulation in scenes with glossy surfaces. The SDRC algorithm takes advantage of the smoothness of shading on glossy surfaces by interpolating the indirect illumination from a set of sparsely distributed radiance samples that are both spatially and directionally close. We show that SDRC outperforms the original radiance caching proposed by Křivánek et al. [KGPB05] and also the Monte Carlo-based methods based on BRDF importance sampling.

In the next part of the thesis, we propose an efficient and accurate *local principal component analysis* (LPCA) algorithm for dimensionality reduction and data compression of large data sets. To achieve efficiency our new algorithm, called SortCluster-LPCA, passes varying information from previous iteration to the next, showing a speed up of up to 20. Improved accuracy is achieved through better initial seeding of cluster centroids in LPCA, producing substantially better data approximation.

Finally, we describe a work in progress focusing on the development of an algorithm for interactive relighting of animation sequences with indirect illumination. We formulate the relighting problem as a large 3D array expressing light propagation in a scene over multiple frames. We suggest an adaptive algorithm to make the pre-computation tractable exploiting coherence in light transport. Since our approach has not been implemented yet we leave its practical verification as a future work.

Keywords. Computer graphics, rendering, coherence, global illumination, light transport simulation, radiance caching, principal component analysis, PCA, k-means, precomputed radiance transfer, cinematic relighting.

Contents

List of Figures

Introduction

1

Contents

Let us imagine we are observing some objects in the real world and let us have an accurate virtual model of these objects represented in a computer. One of the important problems in the field of computer graphics is to generate an image of the virtual objects so that the difference between a photograph of the reality and the computer-generated image is not noticeable. Generating such photorealistic images is called *Realistic Image Synthesis* and has a number of applications. One of the main applications of realistic image synthesis is the film industry. More and more movies contain miscellaneous visual effects which are impossible or very expensive to produce in a traditional way. Photorealism is required to provide the illusion of watching the real world. In architecture one can design the interior or exterior of buildings and create photorealistic images of the designs before the objects actually exist. Realistic image synthesis is also useful for lighting design: Lights can be simulated in a virtual scene and then the final lighting configuration can be used in the real world. We could continue giving examples of other applications of realistic image synthesis. Generally realistic image synthesis is used everywhere, where we need to provide the user with the illusion of reality.

1.1 Global Illumination Problem

The input of any realistic image synthesis system is a complete description of a virtual world. It consists of the objects' geometry, the material properties of those objects, and the geometry and emission characteristics of light sources. In order to create an image of the virtual world we need to calculate the amount

(a) Color bleeding

(b) Refraction

(c) Caustic

(d) Glossy reflections

Figure 1.1 – Global illumination effects. Images (a), (b), (c) were borrowed from [KG09], and (d) is our photograph of Walt Disney Concert Hall in Los Angeles.

of light entering a virtual camera which is positioned in the scene. Light enters the camera directly from the light sources but also indirectly as it is reflected or refracted off the objects in the scene. The effects of light as it bounces in the scene are referred to as *global illumination* (GI); this is the main objective of realistic image synthesis. Some examples of global illuminated scenes are given in Figure 1.1.

The fundamental mathematical foundation for realistic image synthesis is the rendering equation [Kaj86]. The rendering equation describes the energy balance in the scene. One approach to solve the rendering equation is the finite element method that the radiosity algorithms are based on [CW93, SP94]. The radiosity algorithms have been shown to be very efficient, especially for relatively simple scenes with diffuse surfaces. For complex environments with a lot of materials with different reflectance functions, however, radiosity algorithms are impractical in terms of memory space and computation time.

Other approaches to solve the rendering equation are based on Monte Carlo methods. Instead of solving the rendering equation directly they calculate it in an explicit form: they expand the equation into an infinite series of finite-dimensional integrals that can be evaluated numerically. The pioneering work in solving the finite-dimensional integrals using Monte Carlo was distributed ray tracing [CPC84] and path tracing [Kaj86], as well as the follow-up algorithms like bi-directional path tracing [LW93, VG97], etc. Such methods are general and robust. But it can take hours to generate artifact-free images.

The main source of inefficiency of above Monte Carlo algorithms is that they reuse no (or just very little) information about the contributions of previously calculated light paths, essentially ignoring any coherence of light transport in the scene. To speed up the global illumination calculation other algorithms trade generality of provided solution for restrictions imposed on the scene configuration and/or make use the coherence of light transfer in the scene [WRC88, KGPB05, GBP07, HPB07]. Irradiance and radiance caching algorithms [WRC88, KGPB05] are efficient methods for solving global illumination in scenes with diffuse and low-gloss surfaces, respectively, using the observation that indirect lighting changes slowly over surfaces. Another observation is that lighting changes smoothly whenever the camera and/or objects in the scene move slowly [GBP07, HPB07].

Mathematical foundation for light transport coherence was presented in [DHS+05, MSRB07, PML+09]. These works how that light transport has a high degree of coherence over directional, spatial, and temporal domains. As suggested in these works, the use of light transfer coherence can play the key role in further acceleration of global illumination algorithms. In this thesis we focus on the very goal of speeding-up global illumination algorithms by exploiting the coherence of light transfer in the scene.

1.2 Motivation

A huge amount of calculation must be performed to provide a high-quality image with global illumination. To reduce the amount of calculation it is customary

to simplify the problem in an application-specific manner. In this thesis we develop three algorithms of different applications making use of light transfer coherence for efficient calculation of GI: one for general computation of GI on glossy surfaces, another for the acceleration of *local principal component analysis* (LPCA) used for compression of large data matrices in *precomputed radiance transfer* (PRT) [SKS02, SHHS03] and image databases of *Bi-directional texture function* (BTF) [FH09], and the last one which is a work in progress extending PRT for animation sequences. In the following, we shortly introduce these three applications, before summarizing our contributions.

Efficient Algorithm for Global Illumination on Shiny Surfaces. As we mentioned above, computing full global illumination in virtual scenes is very time-consuming. Monte Carlo importance sampling [Coo86, LF97, LRR04], Metropolis light transport [VG97] or photon mapping [Jen01] are examples of very general techniques for solving GI. Irradiance caching [WRC88] delivers fast GI solution in scenes with diffuse surfaces. Radiance caching [KGPB05] includes the support for caching in the scenes with low-glossy surfaces. But effective algorithms for computing GI on shiny surfaces are missing. We focus on generalizing caching approaches to shiny surfaces. We propose a novel efficient algorithm that we call *spatial directional radiance caching* (SDRC), for computing GI effects on these surfaces. We use both spatial and angular coherence of light transport in the scene to make our algorithm efficient.

Improving performance and accuracy of Local PCA. Precomputed radiance transfer (PRT) [SKS02] and image databases of Bi-directional texture function (BTF) [MMK03, FH09] are the main applications where the local principal component analysis (LPCA) [KL97] is largely used. PRT refers to a group of methods used for interactive relighting of a virtual scene with GI effects while dynamically changing some parameters of the scene. The original PRT application was used for image relighting of a scene lit by an environment map. In the many follow-up papers [SHHS03, LSSS04, NRH03, FPJY07, HR10] the PRT has been improved, lifting some restrictions on the scene configuration. The PRT methods have been shown to be very popular for lighting design in cinematography [KAMJ05, HPB06]. They have also been used in computer games because of their ability to deliver GI effects at real-time frame rates. To achieve these goals, PRT techniques precompute and compress light transfer in the scene expressed as a large transfer matrix. Precomputation and compression of the light transfer matrix, however, is very time-consuming and is the serious bottlenecks of PRT methods. Huang and Ramamoorthi [HR10] address the slow precomputation of light transfer matrix by exploiting spatial and angular coherence of the light

transfer. Nevertheless they still use the slow LPCA for the light transfer matrix compression. Our goal is to accelerate the LPCA by exploiting the coherence in the compressed data.

Precomputed Radiance Transfer for Animation Sequences. Our last application also deals with PRT. Instead of relighting in static scenes we focus on relighting animation sequences with GI. Some existing works deal with relighting for articulated characters [NSK+07, FPJY07] but these approaches do not scale to the requirements needed for our application domain, the cinematic lighting design: they mostly deal with simple characters and deliver less accurate GI. On the other hand we aim at robust cinematic system for relighting in complex scenes while providing a high-quality rendering of the animation sequence. We believe that animation relighting would be extremely useful in computer cinematography. A designer will be able to design lighting in any frame of the animation sequence while having the possibility to play back the whole sequence with the updated lighting. Technically we build on the idea of direct-to-indirect light transfer [HPB06] but instead of precomputing light transfer for one frame we precompute it for the entire animation sequence. We develop an efficient algorithm to make the calculation efficient by leveraging directional, spatial, and temporal coherence of light transport.

1.3 Summary of Contributions

The contributions of this thesis are efficient methods for realistic image synthesis in three different applications. Our main contributions are the following:

Spatial Directional Radiance Caching (SDRC). We present a new approach for accelerated global illumination computation in scenes with glossy surfaces. Our algorithm combines BRDF importance sampling with the sparse illumination computation used in the (ir)radiance caching algorithm proposed by [WRC88, KGPB05]. To make this approach feasible, we extend the idea of lazy illumination evaluation used by these approaches—query the cache, perform interpolation if possible, otherwise compute a new illumination value and store it in the cache for later reuse—from the spatial to the directional domain. Using importance sampling allows us to apply caching not only on low-gloss but also on shiny materials with high-frequency BRDFs, for which the previous caching algorithms break down.

Improving Performance and Accuracy of Local PCA (LPCA). The LPCA [KL97] is one of the very popular and successful data compression tech-

niques used for compression of large data sets, like light transfer matrices in pre-computed radiance transfer [SKS02, SHHS03] or of Bi-directional texture functions [FH09]. The main disadvantage of LPCA is the slow computation with often somewhat inaccurate result, which is a bottleneck in practical applications. We propose a new algorithm that significantly improves the efficiency and the accuracy of the LPCA. To achieve the efficiency we take the advantage of the information gathered in one stage of the LPCA and pass it to the next stage. To improve the approximation accuracy, we propose a fast initialization algorithm whose result is passed as an initial input to the LPCA. We show that our initialization algorithm produces substantially more accurate data compression with a higher compression ratio.

Precomputed Radiance Transfer for Animation Sequences. Several relighting systems for lighting design in computer cinematography have been proposed, mostly based on precomputed radiance transfer [KAMJ05, HPB06, KTHS06, LZT+08]. They are restricted, however, for relighting of static scenes. We have been working on extending these systems to support for relighting of animated sequences where objects deformation and the camera path will be known in advance. We use a large transfer tensor (3D array) to describe the light transfer over multiple frames. We maximize the use of coherence contained in the light transfer to make calculation of the light transfer tensor feasible. We show how the local principal component analysis (LPCA) can be combined with non-linear wavelet approximation [Mal08] to speed-up compression while preserving good approximation accuracy.

1.4 Thesis Outline

The thesis is divided into five chapters. This chapter introduced the aim of realistic image synthesis, presented the motivation for our objective—exploiting coherence of light transport for accelerating image synthesis algorithms—and summarized our contributions.

Our contributions are concerned with different applications. Instead of giving a chapter for the background and the state-of-the-art, we decided to provide this information separately within each chapter describing the specific application. Chapter 2 presents *spatial directional radiance caching*, a novel algorithm for accelerated global illumination computation in scenes with shiny surfaces, provides some results of our implementation, and compares them to the results of previous methods. Chapter 3 describes local principal component analysis (LPCA) as a popular technique used for the compression of radiance transfer matrices in precomputed radiance transfer (PRT), proposes a novel efficient algorithm for

accelerating the LPCA, and also presents a fast algorithm initializing the LPCA. Chapter 4 describes a work in progress focusing on the development of an interactive algorithm for cinematic relighting of animated sequences with predefined camera motions. Chapter 5 concludes the thesis, giving some ideas for future work.

Spatial Directional Radiance Cache 2

Contents

In this chapter we describe a new approach for accelerated global illumination computation in scenes with glossy surfaces. We borrow the idea of lazy illumination evaluation used in the irradiance cache [WRC88] and extend it from the spatial to directional domain. We verify our algorithm on several testing scenes containing materials of a wide range of supported BRDF. The work described in this chapter was published in [GK08, GKB09].

2.1 Introduction

Global illumination (GI) effects constitute an important aspect of generating realistic images for applications spanning film production, video games, industrial design or architecture. Many algorithms for simulating GI effects have been

proposed, however, these methods are very time-consuming for general environments. Therefore, full global illumination solution is often restricted to a simpler case, such as ambient occlusion or predominantly diffuse indirect illumination, for which efficient algorithms are known [WRC88, WABG06, HPB07]. In this paper, we focus on a more difficult case of indirect illumination on surfaces with arbitrary material properties, coming from both diffuse and glossy objects. We refer to those effects as *directional indirect illumination*.

Monte Carlo (MC) ray tracing algorithms are almost exclusively used for rendering scenes with directional indirect effects. Unfortunately variance of MC estimators gives rise to image noise that decreases only with the square-root of the number of samples. Therefore, research in computer graphics has focused on variance reduction techniques, of which the most widely accepted is importance sampling [PH04, CAE08, CAM08]. Other acceleration techniques trade efficiency for bias in the resulting solution. Some of these techniques calculate illumination only at several locations in the scene exactly. The resulting image is then generated using a reconstruction function from these points. The pioneering work in this area was the irradiance caching algorithm [WRC88], the extensions of which include *radiance caching* on glossy surfaces with low-frequency BRDFs [KGPB05] and in participating media [JDZJ08].

In this paper, we propose an algorithm that combines the sparse computation of indirect illumination used in radiance caching with the variance reduction offered by BRDF importance sampling. The main idea is to extend the sparse illumination evaluation from the spatial to the directional domain: The indirect radiance at a point in a direction is evaluated by interpolating radiance samples from neighboring directions and locations. Using this strategy we obtain an algorithm that has the following advantages: (1) exploits spatial and directional illumination coherence, (2) ensures a smooth integration of the view-dependent BRDFs through interpolation in both these domains, (3) avoids conversion of the scene BRDFs into a special-purpose representation, such as spherical harmonics [KGPB05], thereby making the algorithm more flexible, (4) gains efficiency by exploiting BRDF importance sampling.

The proposed algorithm computes the first bounce of indirect illumination on glossy surfaces. Multiple bounces can be added e.g. by the use of photon mapping [Jen01]. The algorithm can be used as a part of a full solution to global illumination computation. Diffuse inter-reflections could be handled using irradiance caching, highly specular reflections by classical MC sampling methods, while the reflections on glossy surfaces could be computed using our algorithm.

The remainder of this chapter is organized as follows. Section 2.2 summarizes the prior work, Section 2.3 reviews the radiance caching algorithm, Section 2.4

describes our algorithm while Section 2.5 provides its evaluation. The algorithm is discussed in Section 2.6 and Section 2.7 concludes the chapter.

2.2 Related Work

Our algorithm is a variant of illumination caching techniques. The seminal work in this area is irradiance caching [WRC88], that accelerates global illumination computation on diffuse surfaces. Indirect illumination is computed by interpolating previous irradiance values if these are available. If none of the previously cached irradiance values can be used, a new one is computed by sampling the hemisphere and cached. Ward and Heckbert [WH92] propose to use the translation and rotation gradient to improve the quality of interpolation. Tabellion and Lamorlette [TL04] use irradiance caching combined with an approximate lighting model in cinematic lighting. Brouillat et al. [BGB08] propose to use a photon map [Jen01] for fast construction of a coarse approximation of irradiance cache. Arikan et al. [AFO05] speed up irradiance caching by decomposing illumination into near and distant terms. The above mentioned methods work for only diffuse illumination while radiance caching proposed by Křivánek et al. [KGPB05] is designed for the use on glossy surfaces with low-frequency BRDFs. The goal of our work is to modify radiance caching in such a way that caching can be applied for higher-frequency BRDFs.

Other extensions of the (ir)radiance caching algorithms include the support for caching in dynamic environments [GBP07]. Instead of building (ir)radiance cache for each frame from scratch, they reuse and update existing records for several frames. Gautron et al. [GKBP05] also propose speeding up irradiance caching using the graphic accelerator by reformulating the algorithm to better fit the GPU architecture. Jarosz et al. [JDZJ08] extend the radiance caching algorithm to cache lighting in participating media. Later they reformulated the gradient computation to accounts for changes of occlusion [JZJ08]. These extensions show the potential of the caching methods for faithfully rendering global illumination effects.

The idea of reusing partial results of illumination computation is not limited to the caching approaches. For example, photon mapping proposed by Jensen [Jen01] and instant radiosity proposed by Keller [Kel97] reuse the same set of paths initiated from light sources for all image pixels. Approaches for reusing camera-paths and bidirectional paths have also been proposed [BSH02, HDMS03, CSH08]. Other techniques [PBSP08, VP08, LW95] reuse the information gathered during illumination sampling to build an importance function to be used at other locations.

2.3 Background: Radiance Caching

Radiance caching [KGPB05] is based on the observation that indirect illumination tends to change slowly on glossy surfaces. Therefore, it can be computed only at some points, stored in a cache, and later reused for fast interpolation. This approach can significantly speed up the computation.

Whenever indirect illumination needs to be computed at a point $\mathbf{p}$, the cache is queried for nearby *records* (i.e. the cached illumination values) available for interpolation, formally defined by the set:

$$S(\mathbf{p}) = \left\{ i \mid w_i^s(\mathbf{p}) > \frac{1}{a} \right\}, \tag{2.1}$$

where a is a user defined maximum interpolation error. The greater the value of a, the more allowance for interpolation. The spatial weight $w_i^s(\mathbf{p})$ determines the contribution of the i-th record to illumination at a given point $\mathbf{p}$ with normal $\mathbf{n}$. It is given by:

$$w_i^s(\mathbf{p}) = \left(\frac{\| \mathbf{p} - \mathbf{p}_i \|}{R_i} + \sqrt{1 - \mathbf{n} \cdot \mathbf{n}_i} \right)^{-1}, \tag{2.2}$$

where R_i is the harmonic mean distance to the objects visible from the i-th record's location $\mathbf{p}_i$, and $\mathbf{n}_i$ is the normal at $\mathbf{p}_i$.

If the set $S(\mathbf{p})$ is empty, i.e. no records in the vicinity of $\mathbf{p}$ have been found, it is necessary to compute a new record. The full hemisphere above $\mathbf{p}$ is sampled and the directional incoming radiance is approximated by a vector of spherical harmonics coefficients Λ_i. A SH approximation of $\frac{\partial \Lambda_i}{\partial x}$ and $\frac{\partial \Lambda_i}{\partial y}$, the derivatives of Λ_i with respect to translation along the local x and y axes, is also computed. These vectors are stored as a new record in the radiance cache.

If $S(\mathbf{p})$ is not empty, the total outgoing radiance is computed by spatial interpolation of the contributions $L_i^{\text{out}}(\mathbf{p}, \omega^{\text{out}})$ due to the records in $S(\mathbf{p})$:

$$L^{\text{out}}(\mathbf{p}, \omega^{\text{out}}) = \frac{\sum_{i \in S(\mathbf{p})} w_i^s(\mathbf{p}) \cdot L_i^{\text{out}}(\mathbf{p}, \omega^{\text{out}})}{\sum_{i \in S(\mathbf{p})} w_i^s(\mathbf{p})}$$

The contribution of the i-th radiance record is given by:

$$L_i^{\text{out}}(\mathbf{p}, \omega^{\text{out}}) = \mathbf{R}_i \left(\Lambda_i + d_x \frac{\partial \Lambda_i}{\partial x} + d_y \frac{\partial \Lambda_i}{\partial y} \right) \cdot \mathbf{C}(\mathbf{p}, \omega^{\text{out}}),$$

where $\mathbf{R}_i$ is a rotation matrix used to align coordinate frames at $\mathbf{p}_i$ and $\mathbf{p}$, d_x and d_y is the displacement from $\mathbf{p}_i$ to $\mathbf{p}$ in the local coordinate frame of record i. Finally, $\mathbf{C}(\mathbf{p}, \omega^{\text{out}})$ is the vector of SH coefficient representing the BRDF lobe at $\mathbf{p}$ for the outgoing direction ω^{out}.

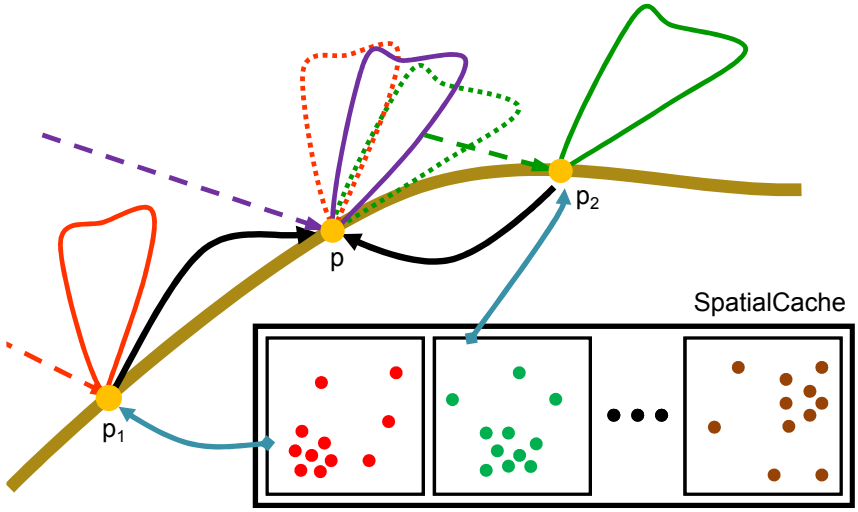


Figure 2.1 – Lazy evaluation scheme in the directional domain. Our goal is to compute the outgoing radiance at point $\mathbf{p}$. Suppose that there are two cache records at points $\mathbf{p}_1$ and $\mathbf{p}_2$ close to $\mathbf{p}$, that store the incoming radiance samples with high density around the BRDF peaks. The incoming radiance at $\mathbf{p}$ may be obtained by merging the information in the records at $\mathbf{p}_1$ and $\mathbf{p}_2$. If, after the merging, some parts of the hemisphere still do not have high enough sample density, additional rays are traced.

2.4 Spatial Directional Radiance Caching

2.4.1 Motivation

The major limitation of radiance caching consists in the *uniform* sampling of full hemisphere used to estimate incoming radiance. With uniform sampling, the rays traced outside the lobe of a glossy BRDF produce wasted effort. Nonetheless, the use of spherical harmonics in radiance caching necessitates a uniform sampling pattern.

To avoid unnecessary computation for shiny glossy surfaces, our algorithm employs BRDF importance sampling when a new cache record is created. Doing so, however, requires to keep track of the sample density on the hemisphere of the cached records, so that they can be reused at other locations, where the BRDF lobe may be slightly different. We approach this issue by extending the radiance caching's lazy evaluation scheme from the spatial to the directional domain, adding directional samples on the fly as needed. Figure 2.1 illustrates our lazy evaluation procedure. Using this approach, we are able to handle shiny materials for which the original radiance caching would not be efficient. As an added benefit, there is no need to convert the BRDFs into the SH basis.

Since we use non-uniform sampling to create cache records, we need a representation for the incoming radiance that offers directional localization. This is why spherical harmonics are not an option. Wavelets do offer localization but the rotation $\mathbf{R}_i$ is a limiting factor. Although it can be made relatively fast by pre-computing the rotation matrices [WNLH06], the memory requirements for a reasonable directional resolution (256×256 or more) render this approach impractical. Furthermore, adding new radiance samples to existing records is not simple. Our choice is, therefore, to retain the individual radiance samples and organize them in a kd-tree for fast access.

2.4.2 Overview

Our algorithm is based on the following caching scheme. To compute outgoing radiance at a point $\mathbf{p}$, we search for nearby records available for interpolation in the *spatial* cache. If no records are available we create a new cache record as follows. We generate random directions using BRDF importance sampling, compute incoming radiance for each direction by ray tracing, project these directions onto a 2D domain, $\mathcal{D}$, and build a kd-tree over the radiance samples. We call this kd-tree the L-tree. We store the entire L-tree in the spatial cache as a single record. Given the L-tree, we then continue the computation as though the spatial cache query succeeded.

If there are one or more L-trees available close to $\mathbf{p}$, we interpolate their contributions. We generate random directions using BRDF importance sampling as before. But instead of computing the incoming radiance for these directions by ray tracing, we try to find close radiance samples stored in the L-trees for each direction and possibly reuse it. If no suitable radiance sample is available for a direction, we shoot a ray to obtain a new radiance sample and update an existing L-tree. Finally, the outgoing radiance is computed as a weighted average of the radiance samples from the individual L-trees. The pseudo-code for our spatial directional radiance caching is given in Algorithm 1.

2.4.3 New Record Computation

To create a new spatial record, we first generate N random directions using BRDF importance sampling and compute incoming radiance for each direction by ray tracing. We then map these samples from the sphere to the domain $\mathcal{D}$ using paraboloid mapping [HS98] which has low distortion and fast analytical transform. We construct a kd-tree over the radiance samples mapped to $\mathcal{D}$. To keep the memory requirements low, we quantize sample position inside $\mathcal{D}$ to 2 bytes (resolution of 256×256) and use Ward's RGBE format to represent the

Algorithm 1 Spatial directional radiance caching

$S \leftarrow$ `LookUpSpatial(`$\mathbf{p}$, $\mathbf{n}$`)` ;
if *(S is empty)* **then**
  $[\omega_j^{\text{in}}]_{j=1}^{N} \leftarrow$ `SampleBRDF(`$\mathbf{p}$, ω^{out}, N`)`;
  foreach ω_j^{in} **do** compute $L^{\text{in}}(\mathbf{p}, \omega_j^{\text{in}})$ using ray tracing;
  L-tree $\leftarrow$ build kd-tree over $[\overline{\omega_j^{\text{in}}}]_{j=1}^{N}$;
  Store L-tree in spatial cache;
  $S \leftarrow \{L\text{-tree}\}$;
end
$[\omega_j^{\text{in}}]_{j=1}^{M} \leftarrow$ `SampleBRDF(`$\mathbf{p}$, ω^{out}, M`)`;
foreach ω_j^{in} **do**
  foreach i ***in*** S **do**
    $T_i \leftarrow$ `LookUpDirectional(`$\overline{\omega_j^{\text{in}}}$`)`;
  end
  if *($\cup T_i$ is empty)* **then**
    Choose $L\text{-tree}_u \in S$ for updating;
    if (no suitable L-tree exists) **continue**;
    $L^{\text{in}}(\mathbf{p}, \omega_j^{\text{in}}) \leftarrow$ `TraceRay(`$\mathbf{p}$, ω_j^{in}`)`;
    Insert $L^{\text{in}}(\mathbf{p}, \omega_j^{\text{in}})$ into $L\text{-tree}_u$;
  end
  $\tilde{L}^{\text{in}}(\mathbf{p}, \omega_j^{\text{in}}) \leftarrow$ `InterpolateRadiance(`$[T_i]_{i \in S}$`)`;
  Update $\tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}})$;
end

incoming radiance. Together with 2 bytes for bookkeeping information, each L-tree node takes 8 bytes. The record computation is completed by inserting the whole L-tree into the spatial cache organized as an octree [WRC88].

The number of rays N used for creating a new record is derived from the number of directions M used to compute the outgoing radiance during interpolation. If N is too low ($N < 8M$), the query to the contributing L-trees often fails and triggers many L-tree updates, negatively affecting performance. The number of L-tree updates stops decreasing for $N > 16M$. A good compromise that works well in our scenes is to use $N \approx 12M$.

2.4.4 Incoming Radiance Interpolation

To compute outgoing radiance at point $\mathbf{p}$ we start by querying the spatial cache. The definition of the set $S(\mathbf{p})$ of records used for interpolation and their spatial

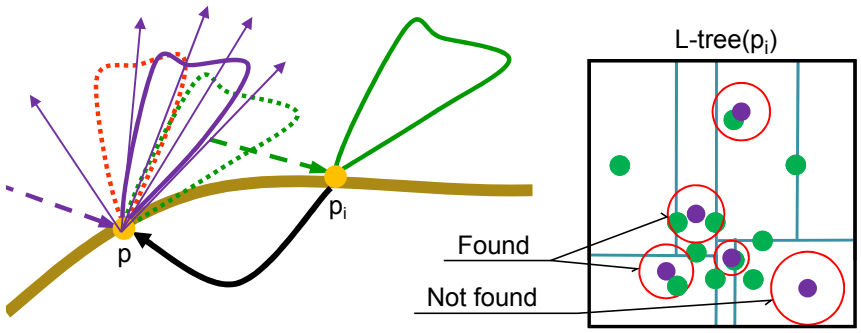


Figure 2.2 – Directional interpolation. We generate M directions ω_j^{in} by BRDF sampling at $\mathbf{p}$. The directions are mapped to points in $\mathcal{D}$. For each point $\overline{\omega_j^{\text{in}}}$ (blue dots) we find directionally close radiance samples (green dots) in the contributing L-trees. These radiance samples are used for interpolation. The directions for which no radiance samples are found yield ray tracing and L-tree update.

weights are borrowed from the original radiance caching algorithm, see Equations (2.1) and (2.2). If the set $S(\mathbf{p})$ is empty, we compute a new record as described above and insert it into $S(\mathbf{p})$.

In the next step we generate M random directions ω_j^{in} using BRDF importance sampling at $\mathbf{p}$. We map these directions to $\mathcal{D}$ and for each direction ω_j^{in} we collect nearby radiance samples stored in the contributing records' L-trees. These samples are used for the directional interpolation.

Given a direction ω_j^{in} and an L-tree i, we use a range query to collect nearby radiance samples stored in the L-tree as shown in Figure 2.2. The query radius $r_d(\omega_j^{\text{in}})$ is given by:

$$r_d(\omega_j^{\text{in}}) = \min\left\{ r_{\max}, \frac{1}{2\pi}\frac{1}{M\sqrt{p(\omega_j^{\text{in}})}} \right\},$$

where $p(\omega_j^{\text{in}})$ is the probability density of the BRDF sampling in direction ω_j^{in}. The radius is designed to adapt both to the number of rays M and to the reflectance properties at $\mathbf{p}$. In particular, the radius will be smaller when the pdf value is high, i.e. around the peak of the BRDF lobe. We use a ceiling of $r_{\max} = 0.15$ to avoid errors when the pdf value is very small. The collected radiance samples are formally defined by the set $T_i(\overline{\omega_j^{\text{in}}})$:

$$T_i(\overline{\omega_j^{\text{in}}}) = \left\{ k \mid w_{ik}^d(\overline{\omega_j^{\text{in}}}) > 0 \right\},$$

The upper bar denotes the paraboloid mapping from the sphere to $\mathcal{D}$, i.e. $\overline{\omega}$ is ω mapped to $\mathcal{D}$. The *directional weight* $w_{ik}^d(\overline{\omega_j^{\text{in}}})$ of the k-th radiance sample in the

i-th L-tree with respect to $\overline{\omega_j^{\text{in}}}$ is defined as:

$$w_{ik}^d(\overline{\omega_j^{\text{in}}}) = \max\left\{0, 1 - \frac{\|\overline{\omega_{ik}^{\text{in}}} - \overline{\omega_j^{\text{in}}}\|^2}{r_d(\omega_j^{\text{in}})^2}\right\},$$

where $\overline{\omega_{ik}^{\text{in}}}$ is the coordinate of the k-th radiance sample in the i-th L-tree.

All the collected radiance samples in sets $T_i(\overline{\omega_j^{\text{in}}})$ from all the contributing L-trees participate in the directional interpolation. Suppose we have two contributing L-trees. The first one is spatially close to $\mathbf{p}$ but the radiance sample found in it lies almost by the edge of the directional search radius. The second one has exactly the opposite property, i.e. it is spatially far from $\mathbf{p}$ but the radiance sample found is incident with the direction sample. For proper evaluation of both these cases it is necessary to find a relationship that correlates between the spatial and the directional weights. We use the following weighted sum:

$$\tilde{L}^{\text{in}}(\mathbf{p}, \omega_j^{\text{in}}) = \frac{\sum_{i \in S} \sum_{k \in T_i} w_i^s(\mathbf{p}) w_{ik}^d(\overline{\omega_j^{\text{in}}}) L_{ik}^{\text{in}}}{\sum_{i \in S} \sum_{k \in T_i} w_i^s(\mathbf{p}) w_{ik}^d(\overline{\omega_j^{\text{in}}})}$$

where L_{ik}^{in} is the k-th radiance sample stored in the i-th L-tree and $\tilde{L}^{\text{in}}(\mathbf{p}, \omega_j^{\text{in}})$ is the interpolated incoming radiance.

If, however, no radiance sample close to ω_j^{in} was found in any L-tree, a new radiance sample is computed using ray tracing. A question arises how to choose the ray origin and update the contributing L-trees. The first option is to place the ray origin at the interpolation location $\mathbf{p}$. However, such a sample cannot be used to update the contributing L-trees since that could potentially lead to its reuse at a too distant spatial location. Instead, one could create a new L-tree at $\mathbf{p}$ with all the added radiance samples. However, this approach generates many new L-trees with only a few radiance samples and the interpolation becomes inefficient.

The second option, which we use in our algorithm, is to place the ray origin at the location of one of the L-trees. The L-tree is selected randomly from among the contributing L-trees for which ω_j^{in} is above their tangent plane. The computed radiance sample is then inserted into the selected L-tree. A simple implementation of the insertions turned out to be the most efficient one: We keep a buffer of added radiance samples; when it is full, the tree is rebuilt. If no L-tree can be used for update, which occurs very rarely, we reject the sample $\overline{\omega^{\text{in}}}$. We have opted for the second technique since it distributes the additional radiance samples well among the existing L-trees.

2.4.5 Outgoing Radiance Computation

The interpolated incoming radiances $\tilde{L}^{\text{in}}(\mathbf{p}, \omega_j^{\text{in}})$ calculated for each ω_j^{in} are used to compute the outgoing radiance $\tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}})$. The value of $\tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}})$ at a point $\mathbf{p}$ in the direction ω^{out} is computed using the following Monte Carlo estimator:

$$\tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}}) = \frac{1}{M} \sum_{j=1}^{M} \frac{\tilde{L}^{\text{in}}(\mathbf{p}, \omega_j^{\text{in}}) f_r(\mathbf{p}, \omega_j^{\text{in}}, \omega^{\text{out}}) \cos\theta_j^{\text{in}}}{p(\omega_j^{\text{in}})} \tag{2.3}$$

where $f_r(\mathbf{p}, \omega_j^{\text{in}}, \omega^{\text{out}})$ is the BRDF at $\mathbf{p}$ and θ_j^{in} is the angle between the surface normal at $\mathbf{p}$ and ω_j^{in}.

2.4.6 Cache Record Density Control

For faithful reconstruction of indirect illumination, the distribution of the records should be proportional to its change rate. In the original irradiance cache algorithm the rate of change is estimated based on the information about surrounding geometry obtained during hemisphere sampling [WRC88]. Estimating the rate of change of indirect illumination for glossy surfaces is more difficult, though, since it should take into account the reflectance properties and the viewing direction. The formula is difficult to derive even for the simplest reflectance models. Instead, our interpolation criterion is based on the original formulas for diffuse surfaces with the addition of the following heuristics.

While creating a new cache record, we estimate the derivatives of outgoing radiance with translation, $\vec{\nabla}_x \tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}})$ and $\vec{\nabla}_y \tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}})$. The derivatives are used for clamping the harmonic mean distance R_i of the new record, similar to regular irradiance caching [KG09]:

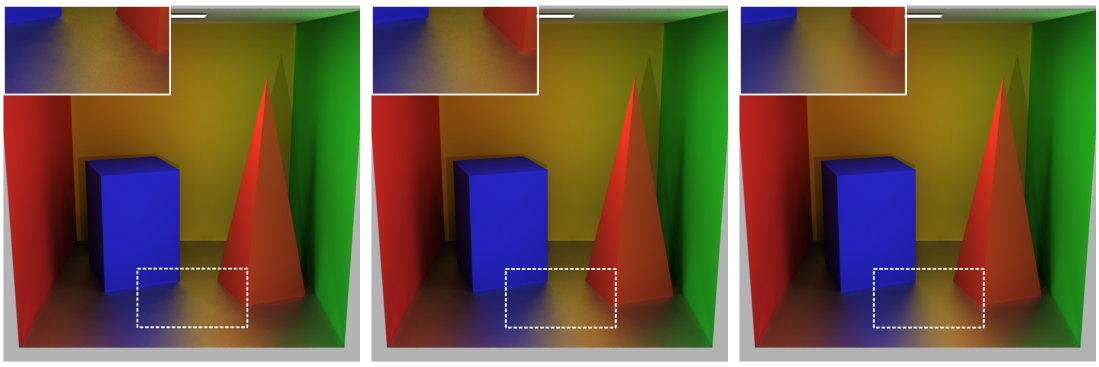
$$\textbf{if} \quad \triangle_i > 1/R_i, \quad \textbf{then} \quad R_i := 1/\triangle_i,$$

where

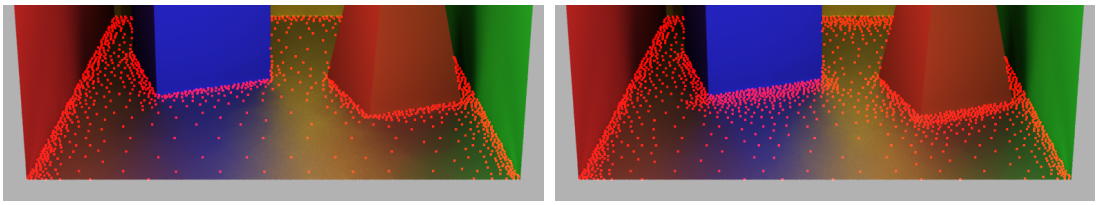
$$\triangle_i = \frac{\sqrt{\| \vec{\nabla}_x \tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}}) \|^2 + \| \vec{\nabla}_y \tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}}) \|^2}}{\tilde{L}^{\text{out}}(\mathbf{p}, \omega^{\text{out}})}.$$

This heuristic automatically decreases the radius of the cache record where indirect illumination changes quickly. Although the method is formally not correct, it gives plausible results in practice as shown in Figure 2.3.

Finite differences are used to estimate the derivatives of outgoing radiance. For each incoming radiance sample, we displace the ray origin along local x and y coordinate axes, update the ray direction, and re-evaluate the BRDF. We use the updated BRDF values to compute the approximation of the outgoing radiance



(a) No heuristic **(b)** With heuristic **(c)** Reference image



(d) No heuristic zoom out (1.08k recs) **(e)** With heuristic zoom out (1.43 recs)

Figure 2.3 – Cache record density control. The top images, from left to right, show (a) rendering of a simple scene using our algorithm without detection of high changes of indirect illumination (no heuristic), (b) rendering with the detection of high changes (with heuristic) and (c) a reference image. The images in the bottom row show the distribution of cache records with and without the gradient-based record density heuristic. Note the higher density of cache records in places with large changes of indirect illumination.

at the displaced positions. For the sake of simplicity we assume the incoming radiance does not change with the displacement (which may not be true if the contributing surface itself is glossy [JZJ08]).

2.5 Results

We compare our algorithm (spatial directional radiance caching, SDRC) with the original radiance caching algorithm as described in [KBPv06] (spherical harmonics radiance caching, SHRC) and Monte Carlo importance sampling (MC). The algorithms were implemented as plug-ins in the PBRT ray tracer [PH04]. All images were rendered on a Mac-Book Pro with Intel Core 2 Duo 2.40GHz (using one core). No optimizations using the GPU were used.

Images generated by the compared algorithms exhibit artifacts of very different kinds. Images generated using MC contain high-frequency noise. On the other hand, SHRC suffers from low-frequency error which can be seen as splotches

in the image. The SDRC exhibits both types of errors in lesser amplitude. As a result of the different nature of the errors, it is difficult to compare rendering times needed to obtain images of the same visual quality. Instead, we compare the image quality obtained by the algorithms for equal-time computation. For each image we show RMS error, although this error measure may not be quite meaningful in terms of visual quality.

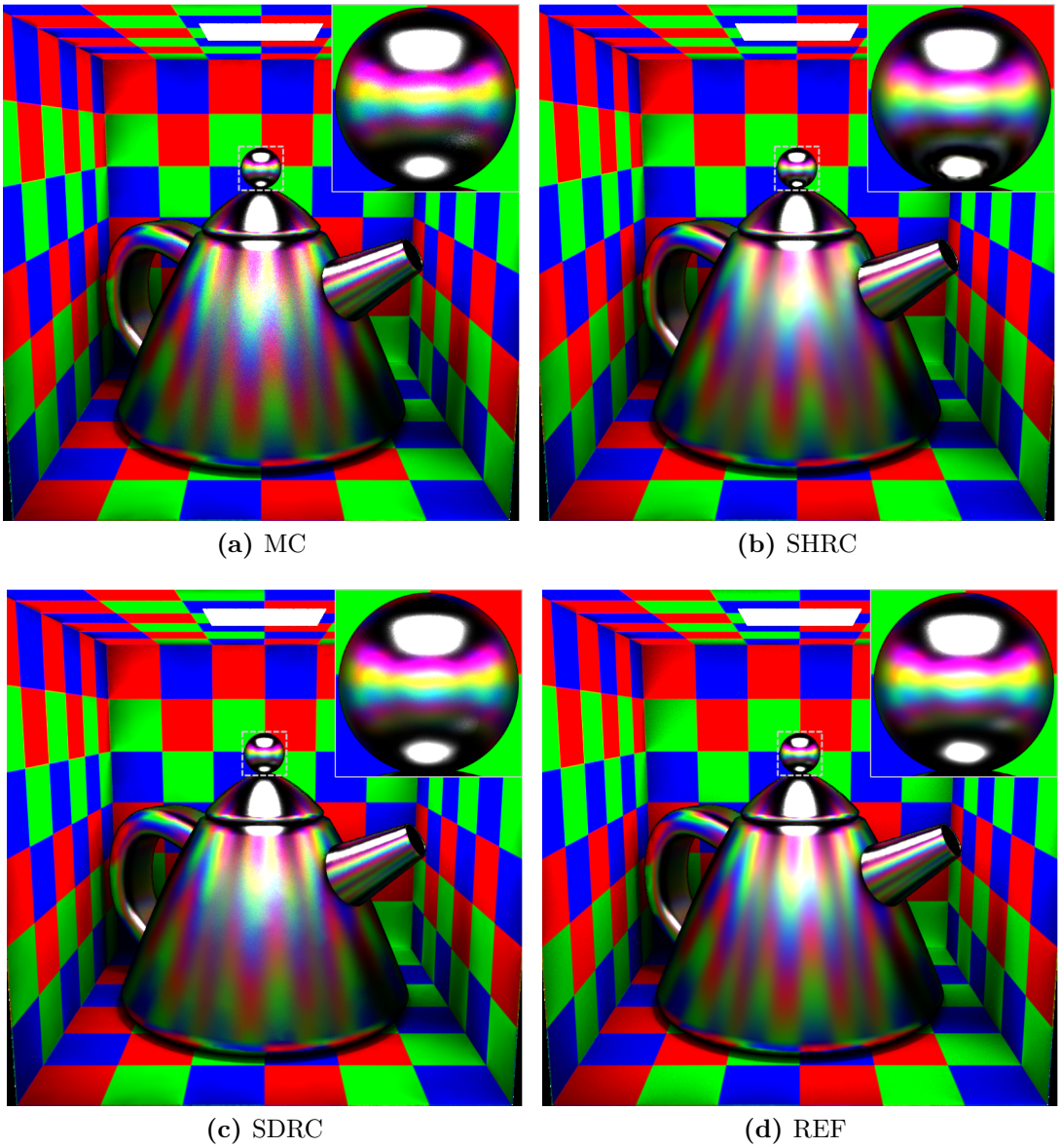
Images were rendered with the global illumination effects up to the fourth bounce indirect lighting. The first bounce was computed using the compared algorithms while photon mapping was used for the higher recursion levels. Irradiance cache was used to compute diffuse indirect lighting. We used the following default settings for the algorithms. In SHRC, we used spherical harmonics order of 10. The maximum allowed caching error was set to $a = 0.25$ and the number of rays for hemisphere sampling is set to ensure the same computation time. In SDRC, the default maximum allowed error was $a = 0.22$. Both the SHRC and SDRC used the neighbor clamping heuristic to improve spatial record distribution [KBPv06].

The SHRC and SDRC algorithms render the image in two passes. The first pass, rendered with one camera ray per pixel, populates the spatial cache. The second pass then uses four camera rays per pixel to generate the image. The approximation error a is increased 1.3 times in the second pass to improve the smoothness of interpolated illumination.

Kettle scene. Figure 2.4 shows renderings of a glossy kettle in a diffuse box. The kettle material is represented using anisotropic Ward's BRDF [Wal05] with the roughness of $\alpha_x = 0.18$ and $\alpha_y = 0.09$. The box walls are covered with a diffuse texture. The scene is lit by an area light source.

Uniform hemisphere sampling is not efficient for glossy surfaces with a narrow BRDF lobe. A lot of computation time can be saved using importance sampling as we do in the SDRC. The saved time can be used to set a smaller value for the allowed interpolation error for the SDRC, $a = 0.17$. Total number of records generated by SHRC is 2530; for SDRC, it is 4670. Rendering using the SDRC shows 1.58 times smaller RMS error than the SHRC and 1.26 times smaller than MC. See table 2.1 for details.

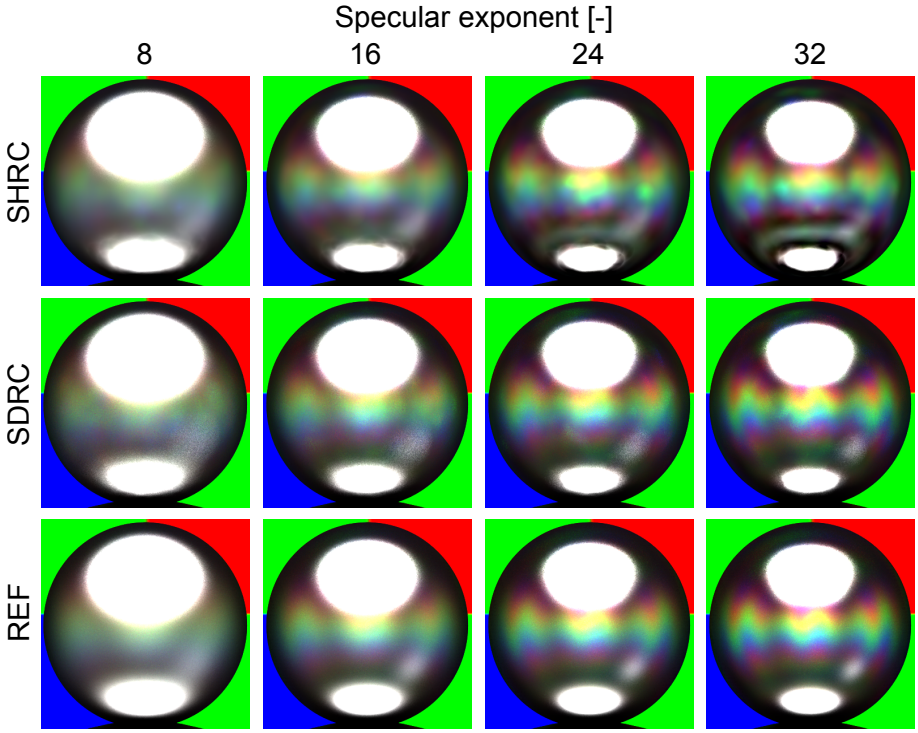
Figure 2.5 shows the RMS error produced by the SHRC and the SDRC as a function of the surface finish. For the sake of simplicity we use Phong's BRDF model instead of anisotropic one for the kettle material. We can see that the SHRC works well for a low BRDF lobe exponent. Importance sampling has no advantage compared with sampling the whole hemisphere. However, the bigger the exponent, the bigger the RMS error of SHRC. Spherical harmonics are not



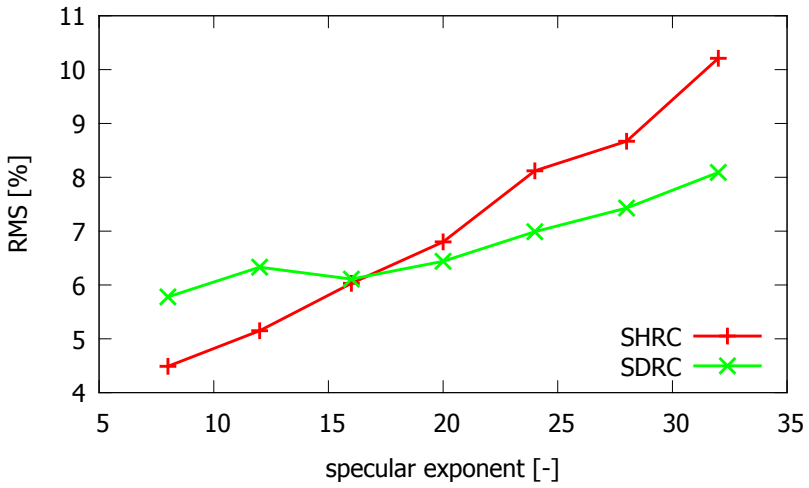
(a) MC **(b)** SHRC

(c) SDRC **(d)** REF

Figure 2.4 – Equal-time comparison of the rendering quality achieved using the MC, SHRC and SDRC algorithms. The images were rendered at a resolution of 800 × 800 in approximately the same time, 398 seconds. Indirect lighting computation on glossy surfaces up to the 4th bounce took approximately 171 seconds out of the total time. We can see that SDRC is able to faithfully simulate glossy BRDF on the kettle while the SHRC provides a blurrier image. The image rendered using MC exhibits a high noise level. Note that the walls of the Cornell Box are diffuse only. The irradiance cache was used to compute the indirect illumination term on them.



(a) Visual example of the the RMS dependency for the lobe exponent



(b) RMS dependency on the Phong's lobe exponent

Figure 2.5 – Renderings for different surface finish. The image in 2.5b shows the RMS dependency on the Phong's lobe exponent. The SHRC is suitable for the indirect term computation up to the lobe exponent of 16. For higher exponents, it blurs out the reflection thereby altering the material perception. Our algorithm, on the other hand, provides a sharper image for higher lobe exponent. The image in 2.5a shows a visual example of the RMS error dependency for the lobe exponent. The details show a glossy reflection of the checkered walls on the handle of the teapot lid in Figure 2.4.

Scene	Method	#Sec rays/ pixel	#Records	#Rays/ record	#Additional rays	Memory	RMS error	Time	Total time
Teapot kettle	MC	62.6	-	-	-	-	1.70	168	385
	SHRC	47.7	2.53k	4608	-	9.1M	2.13	174	391
	SDRC	12.8	4.67k	512	101k	19.9M	1.35	171	388
Chess scene	MC	90.7	-	-	-	-	6.56	366	469
	SHRC	54.0	16.0k	2244	-	57.6M	10.25	369	472
	SDRC	22.2	26.1k	512	306k	109.5M	5.50	360	463
Flamingos	MC	41.6	-	-	-	-	3.23	416	625
	SHRC	26.5	14.7k	1986	-	52.9M	4.98	407	616
	SDRC	11.0	22.2k	512	638k	96.2M	2.73	410	619

Table 2.1 – Rendering settings and statistics for the example scenes. The columns list the number of secondary rays traced per pixel, the number of spatial records generated, the number of rays traced to create a new cache record, the number of rays traced to update the L-trees and the memory requirements. The rightmost section of the table shows the RMS error, the time spent on the indirect term computation on glossy surfaces and the total rendering time. The difference between 'Time' and 'Total time' consists of the photon map construction, primary ray casting, direct lighting computation and the irradiance caching on diffuse surfaces.

able to approximate a narrow BRDF lobe accurately. In this case, it is preferable to use the SDRC for the exponent higher than 16.

Chess scene. Figure 2.6 shows the renderings of the chess scene. White chess pieces are represented using Ward's BRDF model with the roughness of $\alpha_x = 0.10$ and $\alpha_y = 0.16$. Black ones are represented using Phong's BRDF model with the lobe exponent of 30. The chessboard is represented using the same model with the lobe exponent of 14. The scene is lit by 5 spot lights and by an environment map of the sky.

The chess scene presents a challenge for the SHRC because of a lot of glossy and curved surfaces. On such surfaces, cache records cannot be reused at many pixels. In addition, the interpolation of the cache record require the costly rotation. See table 2.1 for the rendering settings and the number of the records generated by the caching algorithms. Note that the rendering using the SDRC shows 1.86 times smaller the RMS error than SHRC and 1.19 times smaller than MC.

Flamingos. Figure 2.7 shows renderings of a scene with glossy flamingos. The flamingos are represented using the Ward's BRDF model with the roughness in the range of 0.08 to 0.20. Other surfaces are purely diffuse. Table 2.2 shows the scalability of the rendering time to the number of directional rays N used.

Figure 2.6 – Renderings of the chess scene. Images were rendered at a resolution of 1024×768 in approximately the same time, 468 seconds. The indirect term computation on glossy surfaces took 365 seconds. Note the sharper glossy reflection of black pieces on the white chess piece in the details. Chess pieces courtesy of Toshiya Hachisuka.

The rendering time spent for indirect lighting computation on glossy surfaces is directly proportional to N.

	N	32	64	128	256	512
SDRC	Time	43	75	150	273	410
	RMS	5.48	4.85	3.88	3.19	2.81
MC	Time	50	77	156	266	416
	RMS	8.71	7.32	5.00	3.76	3.23

Table 2.2 – Rendering time scalability. The rows list the number of directional samples used, rendering time spent for indirect lighting computation on glossy surfaces and the RMS error for both the SDRC and MC algorithms.

2.6 Discussion and Limitations

View dependency. Unlike in the original radiance caching, the cache records generated by our algorithm contain useful information only within some parts of the hemisphere—they are view-dependent. Hence, it may seem problematic to reuse them for rendering new views of the scene. However, since new radiance samples are added to the L-trees on the fly as needed, our algorithm behaves in a view-independent manner overall.

Interpolation. The idea of sparse illumination computation and interpolation is based on the assumption that indirect illumination changes slowly on surfaces. However, as the shininess of the reflections goes up, this assumption is less valid. For sharp reflections, Monte Carlo importance sampling performs better than our algorithm.

Gradients. In irradiance and radiance caching, the use of rotation and translation gradients of the incoming illumination significantly improves the interpolation quality. The *rotation* gradient is not necessary in our algorithm due to the way we access individual radiance samples. We have implemented the *translation* gradient according to [KGPB05]. However, the gradient-based interpolation becomes fairly involved. In our scenes, the computational overhead did not pay for the image quality improvement, which is why we decided against the use of translation gradients. Instead, we generate the image in two passes. Increasing the allowed interpolation error a in the second pass generates a smoothed image even without the use of gradients.

Figure 2.7 – Rendering time scalability. The images were rendered at a resolution of 1024×768. Images on the left shows rendering quality of the SDRC for the value of N set to $N = 64, 128$ and 256. Images on the right were rendered using the MC in approximately the same time for the indirect term computation on glossy surfaces, as the top images, 76, 153 and 269 seconds, respectively.

Supported materials. The original radiance caching algorithm is difficult to use on spatially varying glossy surfaces since the scene BRDFs have to be projected onto the spherical harmonics basis. Our algorithm does not require any special BRDF representation, although many sudden changes in material properties may reduce its efficiency due to frequent L-tree updates. Furthermore, spatial directional caching relies on the availability of an efficient and effective sampling procedure for the BRDFs—it cannot be used directly with measured BRDF data without first fitting a model that can be sampled.

2.7 Conclusion

This chapter describes a new algorithm for indirect lighting computation on glossy surfaces. The algorithm adopts the lazy illumination evaluation scheme used in the irradiance and radiance caching algorithms and extends it from the spatial to the directional domain. Explicit storage of directional incoming radiance samples allows us to exploit BRDF importance sampling for noise reduction and still retain the overall view-independent nature of the algorithm.

Our new caching algorithm outperforms the original radiance caching for scenes with shiny surfaces, where radiance caching produces blurring of reflections or banding artifacts. Compared to Monte Carlo importance sampling, our algorithm produces less noisy images in the same time. The main disadvantages of our algorithm is a higher memory demand and potentially difficult parallelization due to the continual updates of cache records.

In future work, we want to devise a more accurate interpolation criterion for glossy surfaces. Not only should such a criterion adapt to the rate of change of the indirect illumination but also to the surface reflectance properties. In addition we would like to investigate the correlation of illumination coherence in the spatial and directional domains and address flickering in animation rendering.

Improving Performance and Accuracy of Local PCA

3

Contents

In this chapter we focus on improving the efficiency and accuracy of *local principal component analysis* (LPCA) used for dimensionality reduction and data compression of large data sets encountered in computer graphics. We tested our approach for compression of radiance transfer matrices in *precomputed radiance transfer* (PRT) and of *bi-directional texture function* (BTF) showing a speed-up of 5 to 20 on the first data sets and lower on the second. The work described in this chapter was published in [GKB11].

3.1 Introduction

Precomputation-based and data-driven approaches have emerged in computer graphics as important tools for improving rendering performance and increasing image fidelity [Ram09, FH09]. Common to these techniques is the need to compress large data sets consisting of high-dimensional data points. One of the successful data compression approaches has been the so called *local principal component analysis* (LPCA), also known as *clustered PCA* (CPCA) [KL97]. LPCA was shown to be effective for compression of transfer matrices in pre-computed radiance transfer [SKS02, SHHS03, HR10] or of Bidirectional texture function (BTF) data sets [FH09, MMK03]. However, slow performance of the data compression (often several hours for a single data set) is a serious bottleneck in the data processing pipeline. For instance, though PRT may be a useful tool for scene relighting, the long pre-computation times (including the data compression) may hinder its practical applicability. Quite surprisingly, performance of data compression has been largely unaddressed in previous computer graphics research. Huang and Ramamoorthi [HR10] do accelerate the transfer matrix precomputation, however, they still rely on the slow LPCA algorithm to compress the resulting data set.

In this paper we improve both performance and data approximation accuracy of the LPCA algorithm. Being a variant of k-means clustering, the bottleneck of LPCA consists in the repeated classification of data points to the nearest clusters. Due to the high data dimensionality, simple approaches such as spatial data indexes [PM99, KMN+02] are ineffective at accelerating this process. Nonetheless, we found that a significant speed-up can be achieved by taking advantage of the information gathered as the algorithm progresses to eliminate some point-cluster distance calculations that provably cannot change current point-cluster assignment. Though this simple and effective idea (known as SortMeans) has been previously used to accelerate k-means clustering [Phi02, Elk03], to our knowledge our work is the first to investigate its use in computer graphics. Our main contribution, however, consists in extending the SortMeans algorithm [Phi02] from the simple k-means problem (where each cluster is represented by one mean point) to the LPCA problem (where a cluster is represented by an affine subspace). In addition, we improve the approximation accuracy of LPCA by seeding the clusters using the k-means++ algorithm [AV07]. To minimize the performance impact of this advanced initialization, we apply a variant of SortMeans to cut some of the unnecessary distance calculations. We present extensive measurements of performance and approximation accuracy of the presented algorithms for four radiance transfer matrices (similar to [HR10]) and three BTF data sets

from the University of Bonn database. The speed-up of classification is 5 to 20 for the coherent transfer matrices and 1.2 to 1.6 for the complex BTF data. We show that the k-means++ algorithm substantially improves the approximation accuracy for the investigated data sets. Our results could be useful both in and outside of computer graphics.

3.2 Related Work

LPCA in computer graphics. Local principal component analysis (LPCA) was devised by Kambhatla and Leen [KL97] in the machine learning community. It was introduced to computer graphics (under the name Clustered PCA) by Sloan et al. [SHHS03] who used it for the compression of low-frequency light transfer matrices in precomputed radiance transfer [SKS02]. LPCA was later used for all-frequency light transfer on glossy surfaces [LSSS04, MSRB07, XJF+08, HR10]. Mahajan et al. [MSRB07] adapt the clusters to optimize a rendering performance metric, and use a hierarchical splitting algorithm instead of k-means-based clustering. Huang and Ramamoorthi [HR10] achieve a substantial speed-up of LPCA compression by reducing the size of the input data set. Our work, on the other hand, focuses on the acceleration of the LPCA algorithm itself and is complementary to their work. LPCA is also one of the most widely used algorithms for BTF data compression because it provides both high compression ratio and low approximation error [FH09, MMK03].

Acceleration of k-means and related methods. Despite a wide range of LPCA applications, we are not aware of any work on improving the algorithm's performance. Most of the previous research has focused on the acceleration of the simpler k-means clustering. The idea is to reduce the number of distance calculations when classifying data points to their nearest cluster. Some works organize the data points in a spatial data structure such as kd-tree [PM99, KMN+02, Moo00]. With the exception of [Moo00] these approaches are effective only for low-dimensional data points. Other works are based on using the triangle inequality to avoid unnecessary point-cluster distance calculation. Works of Hodgson [Hod88] and Phillips [Phi02] propose the use of an upper bound on the distance between a data point and a cluster. Our algorithm is based on their ideas, specifically on Phillips' SortMeans algorithm. We generalize the algorithm to the more complex LPCA problem. More recent works [Elk03, Ham10] add the use of a lower bound on the point-cluster distance to make k-means clustering even faster. The computation of the lower bound is, however, not trivial in the case of LPCA, and we do not use it.

Initialization of k-means. The k-means algorithm is prone to getting stuck in local minima of the objective function thereby producing suboptimal clustering results. Random restarts [KMN+02] partially alleviate the problem, however at a high computational cost. More advanced initialization algorithms attempt to seed the centroids close to the target clustering. The farthest-first heuristic [HS85] tends to place the initial centroids on the outer hull of the space subtended by the data points. The state-of-the-art k-means++ algorithm [AV07] achieves better results by randomizing this process. In spite of using a more advanced heuristic for choosing the first two centroids, the variant of k-means++ described by Ostrovsky et al. [ORSS06] did not significantly improve the results in our experiments.

3.3 Preliminaries

3.3.1 Problem definition

We start by formally defining the data approximation problem that we address in this work. Let the input data set $\mathcal{X} \subset V^d$ be a subset of a d-dimensional linear space V^d with the dot product $\langle \cdot\ ,\ \cdot \rangle$, and let k and l be integers. The goal is to choose k affine subspaces $\mathbf{a}_1, \ldots, \mathbf{a}_k \subset V^d$ of dimension up to l, $l < d$ such that the following objective function is minimized:

$$\phi = \sum_{\mathbf{x} \in \mathcal{X}} \min_{j=1}^{k} d(\mathbf{x}, \mathbf{a}_j).$$

Here $d(\mathbf{x}, \mathbf{a})$ is the distance of data point $\mathbf{x} \in \mathcal{X}$ from affine subspace $\mathbf{a}$. The assignment of $\mathbf{x}$ to their nearest affine subspaces induces a clustering (partition) of the data set $\mathcal{X}$.

The distance $d(\mathbf{x}, \mathbf{a})$ is defined as [KL97]:

$$d(\mathbf{x}, \mathbf{a}) = \|(\mathbf{x} - \mu(\mathbf{a})) - \mathbf{x}_s\|, \tag{3.1}$$

where $\mu(\mathbf{a})$ is the origin of $\mathbf{a}$, $\mathrm{dir}(\mathbf{a})$ is its basis, and $\mathbf{x}_s$ is the orthogonal projection of $(\mathbf{x} - \mu(\mathbf{a}))$ onto $\mathbf{a}$:

$$\mathbf{x}_s = \sum_{i=1}^{l} \langle \mathbf{x} - \mu(\mathbf{a}), \mathrm{dir}_i(\mathbf{a}) \rangle \ \cdot \mathrm{dir}_i(\mathbf{a}). \tag{3.2}$$

3.3.2 Local Principal Component Analysis (LPCA)

Finding the optimal solution of the above problem is NP-hard, even just for two clusters of zero dimension [DFK+04]. To find an approximate solution, Kambhatla and Leen [KL97] proposed the *local principal component analysis* (LPCA)

algorithm as a modification of the classical k-means algorithm (which solves the problem defined above in affine subspaces with dimension of $l = 0$). The LPCA algorithm proceeds as follows. In the first step, initial centroids for the clusters are selected. Each data point $\mathbf{x} \in \mathcal{X}$ is assigned to the nearest (in terms of Euclidean distance) cluster. In the second step, affine subspaces are found within each cluster and are used as a low-dimensional approximation of the data points assigned to it. Both steps are repeated several times. Then the dimension of the affine subspaces is increased and the whole previous computation is performed again until the desired dimension of the subspaces is reached [SHHS03].

Thanks to its simplicity, LPCA is one of the most widely used algorithms in computer graphics for approximation of high-dimensional signals. However, LPCA is computationally demanding when used for classification of high-dimensional points into a high number of clusters. The inefficiency comes from the fact that no information is passed from one stage to another: the point classification computes distances to *all* PCA subspaces for each data point. Our goal is to avoid unnecessary distance calculations by exploiting the information computed in the previous iteration. Our accelerated algorithm produces exactly the same clustering as the original LPCA algorithm, but more quickly.

3.3.3 SortMeans: Acceleration of k-means Clustering

Our algorithm is based on the SortMeans algorithm proposed by Phillips [Phi02] for accelerating the k-means problem, which is an instance of the data approximation problem defined above, where the dimension of the affine subspaces is $l = 0$. Phillips uses the triangular inequality to avoid unnecessary point-cluster distance computations in the classification stage. For an arbitrary point $\mathbf{x} \in \mathcal{X}$ and two clusters c_a and c_b represented by their centroids $\mu(\mathrm{c}_a)$ and $\mu(\mathrm{c}_b)$, respectively, the triangle inequality says

$$d(\mu(\mathrm{c}_a), \mu(\mathrm{c}_b)) \leq d(\mathbf{x}, \mu(\mathrm{c}_a)) + d(\mathbf{x}, \mu(\mathrm{c}_b))$$

or, equivalently

$$d(\mathbf{x}, \mu(\mathrm{c}_b)) \geq d(\mu(\mathrm{c}_a), \mu(\mathrm{c}_b)) - d(\mathbf{x}, \mu(\mathrm{c}_a)),$$

where $d(\,\cdot\,,\,\cdot\,)$ is the Euclidean distance between two data points in V^d. Therefore if one knows that $d(\mu(\mathrm{c}_a), \mu(\mathrm{c}_b)) \geq 2d(\mathbf{x}, \mu(\mathrm{c}_a))$, then $d(\mathbf{x}, \mu(\mathrm{c}_b)) \geq d(\mathbf{x}, \mu(\mathrm{c}_a))$, e.g. the distance of $\mathbf{x}$ to c_b cannot be lower than the one to c_a and the computation of $d(\mathbf{x}, \mu(\mathrm{c}_b))$ can be safely avoided, see Figure 3.1.

We briefly summarize the SortMeans algorithm [Phi02] built around this idea. At the beginning of each iteration SortMeans precomputes two $k \times k$ matrices (k is the number of clusters), a distance matrix $\mathbf{D}$ and a permutation matrix

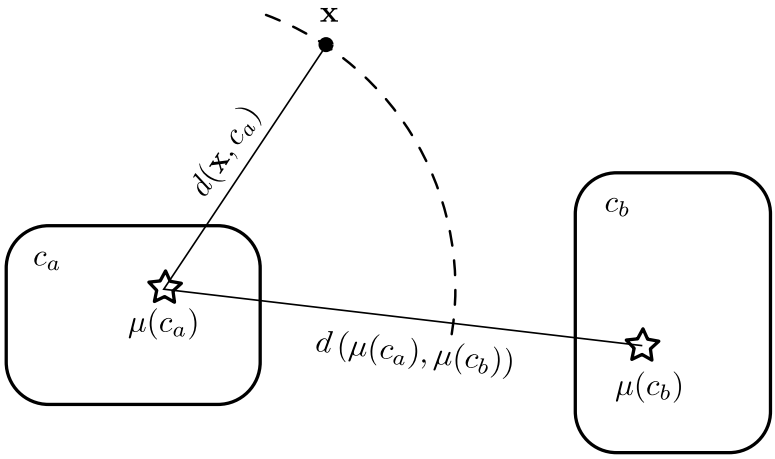


Figure 3.1 – Triangle inequality. A decision if $\mathbf{x}$ is closer to a cluster c_a or c_b have to be made and let $d(\mathbf{x}, \mu(\mathrm{c}_a))$ and $d(\mu(\mathrm{c}_a)), \mu(\mathrm{c}_b))$ are known distances. The triangle inequality says if $d(\mu(\mathrm{c}_a), \mu(\mathrm{c}_b)) \geq 2d(\mathbf{x}, \mu(\mathrm{c}_a))$ then $d(\mathbf{x}, \mu(\mathrm{c}_b)) \geq d(\mathbf{x}, \mu(\mathrm{c}_a))$, eliminating the need to calculate distance $d(\mathbf{x}, \mu(\mathrm{c}_b))$.

$\mathbf{M}$. Elements of $\mathbf{D}$ are defined as $\mathbf{D}(i, j) = d(\mu(\mathrm{c}_i), \mu(\mathrm{c}_j))$. Rows of $\mathbf{M}$ represent permutations on the set of cluster indices such that $\mu(\mathrm{c}_{\mathbf{M}(i,j)})$ is the j-th nearest centroid from $\mu(\mathrm{c}_i)$.

For each data point $\mathbf{x} \in \mathcal{X}$, which was assigned to c_i in the previous iteration, the algorithm iterates over all other clusters, keeping the minimal distance found so far, denoted $d_{\min}$ (which is initially set to $d(\mathbf{x}, \mu(\mathrm{c}_i))$). Clusters are iteratively processed in increasing order of their distances from $\mu(\mathrm{c}_i)$ using $\mathbf{M}$. If $\mathbf{D}(i, \mathbf{M}(i, j)) \geq d(\mathbf{x}, \mu(\mathrm{c}_i)) + d_{\min}$ the nearest cluster has been already found and the iteration is stopped. Otherwise the actual distance $d(\mathbf{x}, \mu(\mathrm{c}_{\mathbf{M}(i,j)}))$ is computed. If $d(\mathbf{x}, \mu(\mathrm{c}_{\mathbf{M}(i,j)})) < d_{\min}$, $\mathrm{c}_{\mathbf{M}(i,j)}$ becomes the new nearest cluster and $d_{\min} = d(\mathbf{x}, \mu(\mathrm{c}_{\mathbf{M}(i,j)}))$ is the new best minimal distance. The computation above is then repeated for the next nearest cluster $\mathrm{c}_{\mathbf{M}(i,j+1)}$ (if there is any, i.e. if $j < k$). The process is illustrated in Figure 3.2.

The SortMeans algorithm was used for accelerating k-means clustering. In the next section we use it as a basic building block for developing our new SortClusters LPCA algorithm for accelerating the more general LPCA problem.

3.4 SortClusters LPCA: Acceleration of General LPCA

In this section we develop our SortClusters LPCA algorithm as a generalization of the SortMeans algorithm [Phi02] described above. Since clusters in LPCA are defined by affine subspaces rather than just centroids, we need a definition of

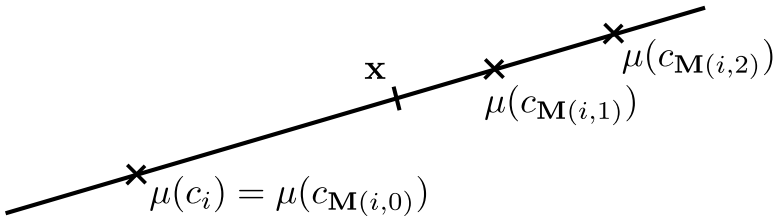


Figure 3.2 – Example of point-cluster classification. Let $\mathbf{x} \in \mathcal{X}$ be a data point that was assigned to cluster c_i in the previous iteration, and let $\mu(\mathrm{c}_{\mathbf{M}(i,1)})$ and $\mu(\mathrm{c}_{\mathbf{M}(i,2)})$ be the 1st and 2nd nearest clusters from c_i. $d_{\min}$ is initialized to $d(\mathbf{x}, \mu(\mathrm{c}_i))$. Inequality $d(\mu(\mathrm{c}_i), \mu(\mathrm{c}_{\mathbf{M}(i,1)}) \geq d(\mathbf{x}, \mu(\mathrm{c}_i)) + d_{\min}$ is checked. Since it does not hold, the true distance $d(\mathbf{x}, \mu(\mathrm{c}_{\mathbf{M}(i,1)}))$ is computed. The distance is less than $d_{\min}$ so $\mathrm{c}_{\mathbf{M}(i,1)}$ becomes the new nearest cluster and $d_{\min} \leftarrow d(\mathbf{x}, \mu(\mathrm{c}_{\mathbf{M}(i,1)}))$. In the second iteration $d(\mu(\mathrm{c}_i), \mu(\mathrm{c}_{\mathbf{M}(i,2)}) \geq d(\mathbf{x}, \mu(\mathrm{c}_i)) + d_{\min}$ is checked. The inequality now holds and the classification ends.

distance between two subspaces. The subspace distance is then used to develop a generalized triangle inequality for a data point and two affine subspaces. Such a triangle inequality is in turn used in our SortClusters LPCA algorithm to avoid unnecessary point-subspace distance calculations.

3.4.1 Distance Between Affine Subspaces and The Generalized Triangle Inequality

Let $\mathbf{a}_a, \mathbf{a}_b$ be arbitrary non-empty affine subspaces in V^d. The distance between $\mathbf{a}_a$ and $\mathbf{a}_b$ is defined as [DK92]:

$$d(\mathbf{a}_a, \mathbf{a}_b) = \inf \{\|\mathbf{p} - \mathbf{q}\|;\ \mathbf{p} \in \mathbf{a}_a, \mathbf{q} \in \mathbf{a}_b\}\,. \tag{3.3}$$

Intuitively, the distance between two affine subspaces is equal to the length of the shortest line that is orthogonal to both subspaces.

Lemma: The following *generalized triangle inequality* holds for an arbitrary point $\mathbf{x} \in V^d$ and arbitrary non-empty affine subspaces $\mathbf{a}_a, \mathbf{a}_b \subset V^d$:

$$d(\mathbf{a}_a, \mathbf{a}_b) \leq d(\mathbf{x}, \mathbf{a}_a) + d(\mathbf{x}, \mathbf{a}_b). \tag{3.4}$$

Proof. For an arbitrary point $\mathbf{x} \in V^d$ using (3.3) we have

$$\begin{aligned} d(\mathbf{a}_a, \mathbf{a}_b) &= \inf \{\|\mathbf{p} - \mathbf{q}\|;\ \mathbf{p} \in \mathbf{a}_a, \mathbf{q} \in \mathbf{a}_b\} \\ &\leq \inf \{\|\mathbf{p} - \mathbf{x}\| + \|\mathbf{x} - \mathbf{q}\|;\ \mathbf{p} \in \mathbf{a}_a, \mathbf{q} \in \mathbf{a}_b\} \\ &= \inf \{\|\mathbf{p} - \mathbf{x}\|;\ \mathbf{p} \in \mathbf{a}_a\} + \inf \{\|\mathbf{x} - \mathbf{q}\|;\ \mathbf{q} \in \mathbf{a}_b\} \\ &= d(\mathbf{x}, \mathbf{a}_a) + d(\mathbf{x}, \mathbf{a}_b). \end{aligned}$$

3.4.2 The SortClusters LPCA Algorithm

We use the generalized triangle inequality (3.4) to reduce the number of point-subspace distance calculations performed during the classification stage of LPCA in a similar way as in the SortMeans algorithm. The structure of our SortClusters LPCA (SC-LPCA) algorithm is similar to the SortMeans algorithm. However, instead of computing distances with respect to the clusters' centroids, we compute distances with respect to affine subspaces. Following the SortMeans algorithm we construct the distance and permutation matrices, $\mathbf{D}$ and $\mathbf{M}$, at the beginning of the classification stage. Elements of $\mathbf{D}$ are set to distances between pairs of affine subspaces. After that, we loop over all data points and for each we find the nearest affine subspace. The pseudo-code for searching the nearest affine subspace for a given point is given in Algorithm 2. The algorithm takes a data point $\mathbf{x} \in \mathcal{X}$, which was assigned to the i-th affine subspace in the previous iteration, and returns the index of the current nearest affine subspace.

Algorithm 2 Point-subspace classification

input : $\mathbf{x}$ ... data point to be classified
i ... index of the affine subspace to which $\mathbf{x}$ was assigned to in the previous iteration

output: index of the new nearest affine subspace

$d_{\min} \leftarrow d(\mathbf{x}, \mathbf{a}_i)$
$i_{\min} \leftarrow i$
for $j \leftarrow 2$ ***to*** k **do**
  if $(\mathbf{D}(i, \mathbf{M}(i,j)) \geq d(\mathbf{x}, \mathbf{a}_i) + d_{\min})$ **then**
    break
  end
  dist $\leftarrow d(\mathbf{x}, \mathbf{a}_{\mathbf{M}(i,j)})$
  if $(\text{dist} < d_{\min})$ **then**
    $d_{\min} \leftarrow$ dist
    $i_{\min} \leftarrow \mathbf{M}(i,j)$
  end
end
return $i_{\min}$

3.4.3 Efficient Evaluation of Inter-Subspace Distance

Determination of the distance between two subspaces defined by (3.3) can be converted to the computation of the distance between a point and a linear subspace using the following Lemma [DK92]: Let $\mathbf{a}_a, \mathbf{a}_b \subset V^d$ be non-empty affine subspaces. Then for arbitrary points $\mathbf{p} \in \mathbf{a}_a$ and $\mathbf{q} \in \mathbf{a}_b$ the distance between $\mathbf{a}_a$ and $\mathbf{a}_b$ is equal to

$$d(\mathbf{a}_a, \mathbf{a}_b) = d(\mathbf{p} - \mathbf{q},\ \mathrm{dir}(\mathbf{a}_a) \cup \mathrm{dir}(\mathbf{a}_b)). \tag{3.5}$$

The above Lemma gives us a recipe to compute the distance between non-empty affine subspaces $\mathbf{a}_a, \mathbf{a}_b \in V^d$. We use Equation (3.1) to solve Equation (3.5). We put $\mathbf{p} - \mathbf{q} = \mu(\mathbf{a}_a) - \mu(\mathbf{a}_b)$ as $\mathbf{x}$ and use the affine subspace with origin of $\mathbf{0}$ and basis of $\mathrm{dir}(\mathbf{a}_a) \cup \mathrm{dir}(\mathbf{a}_b)$ as $\mathbf{a}$. To evaluate Equation (3.1) we need to compute $\mathbf{x}_s$. It requires finding an orthonormal basis $\mathrm{dir}(\mathbf{a}_a) \cup \mathrm{dir}(\mathbf{a}_b)$ followed by the projection of $\mu(\mathbf{a}_a) - \mu(\mathbf{a}_b)$ onto this basis using (3.2). Orthonormalization is, however, a computationally demanding task. Instead, we can compute $\mathbf{x}_s$ directly without explicit construction of the orthonormal basis as shown below. Please, see [DK92] for more details.

Let m and n be the dimension of $\mathrm{dir}(\mathbf{a}_a)$ and $\mathrm{dir}(\mathbf{a}_b)$, respectively. To compute $\mathbf{x}_s$ we need to find a solution $\mathbf{z} = (z_1, \ldots, z_m, z_{m+1}, \ldots, z_{m+n})^T \in \mathbb{R}^{m+n}$ of the following linear system and set $\mathbf{x}_s = \mathbf{K} \cdot \mathbf{z}$:

$$\mathbf{G}(\mathbf{K}) \cdot \mathbf{z} = \mathbf{K}^T \cdot (\mu(\mathbf{a}_a) - \mu(\mathbf{a}_b)),$$

where $\mathbf{K} = [\mathrm{dir}_1(\mathbf{a}_a), \ldots, \mathrm{dir}_m(\mathbf{a}_a), \mathrm{dir}_1(\mathbf{a}_b), \ldots, \mathrm{dir}_n(\mathbf{a}_b)]$, and $\mathbf{G}(\mathbf{K})$ is the Gram matrix of all inner products of $\mathbf{K}$. We solve the linear system through the Cholesky decomposition of $\mathbf{G}(\mathbf{K})$, since $\mathbf{G}(\mathbf{K})$ is positive semi-definite. Finally, we compute the distance between $\mathbf{a}_a$ and $\mathbf{a}_b$ as $d(\mathbf{a}_a, \mathbf{a}_b) = \|\mu(\mathbf{a}_b) - \mu(\mathbf{a}_a) - \mathbf{x}_s\|$.

3.5 Cluster Initialization

The very first step of the LPCA (and k-means) algorithm is to choose k initial cluster centroids among the data points. We found that the initialization has a significant impact both on the approximation accuracy and—quite surprisingly—on the performance of our SortClusters LPCA algorithm. We have investigated random initialization with uniform probability [SHHS03], random initialization based on distance sums inpired by [HPB07], and the state-of-the-art k-means++ algorithm [AV07].

Distance sums-based initialization. For each data point $\mathbf{x}_i$, we compute the sum of squared distances to other data points, $\alpha_i = \sum_{\mathbf{x}\in\mathcal{X}} \|\mathbf{x} - \mathbf{x}_i\|^2$, and use it as a probability distribution for picking the k initial cluster centroids.

k-means++. The first centroid is chosen from among the data points at random with uniform probability. Distance to all other data points is then used as a probability distribution for choosing the second centroid. When a new centroid is chosen, all remaining data points are classified to their nearest centroid. The updated distance to the nearest centroid for each data point is then used as a probability distribution for choosing another centroid, and so on until we have k initial centroids.

The k-means++ initialization produces by far the most accurate data approximation, however the computation cost can be high (on a par with one iteration of k-means). Below, we describe our new SortMeans++ algorithm which uses the ideas on which the SortMeans algorithm is built to accelerate the k-means++ initialization.

3.5.1 SortMeans++: Accelerated k-means++

The structure of our SortMeans++ algorithm is similar to k-means++ but we eliminate some of the point-cluster distance calculations when a new cluster is created. For each data point, the algorithm maintains the distance to its nearest cluster in point $\mathbf{N}$ and the index of its nearest cluster in point $\mathbf{I}$. Similar to k-means++ we start by choosing the first centroid $\mathbf{x}_{\text{new}} \in \mathcal{X}$ at random with uniform probability. Point $\mathbf{N}$ is initialized with distances to this centroid, i.e. $\mathbf{N}(i) \leftarrow \|\mathbf{x}_{\text{new}} - \mathbf{x}_i\|$, and point $\mathbf{I}$ is initialized with all ones (index of the only existing cluster). The following steps are then repeated until the desired number of clusters is created.

1. Pick a new centroid $\mathbf{x}_{\text{new}} \in \mathcal{X}$ at random with probability proportional to $\mathbf{N}$, and create a new cluster c_{new} with $\mu(\mathrm{c}_{\text{new}}) = \mathbf{x}_{\text{new}}$.

2. Compute the distance from the new cluster to all existing clusters, $d_{\text{new}}(i) \leftarrow \|\mu(\mathrm{c}_{\text{new}}) - \mu(c_i))\|$.

3. Loop over all data points $\mathbf{x}_i \in \mathcal{X}$ and check if the new point c_{new} is closer than their currently assigned cluster. If $d_{\text{new}}(\mathbf{I}(i)) \geq 2\mathbf{N}(i)$, then the assignment of $\mathbf{x}_i$ cannot change and we can safely skip the calculation of the point-cluster distance $\|\mu(\mathrm{c}_{\text{new}}) - \mathbf{x}_i\|$. Otherwise, we calculate the distance and if it is smaller than $\mathbf{N}(i)$, we update $\mathbf{N}(i)$ and $\mathbf{I}(i)$. After that, we proceed to the next data point.

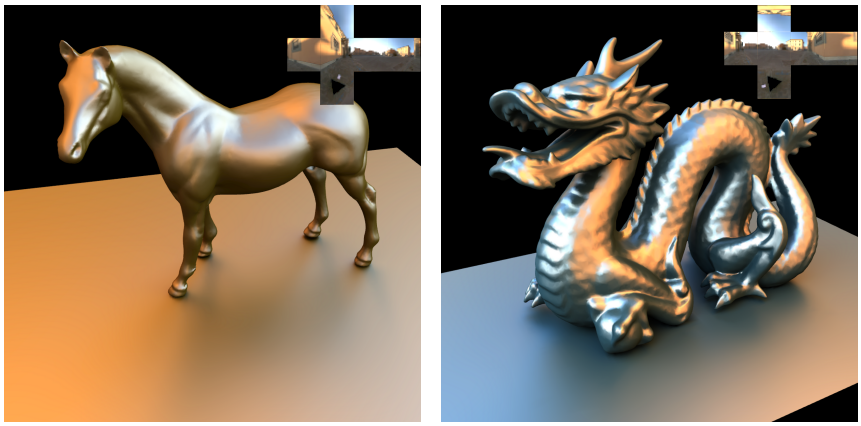
3.6 Results

We compare our algorithm (SortCluster LPCA) with the original LPCA [KL97] for compression of PRT and BTF datasets. Both algorithms were implemented in C++ using Intel MKL library for matrix computations. All measurements were performed on a PC with Intel Xeon W3540, 2.93 GHz and 14GB RAM. We use all 4 CPUs exploiting the parallelism of MKL routines yielding the CPU load of about 90% (as reported by Windows 7 Task Manager).

3.6.1 Compression of PRT data

We tested our algorithms on PRT data matrix precomputed for three different scenes shown in Figure 3.3. According to [NRH03] we use a raw lighting basis of cubemap pixel lights when computing the transfer matrix. The cubemap resolution is $6 \times 32 \times 32$. We account for self-shadowing effects only and we eliminate visibility alias by applying a 4×4 super-sampling to estimate the light transfer for each cubemap pixel. We used $k = 256$ clusters for the matrix compression, and up to $l = 24$ basis vectors for each cluster. According to Sloan et al. [SHHS03] we perform several iterations of LPCA before we increase the dimension of PCA clusters. More specifically, we use the following iterative scheme (0:15), (2:10), (4:7), (8:5), (12:4), (16:2), (24:1), where the first number is the maximum dimension of PCA subspaces and the second one is the number of iterations done for the same dimension. The plots in this section report the time for point classification only and do not include the time for computing the PCA approximation of the data points within clusters.

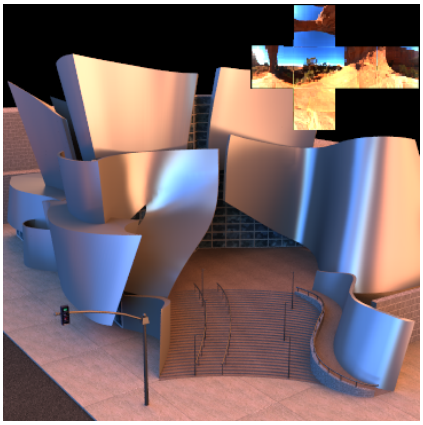
Scalability with the PCA dimension. Breakdown of the average computation times for one iteration of the classification using our SortClusters LPCA (SC-LPCA) and original LPCA as a function of the dimension of affine subspaces is shown in Figures 3.4. The average number of distances computed for both algorithms is shown in Figure 3.5. We can see that our SC-LPCA scales well with the dimension of affine subspaces. We achieve more than 20× speed up for the Horse scene while the speed-up for the more complex models (Dragon, Buddha, Disney) is about 6. The reason for higher speed-up for the Horse scene is presumably the higher degree of light transport coherence in this relatively simple scene. Note that the output of our SC-LPCA is exactly the same as provided by the original LPCA algorithm.



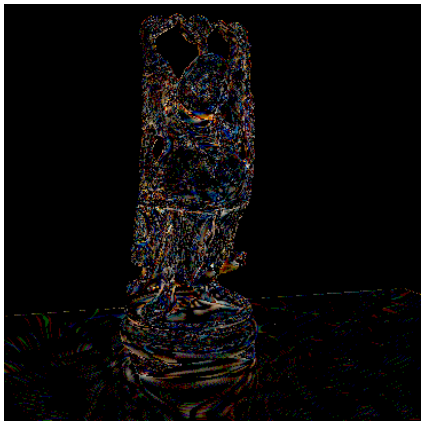
(a) Horse

(b) Dragon

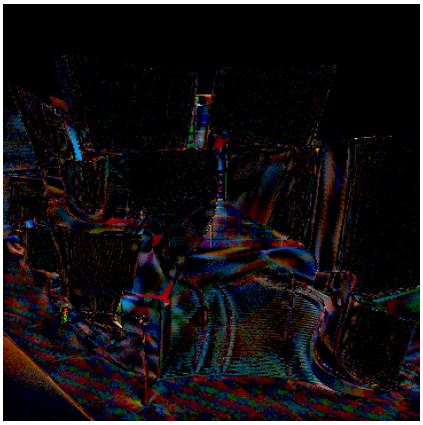
(c) Happy Buddha



(d) Walt Disney

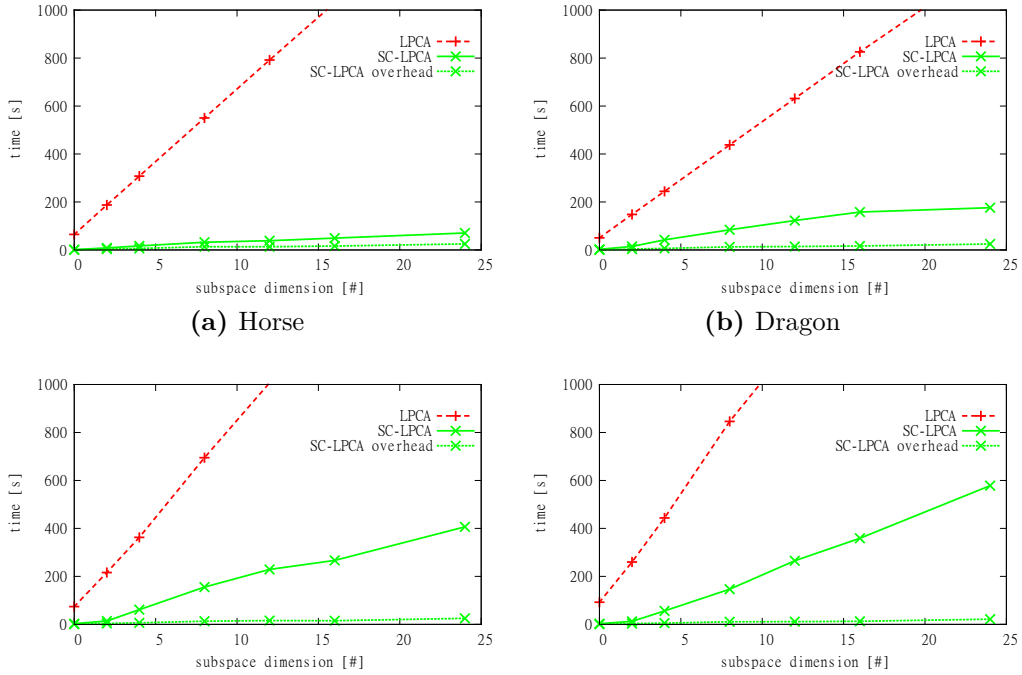


(e) Happy: 16× Error Image



(f) Disney: 16× Error Image

Figure 3.3 – Scenes used in our experiments. We tested our SortClusters LPCA (SC-LPCA) for compression of transfer matrices for these scenes. The models in 3.3a, 3.3b, and 3.3c are made of glossy materials represented using Phong's BRDF with exponent from 10 to 30. The model in 3.3d is represented using Ward's BRDF with the roughness of 0.1. Compared to LPCA, we achieve a 5× to 20× speed-up using SC-LPCA, while providing identical output. For the sake of completeness, Figure 3.3e and 3.3f show a 16× amplified difference between images rendered using the original and compressed transfer matrix.



(a) Horse **(b)** Dragon

(c) Happy Buddha **(d)** Walt Disney Hall

Figure 3.4 – Average computation time per one iteration of our SC-LPCA and original LPCA. Note that the curves for our SC-LPCA contain both the overhead and the time required for the classification itself. The overhead of the SC-LPCA consist of the computation of distance and permutation matrices **D** and **M**, respectively, and is very small even for a high number of PCA basis vectors.

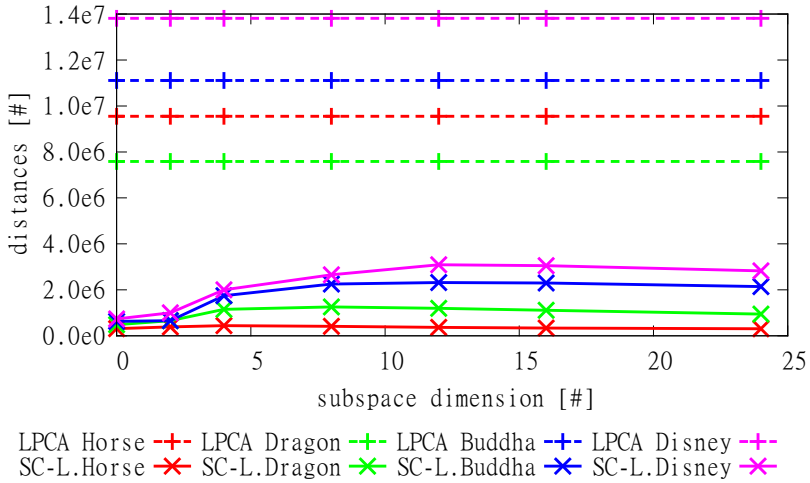


Figure 3.5 – Average number of distances evaluated by the compared algorithms per one iteration of the classification.

Scalability with the number of clusters. Breakdown of the classification times in dependence on the total number of clusters is shown in Figure 3.6. We use the Dragon scene for this comparison. The computation time of our SC-LPCA stays roughly constant with the number of clusters, meaning that our SC-LPCA is well suited for applications where clustering to a high number of clusters is required. The computation time of the classical LPCA, on the other hand, increases linearly with the number of clusters. Interestingly, the average time for an SC-LPCA iteration for subspace dimension of 24 *decreases* with the number of clusters, in spite of the fact that the SC-LPCA overhead (computation of the distance matrix **D**) increases quadratically.

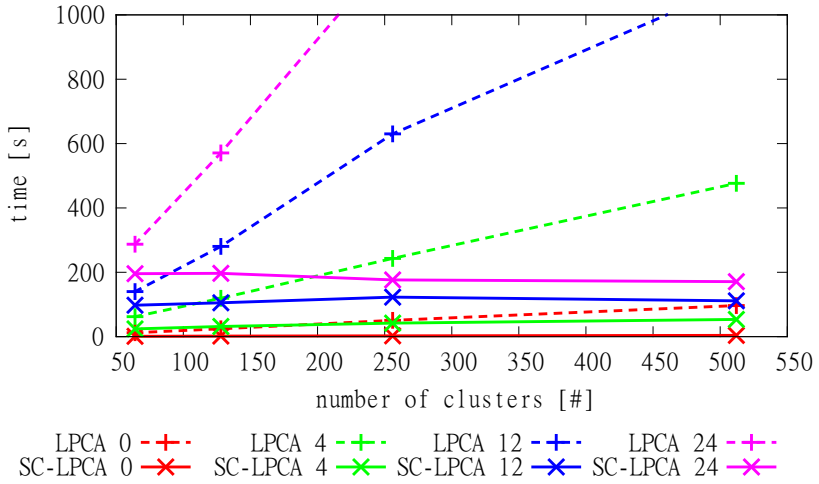
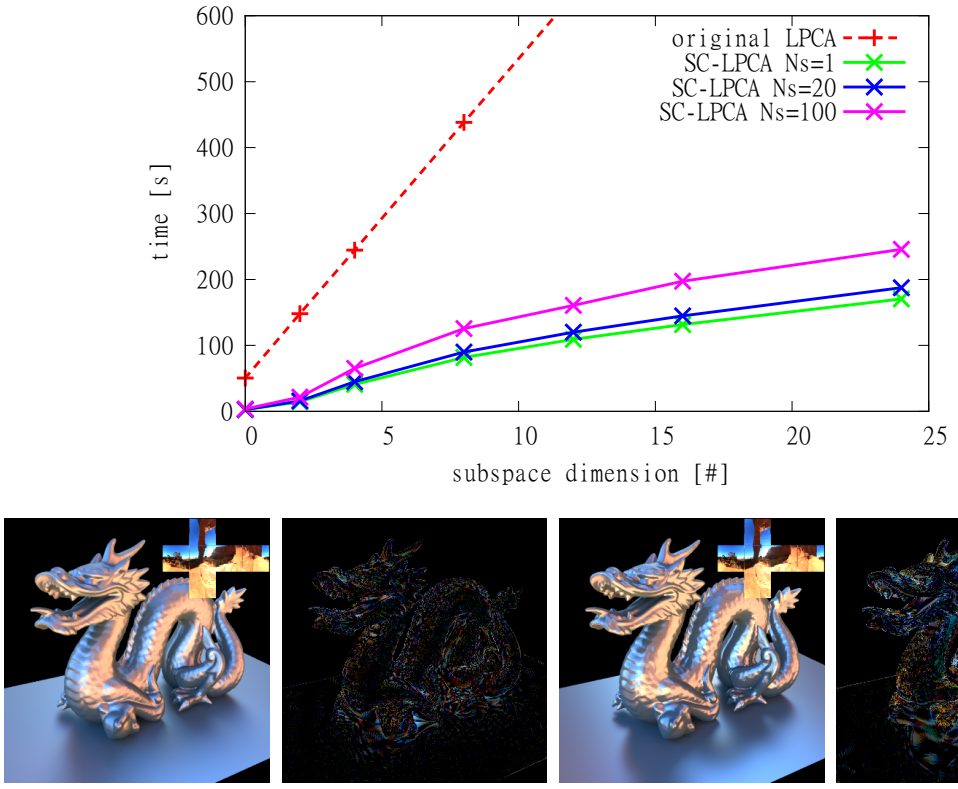


Figure 3.6 – Scalability with the number of clusters for the Dragon scene. The plots show the average time spent on one iteration of the classification for subspaces' dimensions of 0, 4, 12, and 24.

Scalability with the Phong's lobe exponent. Unlike the original LPCA, the performance of our SC-LPCA does depend on the data set itself. Figure 3.7 illustrates the scalability of SC-LPCA with different Phong's lobe exponent. We can see only a small decrease in performance for a high Phong's lobe exponent, meaning that our algorithms is applicable for a wide range of materials.

Latency and accuracy of initialization algorithms. Our previous results only focused on speeding-up LPCA without improving its accuracy. Here we investigate different strategies for initial centroid seeding and their impact on accuracy and performance. The results for the Dragon scene are summarized in Figure 3.8. We perform 50 iterations of the original SortMeans algorithm [Phi02] after clusters' initialization and plot averages from 5 different runs. The naïve



(a) $N_s = 20$ **(b)** 16 x Error **(c)** $N_s = 100$ **(d)** 16 x Error

Figure 3.7 – Scalability with the Phong's lobe exponent N_s for the Dragon scene. The top image shows the average time spent on one iteration of the classification. The bottom images show visual examples of the scene with dragon made of glossy material represented using Phong's BRDF with the lobe exponent of $N_s = 20$ and $N_s = 100$ in Figures 3.7a and 3.7c, respectively. Images in Figures 3.7b and 3.7d shows the corresponding 16× difference images between renderings of the scenes using original and LPCA-compressed light transfer matrices.

algorithm that select centroids purely at random [SHHS03] is prone to getting stuck in a local minimum. The k-means++ algorithm [AV07] provides a much lower approximation error at the cost of high initialization latency. While maintaining the approximation quality, our SortMeans++ algorithm decreases the start-up latency of the original k-means++ 6× to a value that is even less than the latency for purely random initialization (because SortMeans++ initialization produces a valid point-cluster classification).

In addition to producing a better clustering, a good initial seeding also improves the performance of our SC-LPCA in subsequent iterations. The total computation time spent by SC-LPCA over all iterations using our SortMeans++ for the initial seeding was 1700s with resulting error $\phi = 0.174$. On the other

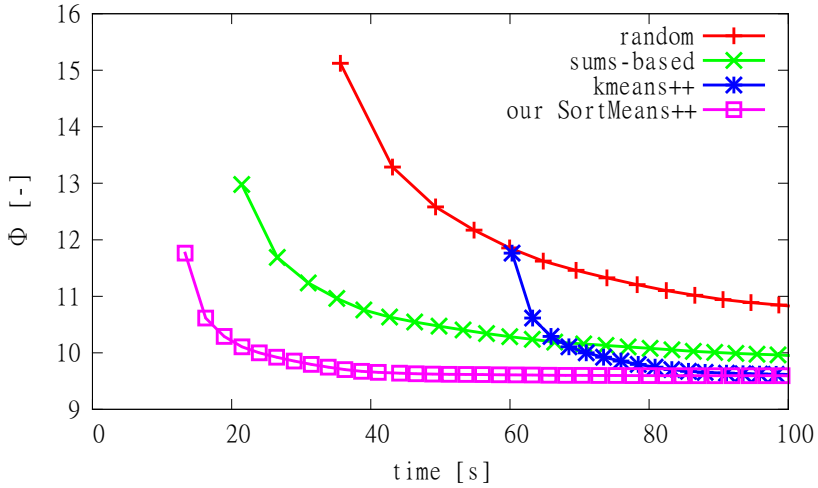


Figure 3.8 – Latency and accuracy of initialization algorithms. Note that our algorithm SortMeans++ maintains the quality of approximation provided by the original k-means++ while substantially decreasing the start-up latency of the point-cluster assignment.

hand, when we initialize the centroids randomly, the total computation time is 1780s and the error increases to $\phi = 0.224$.

Overall statistics. Table 3.1 lists the statistics about the pre-computation of the PRT matrices and the overall computation time required for their compression. Note that the most computationally demanding part is the classification of high dimensional data points to affine subspaces, which is the focus of our work. Evaluating PCA approximations within clusters is only a minor part of the total compression time for our data sets.

scene	vertices [#]	PRT [s]	Classification [s] LPCA	SC-LPCA	speedup	PCA [s]	ϕ [-]
Horse	67.6k	22.5	13 700	674	20.3	146	0.029
Dragon	57.5k	25.5	10 900	1700	6.38	155	0.174
Buddha	85.2k	31.6	16 700	3 020	5.55	170	0.316
Disney	106.3k	46.5	20 900	4 080	5.12	170	0.394

Table 3.1 – Summary results and timings for the example scenes. The columns list the total number of vertices in the scene, transfer matrix computation time (PRT) and the total classification time using original LPCA and our SC-LPCA algorithm. The rightmost two columns shows the time spent on the PCA approximation evaluation and the value of the objective function ϕ. Transfer matrices were computed on a Geforce GTX 580 GPU, while the Classification and PCA computation we performed on 4 CPU cores.

3.6.2 BTF compression

To investigate the efficiency of our algorithm on more complex data sets, we ran our algorithm on three measured BTFs (shown in Figure 3.9) from the BTF data provided by University of Bonn, http://btf.cs.uni-bonn.de. For each material the total 81×81 images were taken from different directions of the camera and light source. The images have a resolution of 256×256 pixels.

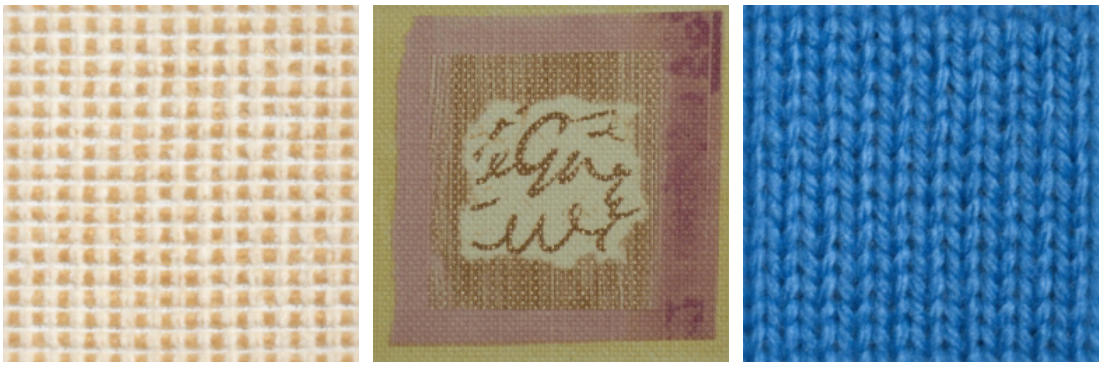


Figure 3.9 – BTF datasets used for testing our algorithm: proposte, wallpaper, and wool BTFs.

We run the LPCA in the apparent BRDF arrangement [FH09] (i.e. one data point corresponds to an image of the apparent BRDF). For this arrangement the LPCA provides an approximation of higher quality than for standard arrangement, while maintaining the same compression ratio [MMK03]. The total number of clusters and the maximum dimension of the affine subspaces was set to 32 and 8, respectively, in accordance with Müller [MMK03].

Breakdown of the classification times using the SC-LPCA and original LPCA for all tested materials is shown in Figure 3.10. For the proposte BTF the speed-up of the SC-LPCA is about 1.6. For the wallpaper and wool materials, which exhibit different reflectance characteristics in all sampled dimensions, the speed up of the SC-LPCA is about 1.2.

3.7 Conclusion

We present an accelerated and more accurate local PCA (LPCA) algorithm for compact approximation of large matrices. The improved performance is due to the significant reduction of point-cluster distance calculations in the classification stage of the algorithm. The accuracy is achieved through improved seeding of the initial cluster centroids. Our measurements on computer graphics data sets show a speed-up of 5 to 20 for radiance transfer matrices, though the speed-up is lower

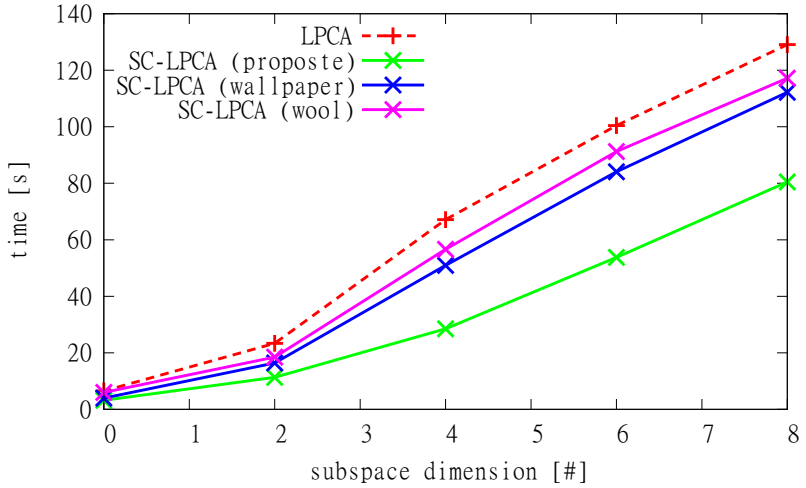


Figure 3.10 – Computation time for different BTFs. Computation time for the original LPCA algorithm is independent of the BTF material. On the other hand our algorithm performs better for simple materials with a regular spatial/angular structure.

for complex and high-dimensional BTF data sets. Devising new algorithms for accurate and efficient compression of these more complex data set is an exciting avenue for future work. In the shorter term, our fast LPCA could be combined with a GPU implementation of Huang and Ramamoorthi's algorithm [HR10] to obtain a near-interactive pre-computation and compression of PRT data sets. We want to investigate the efficiency of the proposed approach on other computer graphics data sets, such as local light transport [HPB06], and to achieve further speed-up by performing the clustering in wavelet domain.

Relighting of Animation Sequences 4

Contents

In this chapter we describe a work in progress focusing on the development of an interactive system for cinematic relighting of animation sequences with indirect lighting. We build on the idea of direct-to-indirect light transfer used in existing relighting systems [HPB06, KTHS06, LZT+08] expressing light propagation in a scene. We aim to precompute the direct-to-indirect light transfer not just in a static scene but also for an animated dynamic scene, where the movement of objects in the scene is known in advance. The chapter describes our relighting framework. However, as it has not been implemented yet we do not provide any results. This is left as a subject for future work.

4.1 Introduction

One of the important problems in computer cinematography is lighting design in a scene. Lighting design usually proceeds as follows: a lighting designer places

light sources in the scene, sets their parameters and renders the scene; then he adjusts the parameters of the light sources and renders the scene again, ... and so on until he obtains the desired lighting. Screenshots of a scene as illuminated by different lights is in Figure 4.1.

Relighting algorithms must meet several requirements. First, the relighting of a scene must be done interactively as an immediate response to the designer's change of light parameters. Second, the relit image must contain lighting effects that the final quality rendering of the scene will contain. The relighting algorithms that produce images that do not closely match the final result are useless for cinematic lighting design. Third, the relighting algorithms must be able to handle scenes with high geometric complexity, containing various kinds of reflectance functions, as well as flexible light source models including distant and local lights.

Relighting algorithms usually work in two phases: an offline, precomputation part and a real-time relighting phase. In the first phase visibility and partial shading in a scene are usually precomputed. The precomputed data are then used for fast rendering in run-time. An overview of relighting systems focusing on direct lighting with shadows can be found in [PVL+05]. Recently several methods have been proposed to handle also indirect lighting on-the-fly [HPB06, KTHS06, LZT+08] providing different trade-off between rendering accuracy, scene complexity, and view independency. However, to our knowledge there is no relighting system for animation sequences. Such a system would be extremely useful for lighting design in computer cinematography.

(a) Spot-light source **(b)** Spot and area-light sources

Figure 4.1 – Renderings of a scene with a hair ball relit using different configurations of light sources. Images were rendered by our implementation of cinematic relighting system [HPB06].

Our aim is to lift the restriction of existing relighting systems to static scenes, and extend them to animated scenes, where the movement of objects is known in advance. Using modern graphics hardware we can calculate direct lighting

including shadows in real-time. But much more operations must be performed to calculate accurate indirect lighting. In our work we focus on a relighting system delivering high-quality indirect lighting in the scene on-the-fly. To be able to achieve this goal, we use the idea of *direct-to-indirect* (DTI) light transfer as a linear mapping that transforms direct lighting in a scene to indirect lighting in this scene. If the scene is static, the DTI light transfer is constant over time. But we want to handle animated scenes with predefined object deformation and camera movement. In the case of dynamic scene the light transfer is not constant over time any more. To describe it over time we propose a tensor formulation of the problem: instead of pre-computing a light transfer matrix expressing the DTI light transfer in the static scene, we pre-compute a light transfer tensor expressing the light transfer in the animation. But it makes the precomputation even harder in memory space and computation time. A naïve pre-computation of the light transfer tensor is not feasible in practice. Fortunately, there is a high degree of coherence contained in the light transfer tensor. Changes in the scene often occur in localized parts in the dynamic scene while other parts remain mostly static, making DTI light transfer highly coherent. We exploit the coherence in our efficient algorithm for pre-computation and compression of the light transfer tensor.

4.2 Related Work

Precomputed Radiance Transfer (PRT). PRT refers to a group of methods that precompute light transfer in a scene, using it for interactive rendering of the scene with global illumination effects while allowing some changes in the scene. Many works are devoted to relighting of static scenes lit by an environment map [SKS02, SHHS03, LSSS04, NRH03, NRH04, HR10]. Common to these approaches is that they precompute the light transfer from the environment map in all the vertices in the scene and compress it subsequently. Ng et al. [NRH03] precompute the light transfer from an all-frequency environment map allowing either geometry or image relighting. In the first case the view point can arbitrarily change, assuming diffuse surfaces. In the second case the view point must remain static, allowing glossy surfaces. Liu et al. [LSSS04] and Wang et al. [WTL04] lift the restriction allowing both arbitrary view changes and glossy surfaces. Instead of baking the bi-directional reflectance distribution function (BRDF) into the light transfer, they extract the view-dependent component of the BRDF. To solve the same problem Ng et al. [NRH04] propose a triple wavelet product integral. But evaluation of the integral is still costly and cannot be performed in real-time.

The above methods deal with interactive relighting producing high-quality renderings. But the computation time needed for evaluating the light transfer accounting for self-occlusion as well as inter-reflection effects is time-consuming. Huang et al. [HR10] propose to speed up the precomputation through adaptive and sparse sampling of light transfer. They sample the light transfer densely in the directional domain (that corresponds to cube-map pixels) in so called *dense vertices* in the scene. Then they reconstruct the light transfer in other *sparse vertices* from the previously computed dense vertices. Our method builds on their idea of adaptive sampling of the light transfer, generalizing it to adaptive sampling of the transfer in all directional, spatial, and temporal domains simultaneously.

Local principal component analysis (LPCA) or non-linear wavelet approximation are commonly used to compress the precomputed light transfer. Some works [SHHS03, LSSS04, HR10] use the LPCA (under the name *clustered* PCA or CPCA) to exploit the spatial coherence of light transfer across vertices. On the other hand other works [NRH03, NRH04] compress light transfer in each vertex of the scene independently. In our algorithm, we combine both approaches in order to obtain a fast and efficient algorithm for compression of large datasets.

Other extension of PRT methods deal with relighting of articulated characters lit by an environment map [NSK+07, FPJY07]. As an input these methods take a large set of light transfer matrices precomputed for many different poses of an articulated character. Nowrouzezahrai et al. [NSK+07] use the precomputed light transfer matrices to find a linear mapping from the pose space to the space covering precomputed light transfer matrices for this character. Feng et al. [FPJY07] compress the precomputed light transfer matrices using LPCA. Then in run-time they find several nearby light transfer matrices close to the given pose and use them for fast interpolation. Precomputation of light transfer matrices for many different poses of the character and also running these algorithms on them, however, requires a huge amount of computation time. Moreover final rendering of the character is of lower accuracy due to strong compression. In contrast, we focus on high-quality rendering of complex dynamic scenes.

PRT for the support of localized light sources. The major limitation of the aforementioned methods is the restriction to an environment map. Many environment maps have to be sampled in many locations in a scene to be able to accurately simulate global illumination. But doing so prohibits interactive relighting. Anen et al. [AKDS04] propose a simple solution to support lighting from "mid-distant" lights. In addition to evaluating an environment map in some location $\mathbf{x}$ in the scene they also compute the gradients of the environment map. Then in run-time they extrapolate current environment map in the vicinity of $\mathbf{x}$ using the gradients of the environment map. But this technique does not

fully address local lighting, assuming that the visibility in the neighborhood of $\mathbf{x}$ remains constant.

A different solution for relighting of a scene under dynamically changing local lights has been proposed by Kristensen et al. [KAMJ05]. First, they distribute omni-directional light sources in the scene. Then they precompute the light transfer from these lights to mesh vertices, and finally they compress the light transfer using the LPCA. In run-time they find a small subset of omni-directional light sources that lie close to the user-defined local light, and use them to approximate the user-defined local light. Then they apply the precomputed light transfer obtaining indirect lighting on the mesh vertices and add it to direct lighting. The main limitations of the method are the restriction to omni-directional local light sources, and low accuracy because of high compression. Another disadvantage is that the method probably would not scale to complex scenes, since it works on the mesh vertices.

Hašan et al. [HPB06] lift the disadvantages of Kristensen et al.'s work but with the limitation of fixed view point. Instead of precomputing the light transfer from a set of omni-directional lights, they precompute light transfer from a large set of so called *gather samples* to another set of *view samples*. They distribute the gather samples in the whole scene while placing the view samples in positions directly visible from the camera. In real-time they perform the following steps: compute direct lighting on the gather samples, transform direct lighting on the gather samples to indirect lighting on the view samples using pre-computed DTI light transfer matrix, and add the resulting indirect lighting to the direct lighting on the view samples. Their relighting algorithm delivers high-quality renderings even in very complex scenes. We build on their relighting system and generalize it to the more complex problem of animation relighting.

Hierarchical approaches. Another group of techniques for interactive relighting of a scene as viewed from an arbitrary camera position and lit by local lights are based on the finite element method [KTHS06, LZT+08]. These techniques usually define basis functions over scene surfaces and use them to express light propagation in the scene. To speed up calculation of the light propagation, a hierarchy on the basis functions is built. Kontkanen et al. [KTHS06] define a 4D wavelet basis. Since they use mesh quads as the support for the wavelet basis, their approach is limited to simple scenes that are composed exclusively from large quads. To lift this restriction Lehtinen et al. [LZT+08] use a set of samples carefully distributed in the scene as a support for their basis functions. However, to produce a high-quality rendering a huge number of basis functions covering the whole scene must be defined. Having such a large number of basis functions, however, makes the computation too slow for interactive relighting. In contrast

to these methods, we focus on delivering high-quality rendering though at the price of knowing the object and camera position at each frame of the animation sequence.

Goal. The major limitation of the cinematic relighting system proposed by Hašan et al. [HPB06] is the restriction to a static scene as observed from a fixed view point. Lighting design in an animation sequence with dynamic scene is not ideal with such a system: The lighting designer must set lighting design just in one frame; then he must switch to another frame, check if the lighting is correct and tweak it alternatively. These steps must be performed repeatedly often checking the lighting design in one frame several times until the desired lighting in the whole animation sequence is obtained. Such a procedure, however, precludes rapid lighting of animated sequences. Our goal is to develop an interactive system to better support lighting design in animations by allowing to play back the relit animation sequence.

4.3 Overview of Our Contribution

We propose an algorithm for cinematic relighting in animated sequences. To make our relighting system useful for lighting design in computer cinematography we must deliver high-quality rendering of the animation sequence with indirect lighting. But high-quality indirect lighting is time-consuming and cannot be rendered in real-time even on the latest graphic hardware. To achieve this goal, our relighting system is split into two parts: an off-line part for pre-computing the direct-to-indirect (DTI) light transfer tensor, and a run-time part for high-quality rendering of the animation sequence with indirect lighting.

Run-time phase. The run-time part of our relighting system uses the pre-computed DTI light transfer to interactively render global illumination in the predefined animation sequence. The run-time part can be summarized into the following steps, that are described in more detail in Section 4.6.

1. Calculate direct lighting at frame t on a set of points (*gather samples*) distributed uniformly in the scene.
2. Use the precomputed light transfer tensor to transform the direct lighting on gather samples to indirect lighting at points (*view samples*) visible through camera at frame t.
3. Calculate direct lighting on the view samples and add it to indirect lighting, obtaining final rendering of the scene at frame t.

4. Shift to the next frame, i.e. $t \leftarrow t + 1$, and repeat all steps until the end of the animation sequence is reached.

Pre-computation phase. In the off-line part we pre-compute the light transfer tensor. Since the tensor is huge, containing several Tera (i.e. 10^{12}) elements, a problem arises: how to make its pre-computation feasible in terms of memory space and computation time. But as we mentioned above the light transfer tensor contains a high degree of coherence that can be exploited to make the evaluation practical. We use the following strategy to evaluate the light transfer tensor in a compressed form while keeping the memory requirements tractable. See Figure 4.2 for a conceptual overview of our off-line algorithm.

1. Explore the structure of the light transfer tensor; elements of the light transfer tensor in which the light transfer is likely to change rapidly are sampled more densely than other elements.

2. Run a modified version of the local principal component analysis (LPCA) to find linear subspaces that closely approximate light transfer vectors of the parts of the tensor explored so far.

3. Reconstruct light transfer in other unexplored parts of the light transfer tensor and approximate them in the previously computed linear subspaces.

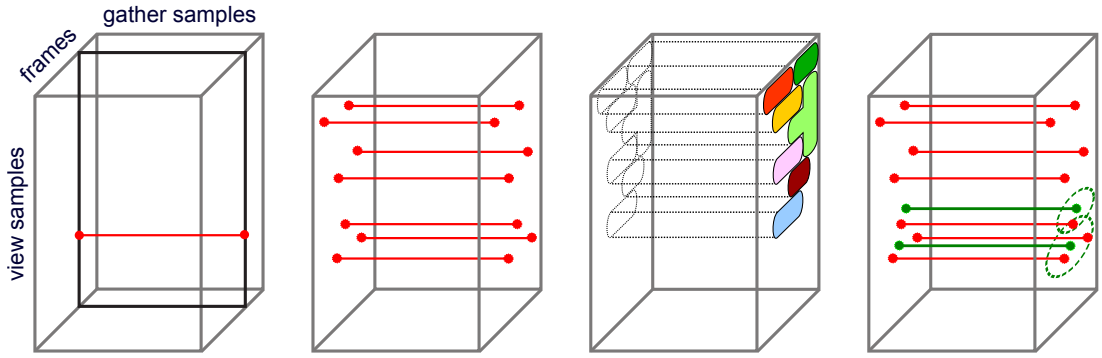
Pre-computation of the light transfer tensor is an essential part of our relighting system. In Section 4.4 we formally define the light transfer tensor and show how we explore its raw structure. In Section 4.5 we present a rapid modified LPCA called *Wavelet LPCA*. Section 4.5.3 gives more details about the reconstruction of the light transfer tensor in unexplored parts.

4.4 Tensor Exploration

4.4.1 Definition of the Light Transfer Tensor

We aim to compute indirect lighting for an animation sequence allowing for dynamic scenes. To achieve this goal we pre-compute DTI light transfer over the animation sequence. We use two sets of samples to express the DTI light transfer over the animation sequence: a set of *gather samples* and a set of *view samples*.

To define the gather samples we distribute d samples on the whole scene surface, and associate them with the object surfaces they lie on. When the objects change (due to translation and/or deformation) the positions, normals, and areas of the associated gather samples may change as well. Having these



(a) Definition **(b)** Exploration **(c)** Compression **(d)** Refinement

Figure 4.2 – Conceptual overview of our offline algorithm. (a) We use a light transfer tensor to describe direct-to-indirect light transfer from one set of samples to another set of samples over multiple frames of an animation sequence. The solid line depicts the contributions of all gather samples at some frame to one view sample at the same frame. (b) We start by exploring structure of the light transfer. (c) Then we run a modified version of local principal component analysis (LPCA) to find linear subspaces that fit the light transfer tensor. (d) Finally we refine the structure of the light transfer tensor and approximate it in linear subspaces found in the previous step.

samples depending on time, we pick d gather samples at each time $t \in 1 \dots f$ of the animation sequence consisting of f frames. Let g_{jt} be the j-th gather sample at time t and let G be the set of all the gather samples

$$G = \{g_{jt} \mid j \in 1 \dots d,\ t \in 1 \dots f\}\,.$$

The view samples correspond to the points visible to the camera through image pixels. Let v_{it} be the i-th view sample at t and let V be the set of all the view samples

$$V = \{v_{it} \mid i \in 1 \dots p,\ t \in 1 \dots f\}\,,$$

where p is the number of pixels captured by the camera.

Having these sets we can formalize DTI light transfer over the animation sequence as a large $|p| \times |d| \times f$ tensor of light interaction between v_{it} and g_{jt}, $i \in 1 \dots p,\ j \in 1 \dots d,\ t \in 1 \dots f$. Each element of the light transfer tensor expresses DTI light transfer from one gather sample to one view sample in one frame of the animation sequence. See Figure 4.3 for a visual illustration of the relationship between v_{it} and g_{jt}.

The DTI light transfer from g_{jt} to v_{it}, i.e. the tensor element at index (i, j, t), is defined as:

$$A(g_{jt})\ f_r(v_{it} \to g_{jt})\ G(v_{it}, g_{jt})\ V(v_{it}, g_{jt}), \tag{4.1}$$

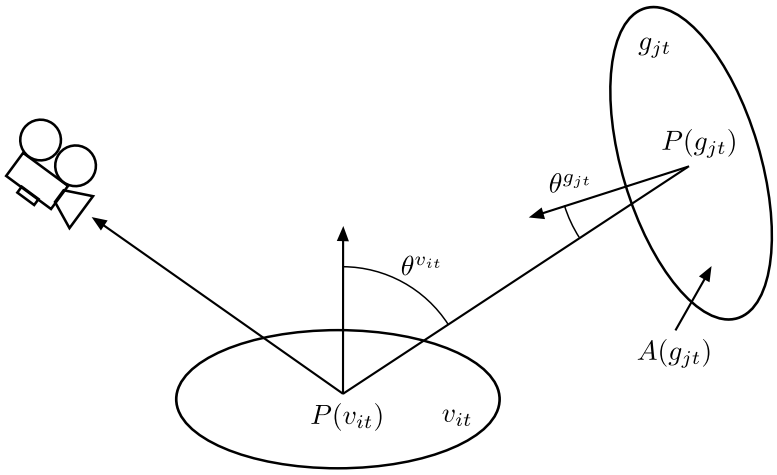


Figure 4.3 – Geometry of direct-to-indirect (DTI) light transfer from a gather sample g_{jt} to a view sample v_{it}.

where $A(g_{jt})$ is the area of the sample g_{jt}, $f_r(v_{it} \rightarrow g_{jt})$ is the bi-directional reflectance distribution function (BRDF) at v_{it} in direction to the camera and to g_{jt}, and $G(v_{it}, g_{jt})$ and $V(v_{it}, g_{jt})$ are the geometry term and visibility between v_{it} and g_{jt}, respectively. The geometry term is defined as:

$$G(v_{it}, g_{jt}) = \frac{\cos\theta^{v_{it}} \cos\theta^{g_{jt}}}{\|P(v_{it}) - P(g_{jt})\|^2},$$

where $P(\,\cdot\,)$ is the 3D position of the sample, $\theta^{v_{it}}$ and $\theta^{g_{jt}}$ are the angles between the normal at v_{it} and g_{jt}, respectively, and the vector $P(g_{jt}) - P(v_{it})$. $V(v_{it}, g_{jt})$ is equal 1 when the g_{jt} is visible from the v_{it} and 0 otherwise.

The usual image resolution p and the dimension d used by Hašan et al. [HPB06] is 300k (i.e. 300×10^3) pixels and 64k samples, respectively. For an animated sequence of one hundred frames the light transfer tensor is of 1.9 Tera (i.e. 1.9×10^{12}) elements. A naïve approach that evaluates the light transfer tensor in each element is not feasible in terms of computation time and also memory space. In the following sections we show how to evaluate and compress the light transfer tensor more efficiently.

4.4.2 Exploring the Structure of the Light Transfer Tensor

To save a large amount of computation we subsample light transfer in carefully selected view samples of the light transfer tensor. The question is where to sample (i.e. at which view sample we should calculate light transfer) and how densely we should sample. Ideally we would like to sample proportionally to the rate of change of light transfer in the animated scene: those regions of the light transfer tensor where the light transfer changes rapidly would be sampled more densely

than other regions. But we have no a priori information about the light transfer at the beginning of the precomputation. To address this problem we adopt the iterative matrix sampling technique proposed by Huang and Ramamoorthi [HR10] and generalize it to efficient sampling of the light transfer tensor. In the following we describe our generalized sampling technique.

The iterative sampling technique that allows us to explore the structure of the light transfer tensor works as follows. We start by assigning to each transfer point $v_{it} \in V$ a uniform sampling weight (probability). Then we pick a small subset of view samples using weighted sampling and evaluate light transfer in the selected view samples. The light transfer in view sample v_{it} is a d-dimensional vector where j-th element expresses the contribution of gather sample $g_{jt} \in G$ to v_{it}. We denote the light transfer vector in v_{it}, i.e. the row of light transfer tensor, by $\mathbf{x}_{it}$.

So far we have only a very crude approximation of the light transfer tensor. To be able to adaptively explore the regions of the light transfer tensor where the light transfer shows a high change rate, we need to update the sampling weights of the remaining, unsampled view samples. To update the sampling weights, we use the local dimensionality of light transfer, which has been shown to be proportional to the rate of change of the light transfer in the scene [HR10]. So we estimate local dimensionality at each of the remaining view samples and assign them new sampling weight proportional to the dimensionality estimate. After that we sample the next batch of view samples, and so on until the desired number of iterations is reached. How to determine the number of view samples picked in each iteration and the total number of the iterations are subjects for future work. The pseudo-code for this iterative adaptive sampling of view samples is shown in Algorithm 3.

To compute the dimensionality of light transfer around a view sample $v_{it} \in V$ we pick a small set S of nearby view samples (sampled so far) which lie "close" to v_{it}. In the case of a static scene the Euclidean distance of view samples visible in the scene is used to measure the closeness of the view samples. But in our case we need to take into account also the closeness of our view samples in time. Let $v_{it_a}, v_{jt_b} \in V$ be two view samples. We propose to define the distance $d(v_{it_a}, v_{jt_b})$ as:

$$d(v_{it_a}, v_{jt_b}) = \sqrt{w_1^2 \|P(v_{it_a}) - P(v_{jt_b})\|^2 + w_2^2 (t_a - t_b)^2}, \tag{4.2}$$

where w_1 and w_2 are user-defined weights. In a future work we plan to set these weights automatically. We suggest to take the inverse of scene size for w_1 and the inverse of the total number of frames for w_2.

Having the set S of view samples which are close to v_{it}, we can estimate the local dimensionality of light transfer around v_{it}. We build a $|S| \times d$ local light transfer matrix with i-th row equal to the light transfer vector of the i-

Algorithm 3 Exploring structure of the light transfer tensor

input : V set of view samples
Scene configuration
output: Approximation of the light transfer tensor; i.e. a set of $\mathbf{x}_{it}$ in carefully chosen $v_{it} \in V$

$\forall v_{it} \in V,\ w(v_{it}) \leftarrow$ uniform sampling weight;
$D \leftarrow$ empty set;

for *several iterations* **do**
  for $i \leftarrow 1$ ***to*** *user-defined number* **do**
    $v_{it} \leftarrow$ weighted sampling from $V \setminus D$;
    $\mathbf{x}_{it} \leftarrow$ light transfer vector in v_{it};
    $D \leftarrow D + v_{it}$;
  end
  foreach v_{it} ***in*** $V \setminus D$ **do**
    $T \leftarrow$ nearby samples from D;
    $w(v_{it}) \leftarrow$ dimensionality of T;
  end
end

th view sample from S. Then we compute the numerical rank of this local light transfer matrix, which is equal to the local dimensionality of light transfer around v_{it} [MSRB07, NBB04]. So in the regions in the scene where the light transfer changes very smoothly, the local light transfer matrix will be of small rank, i.e. the dimensionality of light transfer around v_{it} will be small. Since we set the sampling weights for v_{it} equal to the local dimensionality, the probability of picking the v_{it} will be smaller in that case. Figure 4.4 shows an example of a scene with the rank of the local light transport shown in false color.

4.4.3 Computation of the Light Transfer in View Samples

For every view sample $v_{it} \in V$ selected by the iterative sampling algorithm described above, we need to evaluate the corresponding light transfer vector $\mathbf{x}_{it}$. A naïve approach for doing this would iterate over the gather samples, and for each gather sample $g_{jt} \in G$, $j \in 1 \ldots d$, the transfer from g_{jt} to v_{it} would be evaluated. Such a brute-force approach, however, is very costly.

To speed up the evaluation of the transfer vector $\mathbf{x}_{it}$ in the view sample v_{it}, we adopt Hašan et. al.'s [HPB06] N-body approach. For each frame t we build a perfectly balanced quad-tree on the positions of the gather samples $g_{jt} \in G$. When evaluating the light transfer at v_{it}, we select the quad-tree built on the

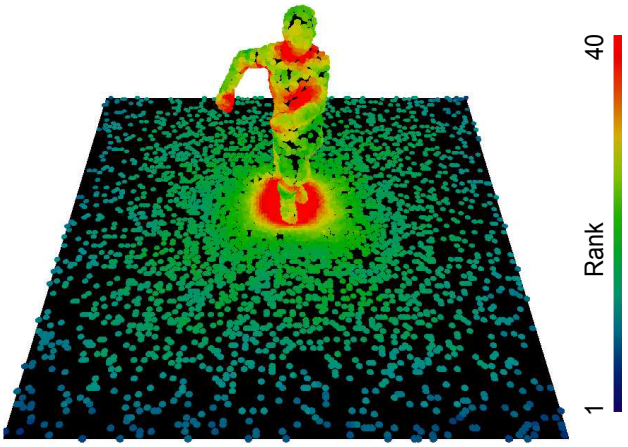


Figure 4.4 – Local dimensionality of light transfer around a $3D$ point in the scene is defined to be proportional to the rank of the local transfer matrix, whose rows are the transfer vectors of several nearby view samples. Colored dots show the points in the scene where the rank of the local transfer matrices was evaluated. We can see that the rank is of highest values around the neck and on the ground beneath the feet, where the light transport varies a lot. In contrast the rank is of low value far away from the runner, since the light transport is smooth here. Image courtesy of Huang and Ramamoorthi [HR10].

gather samples at t, and traverse it in a depth-first manner. We use a solid angle heuristic to decide whether to stop the traversal in the current quadrant or continue to the children: If a quadrant is small and/or far enough, we randomly select one gather sample from the quadrant and calculate its contribution to v_{it}. We approximate contributions of all the other gather samples from the quadrant by the same value.

Alternatively, the light transfer vector $\mathbf{x}_{it}$ at the view sample v_{it} could also be estimated using a Monte Carlo technique described in [OKP+08]. A number of rays can be traced from the position of v_{it} through the hemisphere above v_{it}. For each ray an intersection with the scene is computed and the nearest gather sample $g_{jt} \in G$ is found, using a kd-tree to speed-up the search. Finally, contribution from g_{jt} to v_{it} is computed and adjusted by the inverse of the probability density of sampling that direction.

The transfer vector $\mathbf{x}_{it}$ estimated by the Monte Carlo method provides better image quality near corners and other places in the scene where the density of gather samples is not sufficient. In those places the N-body approach is not suitable and produces energy losses giving rise to local image darkening. Nonetheless, the Monte Carlo method will produce noisy and sparse transfer vectors $\mathbf{x}_{it}$ with many zero elements even if the real contribution of the corresponding elements should be non zero. The N-body approach, on the other hand, provides transfer vectors $\mathbf{x}_{it}$ with lower variance across the individual elements. Such a vector is

more amenable to compression which we perform in the next stage and that is why we decided to adopt the N-body approach in our work.

4.5 Wavelet-LPCA

In the previous section we have described how to calculate a raw approximation of the light transfer tensor. In order to reduce the memory requirements and to be able to perform transformation using these light transfer vectors at run-time, we need to compress them. Two different methods have been used to compress the light transfer vectors in the previous work: non-linear wavelet approximation and compression based on dimensionality reduction through local principal component analysis (LPCA).

In the case of wavelets each light transfer vector is approximated as a sparse vector of wavelet coefficients. Using only the wavelet compression to compress each transfer vector independently, however, would not be sufficient in our case because of the sheer number of individual transfer vectors (remember that we are dealing with an animation sequence). Multi-dimensional wavelet compression across light transfer vectors cannot be used because that would require costly decompression at run-time.

Unlike the wavelet compression that compresses each light transfer vector independently, the local principal component analysis (LPCA) explores coherence across different light transfer vectors. Running the LPCA on the original light transfer vectors would, however, be infeasible, since the calculation time of the LPCA is proportional to the dimension of the light transfer vectors, which in our case is about 64k.

Any one of the two compression techniques (wavelet approximation and LPCA compression) does not seem sufficient for our purpose. But their respective advantages and drawbacks are complementary. If we could shorten the length of the light transfer vectors, we could use the LPCA. To make the light transfer vectors shorter we exploit their sparsity when approximated in non-linear wavelet basis. Then we run the LPCA directly on the sparse wavelet images of the light transfer vectors exploiting coherence across these wavelet images. This, however, requires to reformulate the LPCA to be applicable to sparse wavelet vectors.

In the following we show how to perform the LPCA on sparse wavelet images of light transfer vectors. The goal of the LPCA is to choose several clusters represented by low-dimensional affine subspaces that approximate the light transfer vectors minimizing some error measure. The LPCA iteratively perform two steps:

- Classify all the light transfer vectors to the nearest clusters.
- Update of the clusters' affine subspaces.

For more details about the LPCA, see the Chapter 3. In the following we develop our *Wavelet-LPCA* (WLPCA) algorithm as an efficient LPCA algorithm for compression of high-dimensional data. Our WLPCA will be much faster than running the original LPCA performed on the long, dense light transfer vectors, without loosing much of the accuracy of the original LPCA.

4.5.1 Classification to Nearest Affine Subspaces

4.5.1.1 Distance Calculation in the Wavelet Domain

When classifying a light transfer vector to the nearest cluster, distances of the transfer vector to the clusters must be evaluated. Let V^d be the linear space of all d-dimensional light transfer vectors. To calculate the distance of any light transfer vector $\mathbf{x} \in V^d$ to a cluster represented by an affine subspace $\mathbf{a}$ we need to evaluate Equation 3.1:

$$d(\mathbf{x}, \mathbf{a}) = \|(\mathbf{x} - \mu(\mathbf{a})) - \mathbf{x}_s\|,$$

where $\mu(\mathbf{a})$ is the origin of $\mathbf{a}$, $\mathrm{dir}(\mathbf{a})$ is its basis, and $\mathbf{x}_s$ is the orthogonal projection of $(\mathbf{x} - \mu(\mathbf{a}))$ onto $\mathbf{a}$.

Let us switch the representation of the light transfer vectors and of the affine subspaces to the wavelet domain. Let $\mathbf{x}^w$ be the wavelet image of $\mathbf{x}$ and $\mathbf{a}^w$ be the affine subspace with the origin and basis vectors represented in the same wavelet basis. Formally we can write $\mathbf{x}^w = \mathbf{W}\mathbf{x}$ for a transfer vector $\mathbf{x}$, $\mu(\mathbf{a}^w) = \mathbf{W}\mu(\mathbf{a})$ for origin and $\mathrm{dir}_i(\mathbf{a}^w) = \mathbf{W}\mathrm{dir}_i(\mathbf{a})$ for basis vectors of the affine subspace $\mathbf{a}$. The $d \times d$ matrix $\mathbf{W}$ transforms any vector from V^d into wavelet basis [Mal08]. The question is what does the distance $d(\mathbf{x}^w, \mathbf{a}^w)$. Let us show that $d(\mathbf{x}, \mathbf{a}) = d(\mathbf{x}^w, \mathbf{a}^w)$. For $d(\mathbf{x}^w, \mathbf{a}^w)$ we have:

$$\begin{aligned} d(\mathbf{x}^w, \mathbf{a}^w) &= \| (\mathbf{x}^w - \mu(\mathbf{a}^w)) - \mathbf{x}_s^w \| \\ &= \|\mathbf{W} ((\mathbf{x} - \mu(\mathbf{a})) - \mathbf{x}_s) \| \\ &\overset{(1)}{=} \|\mathbf{W}\| \cdot \| ((\mathbf{x} - \mu(\mathbf{a})) - \mathbf{x}_s) \| \\ &= \| (\mathbf{x} - \mu(\mathbf{a})) - \mathbf{x}_s \| \\ &= d(\mathbf{x}, \mathbf{a}), \end{aligned}$$

where $\mathbf{x}_s^w$ is the orthogonal projection of $\mathbf{x}^w$ onto $\mathbf{a}^w$. In (1) we use the orthonormality of $\mathbf{W}$. For $\mathbf{x}_s^w$ it is also easy to see that $\mathbf{x}_s^w = \mathbf{W}\mathbf{x}_s$:

$$\begin{aligned} \mathbf{x}_s^w &= \sum_{i=1}^{l} \langle \mathbf{x}^w - \mu(\mathbf{a}^w), \mathrm{dir}_i(\mathbf{a}^w) \rangle \, \mathrm{dir}_i(\mathbf{a}^w) \\ &= \sum_{i=1}^{l} \langle \mathbf{x} - \mu(\mathbf{a}), \mathrm{dir}_i(\mathbf{a}) \rangle \, \mathbf{W}\mathrm{dir}_i(\mathbf{a}) = \mathbf{W}\mathbf{x}_s, \end{aligned}$$

since for any vector $\mathbf{x}^w, \mathbf{y}^w \in V^d$ we have

$$\langle \mathbf{x}^w, \mathbf{y}^w \rangle = (\mathbf{x}^w)^T \mathbf{y}^w = \mathbf{x}^T \mathbf{W}^T \mathbf{W} \mathbf{y} = \mathbf{x}^T \mathbf{y} = \langle \mathbf{x}, \mathbf{y} \rangle .$$

So indeed $d(\mathbf{x}, \mathbf{a}) = d(\mathbf{x}^w, \mathbf{a}^w)$. Thanks to this distance invariance property, $\mathbf{x}^w$ will be closest to $\mathbf{a}^w$ if and only if $\mathbf{x}$ would be closest to $\mathbf{a}$. Applying this statement to all the previously evaluated light transfer vectors we get the same vector-cluster assignment made in the classification stage of the LPCA regardless of whether we are working with the original light transfer vectors or with their wavelet images.

4.5.1.2 Distance Calculation for Sparse Vectors

Let $\tilde{\mathbf{x}}^w$ be a compact sparse approximation of $\mathbf{x}^w$, keeping just a few highest wavelet coefficients from $\mathbf{x}^w$. Calculation of $d(\tilde{\mathbf{x}}^w, \mathbf{a}^w)$ using Equation 3.1, however, is not efficient because it cannot allow to exploit sparsity in $\tilde{\mathbf{x}}^w$. In order to speed up the calculation of $d(\tilde{\mathbf{x}}^w, \mathbf{a}^w)$ exploiting the compact, sparse representation of $\tilde{\mathbf{x}}^w$, we reformulate the distance formula into a more suitable form.

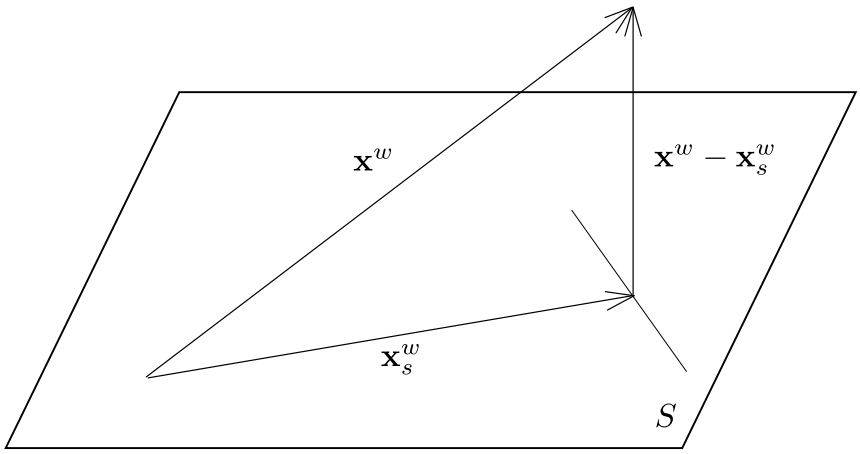


Figure 4.5 – Orthogonal projection of $\mathbf{x}^w$ onto subspace S. Vector $\mathbf{x}^w - \mathbf{x}_s^w$ is perpendicular to any line in subspace S. Specifically the triangle formed by vectors $\mathbf{x}^w$, $\mathbf{x}_s^w$, and $\mathbf{x}^w - \mathbf{x}_s^w$ is right-angled with the right angle at the end of $\mathbf{x}_s^w$.

Let $S \subset V^d$ be the subspace generated by $\mathrm{dir}_i(\mathbf{a}^w)$ of an affine subspace $\mathbf{a}^w$ and let $S^\perp \subset V^d$ be the orthogonal complement of S in V^d. For any $\mathbf{x}^w \neq \mathbf{0}$ the orthogonal projection of $\mathbf{x}^w$ on S is $\mathbf{x}_s^w \in S$, and therefore $\mathbf{x}^w - \mathbf{x}_s^w \in S^\perp$, see Figure 4.5. Then according to the Pythagorean theorem we have:

$$d^2(\mathbf{x}^w, \mathbf{a}^w) = \| (\mathbf{x}^w - \mu(\mathbf{a}^w)) - \mathbf{x}_s^w \|^2 = \|\mathbf{x}^w - \mu(\tilde{\mathbf{a}}^w)\|^2 - \|\mathbf{x}_s^w\|^2.$$

The two terms on the right hand side can be calculated as:

$$\|\mathbf{x}^w - \mu(\mathbf{a}^w)\|^2 = \|\mathbf{x}^w\|^2 - 2(\mathbf{x}^w)^T \mu(\mathbf{a}^w) + \|\mu(\mathbf{a}^w)\|^2 \tag{4.3}$$

and

$$\begin{aligned} \|\mathbf{x}_s^w\|^2 &= \left\| \sum_{i=1}^{l} \langle \mathbf{x}^w - \mu(\mathbf{a}^w), \mathrm{dir}_i(\mathbf{a}^w) \rangle \, \mathrm{dir}_i(\mathbf{a}^w) \right\|^2 \\ &= \|\mathbf{V}^T \mathbf{x}^w - \mathbf{V}^T \mu(\mathbf{a}^w)\|^2, \end{aligned} \tag{4.4}$$

where $\mathbf{V}$ is a $d \times l$ matrix with $\mathrm{dir}_i(\mathbf{a}^w)$ as i-th column. Remember that d is dimension of light transfer vectors and l is dimension of affine subspaces where we approximate the light transfer vectors in.

Distance calculation $d(\tilde{\mathbf{x}}^w, \mathbf{a}^w)$ in this form is more convenient for sparse $\tilde{\mathbf{x}}^w$ than Equation 3.1. Let us have a look why. When evaluating term $\|\tilde{\mathbf{x}}^w - \mu(\mathbf{a}^w)\|^2$ for sparse $\tilde{\mathbf{x}}^w$ in Equation 4.3, $\tilde{\mathbf{x}}^w$ appears in a dot product with dense vector $\mu(\mathbf{a}^w)$, $2(\tilde{\mathbf{x}}^w)^T \mu(\mathbf{a}^w)$. This dot product can be efficiently evaluated by accumulating just $|\tilde{\mathbf{x}}^w|$ wavelet coefficients. Similarly in Equation 4.4, $\tilde{\mathbf{x}}^w$ appears in a multiplication with dense matrix, $(\tilde{\mathbf{x}}^w)^T \mathbf{V}$ which is also very efficient to calculate. Other terms on the right sides of the equations can be precomputed as explained hereafter. Since square norms $\|\tilde{\mathbf{x}}^w\|^2$ do not change during the course of the computation it can be precomputed and passed to WLPCA. In the case of terms $\|\mu(\mathbf{a}^w)\|^2$ and $\mathbf{V}^T \mu(\mathbf{a}^w)$ they are constant for $\mathbf{a}^w$, so we can precompute them once after $\mathbf{a}^w$ is updated. Expecting that a high number of light transfer vectors will be assigned to $\mathbf{a}^w$, the overhead of pre-computation of these terms should be easily amortized. The pseudo-code for classification stage of our WLPCA is given in Algorithm 4.

Acceleration of the distance computation proposed in this section complements the acceleration proposed in Chapter 3. In that chapter, we eliminate distance calculation that provably cannot change the current vector-cluster assignment: If one knows that the distance of a light transfer vector to an affine subspace cannot be less than the current minimum distance found so far, calculation of their true distance can be avoided. On the other hand, the technique described above in this section focuses on quick calculation of the distance itself.

4.5.2 Update of Affine Subspaces

The second important step in LPCA, after the data vector classification, is the update of the affine subspaces in the individual clusters. Let us start by explaining how to calculate the affine subspace that fits a set of light transfer vectors, and then let us switch to the wavelets.

Let S be the set of light transfer vectors $\mathbf{x}_i$, $i = 1 \ldots m$ assigned to a cluster represented by an l-dimensional affine subspace $\mathbf{a}$. To update $\mathbf{a}$ we need to find a new origin $\mu(\mathbf{a})$ and basis vectors $\mathrm{dir}_i(\mathbf{a})$ that better fit S. The new origin is simply the mean of the light transfer vectors from S. The question is how to find

Algorithm 4 Classification stage of our Wavelet-LPCA

input : X set of sparse light transfer vectors $\tilde{\mathbf{x}}^w$ and their norms $\|\tilde{\mathbf{x}}^w\|^2$
A vector of k affine subspaces to be updated
R vector of $\|\mu(A[i])\|^2, i \in 1 \dots k$
S $l \times k$ matrix formed by $\mathbf{V}_i^T \mu(A[i])$ as i-th column; $\mathbf{V}_i^T$ is $d \times l$ matrix formed by $\mathrm{dir}_j(A[i])^T \mu(A[i])$ as j-th column

output: Assignment of $\tilde{\mathbf{x}}^w \in X$ to A

foreach $\tilde{\mathbf{x}}^w \in X$ **do**
  $d_{\min} \leftarrow \infty$;
  $i_{\min} \leftarrow 1$;
  for $i \leftarrow 1$ ***to*** k **do**
    $\mu \leftarrow \mu(A[i])$;
    $\mathbf{V} \leftarrow d \times l$ matrix formed by $\mathrm{dir}_j(A[i])$ as j-th column;
    $y \leftarrow \|\tilde{\mathbf{x}}^w\|^2 - 2(\tilde{\mathbf{x}}^w)^T \mu + R[i]$;
    $z \leftarrow \|\mathbf{V}^T \tilde{\mathbf{x}}^w - S[i\text{−th column}]\|^2$;
    $d \leftarrow y - z$;
    if $(d < d_{\min})$ **then**
      $d_{\min} \leftarrow d$;
      $i_{\min} \leftarrow i$;
    end
  end
  Assign $\tilde{\mathbf{x}}^w$ to $A[i_{\min}]$;
end

the new basis vectors. Let $\mathbf{A}_0$ be the $m \times d$ matrix with i-th row equal to $\mathbf{x}_i$, $\mathbf{1}^m$ be the m-dimensional vector of ones, and put $\mathbf{A} = \mathbf{A}_0 - \mathbf{1}^m \mu(\mathbf{a})$ (i.e. we subtract the mean from the rows of $\mathbf{A}_0$). Computing the *singular value decomposition* (SVD) of $\mathbf{A}$ we obtain $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are the left and right orthogonal rotation matrices, respectively, and $\mathbf{D}$ is a diagonal matrix of singular values arranged in descending order. Having the SVD of $\mathbf{A}$ the new basis consist of the first l rows of $\mathbf{V}$ that are called the right singular vectors.

Now let us switch to the wavelets. Let T be the set of the corresponding light transfer vectors from S in the wavelet basis and let $\mathbf{a}^w$ be the affine subspace calculated as above but using wavelet images of the light transfer vectors from T. We would like to have $\mu(\mathbf{a}^w) = \mathbf{W}\mu(\mathbf{a})$ and $\mathrm{dir}_i(\mathbf{a}^w) = \mathbf{W}\mathrm{dir}_i(\mathbf{a})$. The question is if these equations actually hold. For the first one we immediately have:

$$\mu(\mathbf{a}^w) = \sum_{i=1}^{n} \mathbf{x}_i^w = \sum_{i=1}^{n} \mathbf{W}\mathbf{x}_i = \mathbf{W} \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{W}\mu(\mathbf{a}).$$

So let us prove the second statement. Let $\mathbf{A}_0^w$ be the $m \times d$ matrix with i-th row equal to $\mathbf{x}_i^w$, put $\mathbf{A}^w = \mathbf{A}_0^w - \mathbf{1}^m \mu(\mathbf{a}^w)$ and perform the SVD of $\mathbf{A}^w$. We need to prove that the right singular vectors of $\mathbf{A}^w$ are equal to the right singular vectors of $\mathbf{A}$ transformed to the wavelet basis. In other words, we need to verify that:

$$\mathbf{UD}(\mathbf{WV})^T \text{ is the SVD of } \mathbf{A}^w \text{ if and only if } \mathbf{A} = \mathbf{UDV}^T \text{ is the SVD of } \mathbf{A}.$$

But it is very easy to show. Take $\mathbf{A} = \mathbf{UDV}^T$, multiply both sides by $\mathbf{W}^T$ and rearranging the terms:

$$\begin{aligned} \mathbf{AW}^T &= \mathbf{UDV}^T\mathbf{W}^T \\ (\mathbf{WA}^T)^T &= \mathbf{UD}(\mathbf{WV})^T \\ \mathbf{A}^w &= \mathbf{UD}(\mathbf{WV})^T. \end{aligned}$$

The matrix on the right $\mathbf{WV}$ is orthogonal, since $\mathbf{WV}(\mathbf{WV})^T$ is equal to the identity matrix, and therefore actually $\mathbf{UD}(\mathbf{WV})^T$ is the SVD of $\mathbf{A}^w$.

So now we know how to find the affine subspace for a set of light transfer vectors represented in the wavelet basis. In the following we review several existing SVD algorithms and choose one of them that allow us to exploit the sparsity of the data vectors.

4.5.2.1 Choice of a Suitable SVD Algorithm

Many methods to evaluate SVD of a rectangular matrix $\mathbf{A}$ have been proposed. An overview can be found in [GVL96]. Accurate methods first compute the covariance matrix $\mathbf{AA}^T$ (or $\mathbf{A}^T\mathbf{A}$ depending on which one has a smaller dimension). Then they transform the covariance matrix into a tri-diagonal form using the Householder transformation and solve for eigen-pairs of the tri-diagonal matrix [Dhi97]. The eigen-pairs are then used to build the diagonal matrix of singular values $\mathbf{D}$ and $\mathbf{U}$ (or $\mathbf{V}$ when $\mathbf{A}^T\mathbf{A}$ was taken). Finally, the matrix $\mathbf{V}$ (or $\mathbf{U}$) is computed so that $\mathbf{A} = \mathbf{UDV}^T$ holds.

There are several bottlenecks associated with the use of such algorithms in practice. First, they work with dense matrices only, since even if $\mathbf{A}$ was sparse, the calculation of $\mathbf{AA}^T$ would not map well to the existing computer architecture and moreover $\mathbf{AA}^T$ is dense. Second, the calculation of the SVD of $\mathbf{A}$ starts every time from scratch, i.e. the methods cannot exploit previously calculated results to update existing SVD of $\mathbf{A}$. Third, they often provide the full SVD of $\mathbf{A}$, not just the few leading top right singular vectors that we are interested in, thereby wasting the computation effort. That is why a number of SVD algorithms have been proposed trading these issues. Sirovich [Sir87] proposes an algorithm that evaluates several top leading eigen-pairs to the SVD of $\mathbf{A}$ without evaluating its covariance matrix. Another approach has been presented by Drineas

et. al. [DDH03] using a data-driven approach. They draw a subset of rows and columns of $\mathbf{A}$, scale them appropriately, and build a new small matrix $\mathbf{B}$. After that they perform an accurate SVD on $\mathbf{B}$ and use it to find an approximation to the SVD of $\mathbf{A}$. Both algorithms can exploit sparsity of $\mathbf{A}$ but still the calculation of the SVD of $\mathbf{A}$ must start every time from scratch.

Roweis [Row98] proposes a learning algorithm that works with a sparse $\mathbf{A}$ while progressively updating previously calculated top eigenvectors of $\mathbf{A}^T\mathbf{A}$ in implicit form, i.e. $\mathbf{A}^T\mathbf{A}$ is never built explicitly. The iterative nature of the algorithm is a very nice property in our case, since the algorithm can be well integrated into the LPCA: The converged leading eigenvectors estimated in the previous iteration of the LPCA can be used as starting eigenvectors when we update the affine subspaces in the next LPCA iteration. This property together with the fact that the algorithm works with the sparse $\mathbf{A}$ lead us to choose Roweis's algorithm for our purpose.

4.5.2.2 Roweis's SVD Algorithm on Sparse Transfer Vectors

We briefly summarize the algorithm proposed by Roweis [Row98] for finding the top leading eigenvectors of the covariance matrix $\mathbf{A}^T\mathbf{A}$ of $\mathbf{A}$, whose normalized versions correspond to the desired right singular vectors. At the beginning of the algorithm, l leading eigenvectors of $\mathbf{A}$ must be guessed. Since no information about the eigenvectors is often available at the beginning, randomly chosen rows of $\mathbf{A}$ are used as the initial guess. Then algorithm performs two steps until the convergence of the eigenvectors is reached: *expectation* (e-step) and *maximization* (m-step). In the expectation step, the rows of $\mathbf{A}$ are translated onto the subspace generated by the guessed leading eigenvectors. In the maximization step the translated rows are used to find an updated eigenvector that better fits the rows of $\mathbf{A}$. An example of the process of learning the leading principal vector for Gaussian distributed data is shown in Figure 4.6.

Formally, the algorithm evaluates:

$$\begin{aligned} \text{(e-step):} \quad \mathbf{X} &= \left(\mathbf{C}\mathbf{C}^T\right)^{-1}\mathbf{A}\mathbf{C}^T \\ \text{(m-step):} \quad \mathbf{C}_{new}^T &= \left(\mathbf{X}\mathbf{A}\right)^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}, \end{aligned}$$

where $\mathbf{C}$ is a $l \times d$ matrix of learned principal vectors. Let us switch to wavelet basis and take $\tilde{\mathbf{A}}^w = \tilde{\mathbf{A}}_0^w - \mathbf{1}^m\mu(\mathbf{a}^w)$ instead of $\mathbf{A}$, where $\tilde{\mathbf{A}}_0^w$ is the $m \times d$ sparse matrix formed by sparse light transfer vectors approximated in wavelet bases. Since $\mu(\mathbf{a}^w)$ is a dense vector, $\tilde{\mathbf{A}}^w$ is also dense. In order to exploit sparsity we use expanded form of $\tilde{\mathbf{A}}^w$, i.e. $\tilde{\mathbf{A}}_0^w - \mathbf{1}^m\mu(\mathbf{a}^w)$ in both steps:

$$\begin{aligned} \text{(e-step):} \quad \mathbf{X} &= \left(\mathbf{C}\mathbf{C}^T\right)^{-1}\left[\tilde{\mathbf{A}}_0^w\mathbf{C}^T - \mathbf{1}^m\left(\mu(\mathbf{a}^w)^T\mathbf{C}^T\right)\right]\mathbf{C}^T \\ \text{(m-step):} \quad \mathbf{C}_{new}^T &= \left[\mathbf{X}\tilde{\mathbf{A}}_0^w - \left(\mathbf{X}\mathbf{1}^m\right)\mu(\mathbf{a}^w)^T\right]^T\left(\mathbf{X}\mathbf{X}^T\right)^{-1}. \end{aligned}$$

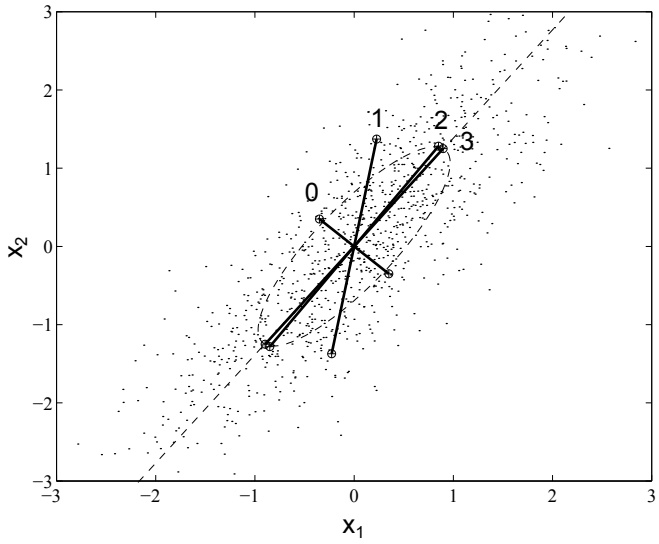


Figure 4.6 – Process of learning of the first eigenvector for a set of $2D$ vectors drawn from a Gaussian distribution. The initial direction of the eigenvector is shown by the solid line with the number 0. The direction of the eigenvector after first, second, and third iteration of the EM algorithm is shown by other solid lines. The direction of the eigenvector, to which the EM algorithm converges, is shown by the dashed line. Image courtesy of Roweis [Row98].

Here $\tilde{\mathbf{A}}_0^w$ is multiplied with dense matrix that as we mention above is very efficient, see term $\tilde{\mathbf{A}}_0^w\mathbf{C}^T$ and $\mathbf{X}\tilde{\mathbf{A}}_0^w$, respectively. Note we also rearrange brackets for when multiplying with $\mathbf{1}^m\mu(\mathbf{a}^w)$, see term $\mathbf{1}^m\left(\mu(\mathbf{a}^w)^T\mathbf{C}^T\right)$ and $(\mathbf{X}\mathbf{1}^m)\,\mu(\mathbf{a}^w)^T$, respectively, to obtain simple matrix-vector multiplications.

4.5.3 Computation of the Sparse Points

The last step of our algorithm is to calculate light transfer vectors at all other view samples that have not been selected during the exploration phase and to assign them in clusters. The question is how to calculate the light transfer vector at such view samples. We could calculate them using either hierarchical or Monte Carlo approach as discussed in Section 4.4.3. But it would be costly. Since the light transport is locally low-dimensional [MSRB07, NBB04] we can reconstruct a light transfer vector at a view sample $v_{it} \in V$ more efficiently from light transfer vectors at previously evaluated nearby view samples that lie close to v_{it} in space and time.

To find such nearby samples we use our distance metric defined by Equation 4.2. Let us say we find k nearest samples $v_{rt}, r \in 1 \ldots k$. Then we can reconstruct a sparse wavelet image of a light transfer vector at v_{it}, $\tilde{\mathbf{x}}_{it}^w$, from the

light transfer vectors $\tilde{\mathbf{x}}^w_{rt}$ at v_{rt} as:

$$\tilde{\mathbf{x}}^w_{it} \approx \sum_{r=1}^{k} \alpha_r \tilde{\mathbf{x}}^w_{rt}, \quad r \in 1 \dots k, \tag{4.5}$$

where α_r are the weights of linear combination. Assuming that elements on the corresponding indices of all light transfer vectors $\tilde{\mathbf{x}}^w_{rt}$ are of similar value (because of locality of light transport), we can pick a small subset of indices I, $|I| \geq k$, and consider a sparse reconstruction:

$$\tilde{\mathbf{x}}^w_{it}[j] \approx \sum_{r=1}^{k} \alpha_r \tilde{\mathbf{x}}^w_{rt}[j], \quad r \in 1 \dots k \text{ and } j \in I,$$

where $[j]$ is the vector operator returning j-th element of the vector. To determine α_r we need to calculate $\tilde{\mathbf{x}}^w_{it}[j]$ for all $j \in I$. But to be able to calculate the wavelet coefficient $\tilde{\mathbf{x}}^w_{it}[j]$ at any index j we would need to know the original d-dimensional light transfer vector $\mathbf{x}_{it}$ (or its approximation). To solve this difficulty we perform sparse reconstruction on the original light transfer vector (not in wavelets) as follows:

$$\mathbf{x}_{it}[j] \approx \sum_{r=1}^{k} \alpha_r \mathbf{x}_{rt}[j], \quad r \in 1 \dots k \text{ and } j \in I,$$

where $\mathbf{x}_{it}[j], j \in I$ we calculate using Equation 4.1. We reconstruct the elements $\tilde{\mathbf{x}}_{rt}[j]$ from $\tilde{\mathbf{x}}_{rt}$ using inverse wavelet transform returning elements just at requested indices. This completes a linear system of $|I|$ equations with $\alpha_r, r \in 1 \dots k, |I| > k$ as unknowns that can be solved by using the least square minimization method. Once we have α_r we reconstruct the wavelet image of the light transfer vector at v_{it}, $\tilde{\mathbf{x}}^w_{it}$, using 4.5. As the last step we assign $\tilde{\mathbf{x}}^w_{it}$ to nearest affine subspace provided by WLPCA.

4.6 Rendering

The run-time algorithm for relighting an animation sequence is straightforward. When rendering an image at frame t, we perform the following steps:

1. calculate direct lighting at gather samples $g_{jt} \in G, j \in 1 \dots d$,
2. transform direct lighting from all the gather samples $g_{jt}, j \in 1 \dots d$ to indirect lighting at view samples $v_{it} \in V, i \in 1 \dots p$ using the precomputed light transfer tensor, and finally
3. add the result to direct lighting at the view samples.

Recall that p is the number of pixels in the rendered images.

Direct lighting on gather samples The first step is to calculate direct lighting on gather samples. To account for visibility in direct lighting we can use shadow maps [LP09]. Another option is to calculate the direct lighting by tracing shadow rays. Compared to the shadow maps, ray-traced shadows provide higher-quality results without the need to tune any constants presents in the shadow maps, though at the price of higher computational costs. Considering the high performance of modern graphics cards (that are capable to trace up to two hundred million rays per second) and the requirement for a high-quality rendering, we opt to calculate direct lighting by ray tracing.

Indirect lighting on view samples Once we have calculated direct lighting at gather samples at time t, we use the precomputed light transfer tensor to calculate indirect lighting at view samples at time t. Let $v \in V$ be an arbitrary view sample at t and calculate indirect lighting at v. Indirect lighting on v can be calculated as:

$$\mathbf{x}^T\mathbf{L},$$

where $\mathbf{x}$ is the light transfer vector at v, and $\mathbf{L}$ is a d-dimensional vector whose j-th element is equal to direct lighting on g_{jt}. Note that we can obtain the same result, i.e. the indirect lighting on v, even if both vectors are expressed in wavelet basis:

$$(\mathbf{x}^w)^T\mathbf{L}^w = (\mathbf{W}\mathbf{x})^T\mathbf{W}\mathbf{L} = \mathbf{x}^T\mathbf{W}^T\mathbf{W}\mathbf{L} = \mathbf{x}^T\mathbf{L},$$

where $\mathbf{L}^w$ is the wavelet image of $\mathbf{L}$. We use the precomputed light transfer tensor to obtain an approximation of $\mathbf{x}^w$, $\tilde{\mathbf{x}}^w$. Let $\mathbf{a}^w$ be the affine subspace that contains $\tilde{\mathbf{x}}^w$, and w_k be k-th coordinate of $\tilde{\mathbf{x}}^w$ in $\mathbf{a}^w$. Then $\tilde{\mathbf{x}}^w$ can be easily reconstructed as:

$$\tilde{\mathbf{x}}^w \approx \mu(\mathbf{a}^w) + \sum_{k=1}^{l} w_k \mathrm{dir}_k(\mathbf{a}^w).$$

But reconstructing light transfer vectors at each view point at t and multiplying with $\mathbf{L}^w$ would be very inefficient. Expecting that there are many view samples assigned to $\mathbf{a}^w$, we can perform calculation of indirect lighting at these view samples more efficiently:

$$\begin{aligned}
(\tilde{\mathbf{x}}^w)^T\mathbf{L}^w &\approx \left(\mu(\mathbf{a}^w) + \sum_{k=1}^{l} w_k \mathrm{dir}_k(\mathbf{a}^w)\right)^T \mathbf{L}^w \\
&= \mu(\mathbf{a}^w)^T\mathbf{L}^w + \sum_{k=1}^{l} w_k \left(\mathrm{dir}_k(\mathbf{a}^w)^T\mathbf{L}^w\right) \\
&= H_0 + \sum_{k=1}^{l} w_k H_k,
\end{aligned}$$

where $H_0 = \mu(\mathbf{a}^w)^T \mathbf{L}^w$ and $H_k = \mathrm{dir}_k(\mathbf{a}^w)^T \mathbf{L}^w$, $k \in 1 \dots l$. Note that they are constant for given $\mathbf{L}^w$ so the calculation of indirect lighting at view sample clustered in $\mathbf{a}^w$ should be amortized. An illustration of indirect lighting calculation in a view sample v is depicted in Figure 4.7.

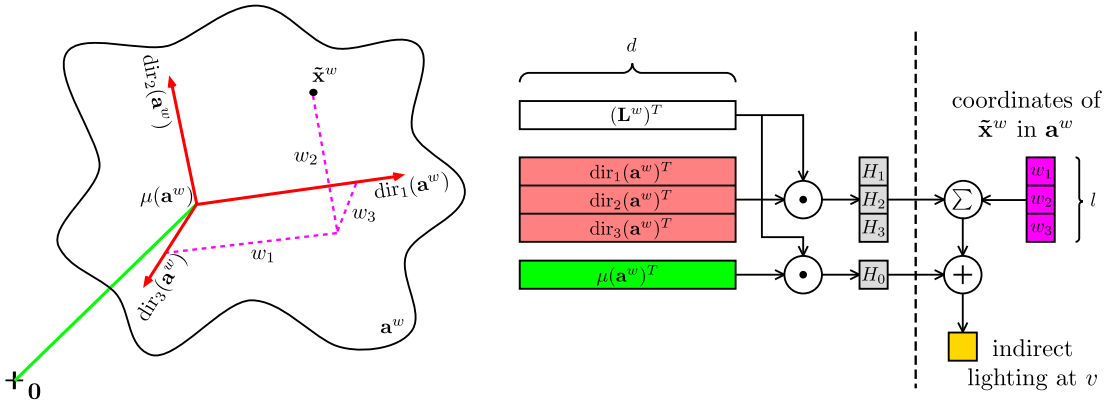


Figure 4.7 – Calculation of indirect lighting at view sample v. Image of direct lighting on gather samples at t represented in wavelet basis, $\mathbf{L}^w$, is projected onto the position vector $\mu(\mathbf{a}^w)$ and basis vectors $dir_k(\mathbf{a}^w), k \in 1 \dots l$ of $\mathbf{a}^w$ obtaining the H_0 and H_k. Having coordinates of the wavelet image of the light transfer vector with respect to an affine subspace $\mathbf{a}^w$, w_k, the indirect lighting at v is approximated by $H_0 + \sum_{k=1}^{l} w_k H_k$.

Once we have calculated indirect lighting on all $v \in V$ at t, we calculate direct lighting at those v and add direct and indirect lighting together. This produces the final rendering at t.

4.7 Conclusion

We have introduced an approach for animation relighting with indirect lighting. We used the tensor formulation to describe the direct-to-indirect light transfer over multiple frames of an animation sequence. We have suggested a possible solution for evaluating the transfer tensor in a compressed form. In our approach we take advantage of directional, spatial, and temporal coherence of the light transport to make computation of the tensor practical. We believe that our approach will be able to preserve all the important lighting effects and deliver high quality indirect lighting even in complex scenes with many various materials of different reflectance characteristics and flexible lighting models. As our approach has not been implemented yet we cannot provide any results leaving them to a future work.

Conclusion and Future Work

5

Contents

Realistic rendering of complex virtual scenes is demanded by a number computer graphics applications. This necessitates fast and accurate algorithms for realistic image synthesis. It has been shown that coherence of light transport can be exploited [DHS+05, MSRB07, PML+09] to accelerate these algorithms. This thesis describes three new algorithms for realistic image synthesis and shows how the coherence was used to make these algorithms efficient.

In Chapter 2 we developed *spatial directional radiance caching* (SDRC) to speed up GI calculation in scenes with glossy surfaces. Efficiency of SDRC is achieved by reusing cached radiance samples calculated adaptively on demand. In Chapter 3 we deal with *local principal component analysis* (LPCA) used for data compression in data-driven approaches. We developed an accelerated algorithm, *SortCluster-LPCA* (SC-LPCA), that produces exactly the same result as the original LPCA but more quickly by exploiting coherence in the compressed dataset. In Chapter 4 we presented a different modification of LPCA called *Wavelet-LPCA* (WLPCA) that combines sparse data approximation in the wavelet domain with LPCA. We used the WLPCA to compress a direct-to-indirect light transfer precomputed for an animation sequence. In the following we summarize our three contributions and suggest ideas for future work.

5.1 Spatial Directional Radiance Caching

We have presented spatial directional radiance caching (SDRC) for efficient calculation of indirect lighting in scenes with glossy surfaces. To do so we utilize spatial and directional coherence in incoming indirect radiance: we cache radiance samples whose calculation is time-consuming and reuse them whenever possible.

Our SDRC delivers high-quality images in glossy scenes. Using the BRDF importance sampling, SDRC automatically adapts to the glossiness of object surfaces. Moreover SDRC needs no conversion of the scene BRDFs into the frequency domain (spherical harmonics) in a preprocess as in the original radiance caching algorithm [KGPB05]. Our results show that caching pays off even in scenes with glossy surfaces: compared to traditional Monte Carlo methods based on BRDF importance sampling our SDRC produces less noisy images in the same time.

In the future work it would be interesting to port our algorithm to GPU and compare it with GPU implementations of other common Monte Carlo based methods [Kaj86, LW93, VG97]. Another subject for future work is to devise a more complete interpolation error criterion than the one borrowed from irradiance caching. Such new interpolation criterion should take surface BRDF into account when picking spatially nearby cache records. In addition, a more sophisticated strategy than just adding spatial and directional samples should be developed to make algorithm practical for rendering animations.

5.2 SortCluster-LPCA and SortMeans++

We proposed a novel fast and more accurate local principal component analysis (LPCA) algorithm for data compression. Having a set of high-dimensional data vectors and a number of clusters k, the goal of LPCA is to find an approximation of these vectors in k low-dimensional affine subspaces minimizing an error criterion. The original LPCA starts with a random initialization of affine subspaces. After that it alternately performs classification of the data vectors into nearest clusters as well as clusters' update. But such an algorithm is very inefficient, since distances to all affine subspaces are computed. Moreover, LPCA is prone to get stuck in a local optimum producing a less accurate data approximation. To improve the efficiency of the LPCA we developed a new algorithm called SortCluster-LPCA which explores coherence in distribution of the data vectors to eliminate unnecessary distance calculations. To address the accuracy we proposed a fast algorithm that produces better initialization for the LPCA.

We tested our SC-LPCA for compression of radiance transfer matrices used in precomputed radiance transfer (PRT) and for compression of bi-directional texture function (BTF) image databases. We achieved speed-up of 5 to 20 for radiance transfer matrices while the speed-up for BTF data sets was lower. Concerning the data approximation accuracy, we consistently achieve a lower approximation error in our tested data sets compared to simple random initialization.

There are several issues as subjects for a future works:

- Our SC-LPCA algorithm uses upper bound on the vector-cluster distance to accelerate the classification stage. In k-means Elkan [Elk03] and Hamerly [Ham10] show that further speed-up can be achieved by using also a *lower bound* on the vector-cluster distance. Generalization of the lower bound from k-means to LPCA, however, is not as simple as in the case of upper bound: when an affine subspace is updated both origin and orientation of basis vectors change. This raises the problem of how to efficiently and quickly update the lower bound.

- The next issue is that the LPCA algorithm produces sharp clustering. Two similar data vectors may be assigned to two different clusters and consequently their approximation in the respective affine subspaces may end up being quite different. This may cause some problems in applications that require smooth transitions between data vectors. Though we have not observed any artifact due to sharp clustering in PRT application for relighting of static scene, sharp clustering may cause some problems in animation relighting. It would be interesting to develop techniques for soft clustering. One possible solution to tackle the problem in the LPCA could be to approximate data vectors that lie in between affine subspaces by using basis vectors from both subspaces.

- The LPCA algorithm treats each data vector as one element. Assume we have PRT application for relighting of a simple scene with one object lit by an environment map. Let us suppose two visible points that lie close to the object and calculate the light transfer vectors for them. Then many elements in the light transfer vectors corresponding to cubemap pixels which lie in the opposite direction toward the object will be similar. On the other hand elements of the light transfer vectors that correspond to cubemap pixels visible in the directions toward the object will be very different because of occlusion. This causes problems for LPCA clustering. The question is whether it would be better to divide long light transfer vectors into several shorter sub-vectors and run LPCA on shorter sub-vectors than run LPCA on original long light transfer vectors.

5.3 Relighting of Animation Sequences.

We described an algorithm for cinematic relighting of animated sequences with global illumination. Our approach extends the idea of direct-to-indirect light transfer from a static scene to an animated sequence where the object deformation and camera path are known in advance.

In the offline part of our relighting system we calculate a light transfer tensor. We use coherence in light transport to make the precomputation of the transfer tensor feasible in terms of memory and computation time. (We remind that the raw light transfer tensor is of Tera-byte size and calculation of each element itself is costly.) First we adaptively explore the structure of the light transfer tensor placing more samples in regions where light transfer changes quickly. Then we run our Wavelet-LPCA that exploits sparsity in the light transfer vectors expressed in wavelet basis. We reconstruct the rest of the transfer tensor using the observation that light transport is locally low-dimensional [MSRB07]. As our approach has not been implemented yet we cannot provide any results. Implementing and verifying our approach is the subject for future work.

Trend in computer graphics We made a contribution in the field of realistic image synthesis making use of coherence of light transfer to speed up image synthesis. In future research we expect more efficient algorithms utilizing coherence for fast GI calculation. In addition, we expect that data-driven approaches will cement their key position in modern computer graphics. Powerful algorithms for processing and compression of large databases that exploit the coherence therein will be of foremost importance.

Author's Publications

[GK08] Gassenbauer V., Křivánek J.: Spatial directional radiance caching. In *Siggraph Asia 2008 NEW HORIZONS, Sketches and Posters* (Singapore, 2008). 9

[GKB09] Gassenbauer V., Křivánek J., Bouatouch K.: Spatial directional radiance caching. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering) 28*, 4 (2009), 1189–1198. 9

[GKB11] Gassenbauer V., Křivánek J., Bouatouch K., Bouville C., Ribardière M.: Improving Performance and Accuracy of Local PCA. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering) 30*, 7 (2011). 29

Bibliography

[AFO05] Okan Arikan, David A. Forsyth, and James F. O'Brien. Fast and detailed approximate global illumination by irradiance decomposition. *ACM Transactions on Graphics*, 24(3):1108–1114, August 2005. 11

[AKDS04] Thomas Annen, Jan Kautz, Frédo Durand, and Hans-Peter Seidel. Spherical harmonic gradients for mid-range illumination. In *Rendering Techniques*, pages 331–336, 2004. 50

[AV07] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007. 30, 32, 37, 43

[BGB08] Jonathan Brouillat, Pascal Gautron, and Kadi Bouatouch. Photon-driven irradiance cache. *Computer Graphics Forum (Proc. of Pacific Graphics)*, 27(7):1971–1978, 2008. 11

[BSH02] Philippe Bekaert, Mateu Sbert, and John Halton. Accelerating path tracing by re-using paths. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, pages 125–134, June 2002. 11

[CAE08] David Cline, Daniel Adams, and Parris Egbert. Table-driven adaptive importance sampling. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 27(4), 2008. 10

[CAM08] Petrik Clarberg and Tomas Akenine-Möller. Practical product importance sampling for direct illumination. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2):681–690, 2008. 10

[Coo86] Robert L. Cook. Stochastic sampling in computer graphics. *ACM Trans. Graph.*, 5:51–72, January 1986. 4

[CPC84] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proc. of SIGGRAPH '84*, 1984. 3

[CSH08] Francesco Castro, Mateu Sbert, and John H. Halton. Efficient reuse of paths for random walk radiosity. *Computers & Graphics*, 32(1):65–81, February 2008. 11

[CW93] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Morgan Kaufmann, 1993. 3

[DDH03] Petros Drineas, Eleni Drinea, and Patrick S. Huggins. An experimental evaluation of a Monte-Carlo algorithm for singular value decomposition. In *Proceedings of the 8th Panhellenic conference on Informatics*, PCI'01, pages 279–296, Berlin, Heidelberg, 2003. Springer-Verlag. 65

[DFK+04] Petros Drineas, Alan M. Frieze, Ravi Kannan, Santosh Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56:9–33, 2004. 32

[Dhi97] Inderjit Singh Dhillon. *A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem*. PhD thesis, EECS Department, University of California, Berkeley, Oct 1997. 64

[DHS+05] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. A frequency analysis of light transport. *ACM Trans. Graph.*, 24:1115–1126, July 2005. 3, 71

[DK92] Arthur M. DuPre and Seymour Kass. Distance and parallelism between flats in R^n. *Linear Algebra and its Applications*, 171:99–107, July 1992. 35, 37

[Elk03] Charles Elkan. Using the triangle inequality to accelerate k-means. In Tom Fawcett and Nina Mishra, editors, *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 147–153. AAAI Press, 2003. 30, 31, 73

[FH09] JiříFilip and Michal Haindl. Bidirectional texture function modeling: A state of the art survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31:1921–1940, November 2009. 4, 6, 30, 31, 45

[FPJY07] Wei-Wen Feng, Liang Peng, Yuntao Jia, and Yizhou Yu. Large-scale data management for PRT-based real-time rendering of dynamically skinned models. In *EGSR '07: Proceedings of the 17th Eurographics workshop on Rendering*, Switzerland, 2007. Eurographics Association. 4, 5, 50

[GBP07] Pascal Gautron, Kadi Bouatouch, and Sumanta Pattanaik. Temporal radiance caching. *IEEE Transactions on Visualization and Computer Graphics*, 13(5), 2007. 3, 11

[GKBP05] Pascal Gautron, Jaroslav Křivánek, Kadi Bouatouch, and Sumanta N. Pattanaik. Radiance cache splatting: A GPU-friendly global illumination algorithm. In *Rendering Techniques (Proc. of Eurographics Symposium on Rendering)*, pages 55–64, 2005. 11

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. 64

[Ham10] Greg Hamerly. Making k-means even faster. In *SIAM International Conference on Data Mining*, pages 130–140, 2010. 31, 73

[HDMS03] Vlastimil Havran, Cyrille Damez, Karol Myszkowski, and Hans-Peter Seidel. An efficient spatio-temporal architecture for animation rendering. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, pages 106–117, June 2003. 11

[Hod88] Michael E. Hodgson. Reducing the computational requirements of the minimum-distance classifier. *Remote Sensing of Environment*, 25(1):117 – 128, 1988. 31

[HPB06] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Direct-to-indirect transfer for cinematic relighting. *ACM Trans. Graph.*, 25:1089–1097, July 2006. 4, 5, 6, 46, 47, 48, 51, 52, 55, 57

[HPB07] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Matrix row-column sampling for the many-light problem. *ACM Trans. Graph.*, 26, July 2007. 3, 10, 37

[HR10] Fu-Chung Huang and Ravi Ramamoorthi. Sparsely precomputing the light transport matrix for real-time rendering. *Computer Graphics Forum (EGSR 2010)*, 29(4):1335–1345, 2010. 4, 30, 31, 46, 49, 50, 56, 58

[HS85] Dorit S. Hochbaum and David B. Shmoys. A Best Possible Heuristic for the k-Center Problem. *Mathematics of operations research*, 10(2):180–184, May 1985. 32

[HS98] Wolfgang Heidrich and Hans-Peter Seidel. View-independent environment maps. In *Proc. of Graphics Hardware*, 1998. 14

[JDZJ08] Wojciech Jarosz, Craig Donner, Matthias Zwicker, and Henrik Wann Jensen. Radiance caching for participating media. *ACM Trans. Graph.*, 27(1), 2008. 10, 11

[Jen01] Henrik W. Jensen. *Realistic Image Synthesis Using Photon Mapping.* AK Peters, Ltd., July 2001. 4, 10, 11

[JZJ08] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. Irradiance gradients in the presence of participating media and occlusions. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 27(4), 2008. 11, 19

[Kaj86] James T. Kajiya. The rendering equation. In *Proc. of SIGGRAPH '86*, 1986. 3, 72

[KAMJ05] Anders Wang Kristensen, Tomas Akenine-Möller, and Henrik Wann Jensen. Precomputed local radiance transfer for real-time lighting design. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1208–1215, New York, NY, USA, 2005. ACM. 4, 6, 51

[KBPv06] Jaroslav Křivánek, Kadi Bouatouch, Sumanta Pattanaik, and JiříŽára. Making radiance and irradiance caching practical: Adaptive caching and neighbor clamping. In *Rendering Techniques 2006 (Proc. of Eurographics Symposium on Rendering)*, pages 127–138, 2006. 19, 20

[Kel97] Alexander Keller. Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 11

[KG09] Jaroslav Křivánek and Pascal Gautron. *Practical Global Illumination with Irradiance Caching.* Morgan-Claypool, 2009. 2, 18

[KGPB05] Jaroslav Křivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *IEEE Transactions on Visualization and Computer Graphics*, 11(5), 2005. i, 3, 4, 5, 10, 11, 12, 25, 72

[KL97] Nandakishore Kambhatla and Todd K. Leen. Dimension reduction by local principal component analysis. *Neural Comput.*, 9:1493–1516, October 1997. 4, 5, 30, 31, 32, 39

[KMN+02] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892, 2002. 30, 31, 32

[KTHS06] Janne Kontkanen, Emmanuel Turquin, Nicolas Holzschuch, and François Sillion. Wavelet radiance transport for interactive indirect lighting. In Thomas Akenine-Möller Wolfgang Heidrich, editor, *Rendering Techniques 2006 (Eurographics Symposium on Rendering)*. Eurographics, jun 2006. 6, 47, 48, 51

[LF97] Paul Lalonde and Alain Fournier. Generating reflected directions from BRDF data. *Comput. Graph. Forum*, 16(3):293–300, 1997. 4

[LP09] Nan Liu and Ming-Yong Pang. Shadow mapping algorithms: A complete survey. *2009 International Symposium on Computer Network and Multimedia Technology*, pages 1–5, 2009. 68

[LRR04] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient BRDF importance sampling using a factored representation. *ACM Trans. Graph.*, 23:496–505, August 2004. 4

[LSSS04] Xinguo Liu, Peter-Pike J. Sloan, Heung-Yeung Shum, and John Snyder. All-frequency precomputed radiance transfer for glossy objects. In *Rendering Techniques*, pages 337–344, 2004. 4, 31, 49, 50

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *COMPUGRAPHICS '93*, pages 145–153, 1993. 3, 72

[LW95] Eric P. Lafortune and Yves D. Willems. A 5D tree to reduce the variance of Monte Carlo ray tracing. In *Rendering Techniques (Proc. of the Sixth Eurographics Workshop on Rendering)*, pages 11–20, 1995. 11

[LZT+08] Jaakko Lehtinen, Matthias Zwicker, Emmanuel Turquin, Janne Kontkanen, Frédo Durand, François X. Sillion, and Timo Aila. A meshless hierarchical representation for light transport. *ACM Trans. Graph.*, 27:37:1–37:9, August 2008. 6, 47, 48, 51

[Mal08] Stphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, 3rd edition, 2008. 6, 60

[MMK03] Gero Müller, Jan Meseth, and Reinhard Klein. Compression and real-time rendering of measured BTFs using local PCA. In *Vision, Modeling and Visualisation 2003*, pages 271–280. Akademische Verlagsgesellschaft Aka GmbH, Berlin, November 2003. 4, 30, 31, 45

[Moo00] Andrew W. Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, UAI '00, pages 397–405, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. 31

[MSRB07] Dhruv Mahajan, Ira Kemelmacher Shlizerman, Ravi Ramamoorthi, and Peter Belhumeur. A theory of locally low dimensional light transport. *ACM Trans. Graph.*, 26, July 2007. 3, 31, 57, 66, 71, 74

[NBB04] Shree K. Nayar, Peter N. Belhumeur, and Terry E. Boult. Lighting sensitive display. *ACM Trans. Graph.*, 23(4):963–979, 2004. 57, 66

[NRH03] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Trans. Graph.*, 22:376–381, July 2003. 4, 39, 49, 50

[NRH04] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. Triple product wavelet integrals for all-frequency relighting. *ACM Trans. Graph.*, 23(3):477–487, 2004. 49, 50

[NSK+07] Derek Nowrouzezahrai, Patricio Simari, Evangelos Kalogerakis, Karan Singh, and Eugene Fiume. Compact and efficient generation of radiance transfer for dynamically articulated characters. In *GRAPHITE '07: Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 147–154, New York, NY, USA, 2007. ACM. 5, 50

[OKP+08] Juraj Obert, Jaroslav Křivánek, Fabio Pellacini, Daniel Sýkora, and Sumanta N. Pattanaik. iCheat: A representation for artistic control of indirect cinematic lighting. *Computer Graphics Forum*, 27(4):1217–1223, 2008. 58

[ORSS06] Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In *In 47th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 165–176, 2006. 32

[PBSP08] Vincent Pegoraro, Carson Brownlee, Peter S. Shirley, and Steven G. Parker. Towards interactive global illumination effects via sequential Monte Carlo adaptation. In *Proc. of the 3rd IEEE Symposium on Interactive Ray Tracing*, pages 107–114, 2008. 11

[PH04] Matt Pharr and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann, 2004. 10, 19

[Phi02] Steven J. Phillips. Acceleration of K-means and related clustering algorithms. In *Algorithm Engineering and Experiments (ALENEX)*, pages 166–177, 2002. 30, 31, 33, 34, 42

[PM99] Dan Pelleg and Andrew Moore. Accelerating exact k-means algorithms with geometric reasoning. In Surajit Chaudhuri and David Madigan, editors, *Proceedings of the Fifth International Conference on Knowledge Discovery in Databases*, pages 277–281. AAAI Press, aug 1999. 30, 31

[PML+09] Pieter Peers, Dhruv K. Mahajan, Bruce Lamond, Abhijeet Ghosh, Wojciech Matusik, Ravi Ramamoorthi, and Paul Debevec. Compressive light transport sensing. *ACM Trans. Graph.*, 28:3:1–3:18, February 2009. 3, 71

[PVL+05] Fabio Pellacini, Kiril Vidimče, Aaron Lefohn, Alex Mohr, Mark Leone, and John Warren. Lpics: a hybrid hardware-accelerated relighting engine for computer cinematography. *ACM Trans. Graph.*, 24:464–470, July 2005. 48

[Ram09] Ravi Ramamoorthi. *Precomputation-Based Rendering*. Foundations and Trends© in Computer Graphics and Vision. Now Publishers Inc, 2009. 30

[Row98] Sam Roweis. EM algorithms for PCA and SPCA. In *Proceedings of the 1997 conference on Advances in neural information processing systems 10*, NIPS '97, pages 626–632, Cambridge, MA, USA, 1998. MIT Press. 65, 66

[SHHS03] Peter-Pike Sloan, Jesse Hall, John Hart, and John Snyder. Clustered principal components for precomputed radiance transfer. *ACM Trans. Graph.*, 22:382–391, July 2003. 4, 6, 30, 31, 33, 37, 39, 43, 49, 50

[Sir87] Lawrence Sirovich. Turbulence and the dynamics of coherent structures: Dynamics and scaling. *Quarterly of Appl. Math.*, XLV:561–590, 1987. 64

[SKS02] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21:527–536, July 2002. 4, 6, 30, 31, 49

[SP94] François Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, San Francisco, 1994. ISBN 1-558. 3

[TL04] Eric Tabellion and Arnauld Lamorlette. An approximate global illumination system for computer generated films. *ACM Transactions on Graphics*, 23(3):469, 2004. 11

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proc. of SIGGRAPH*, pages 65–76, 1997. 3, 4, 72

[VP08] Steven G. Parker Vincent Pegoraro, Ingo Wald. Sequential Monte Carlo adaptation in low-anisotropy participating media. *The Eurographics Association and Blackwell Publishing Ltd.*, 27(4):1097–1104, 2008. 11

[WABG06] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. Multidimensional lightcuts. *ACM Trans. Graph. (Proc. of SIGGRAPH)*, 25(3):1081–1088, 2006. 10

[Wal05] Bruce Walter. Notes on the Ward BRDF. Technical report PCG-05-06, Program of Computer Graphics, Cornell University, April 2005. 20

[WH92] Gregory J. Ward and Paul S. Heckbert. Irradiance gradients. In *Eurographics Workshop on Rendering*, 1992. 11

[WNLH06] Rui Wang, Ren Ng, David Luebke, and Greg Humphreys. Efficient wavelet rotation for environment map rendering. In *Rendering Techniques (Proc. of Eurographics Symposium on Rendering)*, pages 173–182, 2006. 14

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *Proc. of SIGGRAPH*, 1988. 3, 4, 5, 9, 10, 11, 15, 18

[WTL04] Rui Wang, John Tran, and David P. Luebke. All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Rendering Techniques*, pages 345–354, 2004. 49

[XJF+08] Kun Xu, Yun-Tao Jia, Hongbo Fu, Shi-Min Hu, and Chiew-Lan Tai. Spherical piecewise constant basis functions for all-frequency precomputed radiance transfer. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):454–467, 2008. 31