A Novel Approach to Compress Reflection Functions Based on PCA

Björn Olsson*
Linköping University

Anders Hast†
University of Gävle

Anders Ynnerman‡
Linköping University

Figure 1: Three examples of generated images using 25 coefficients.

Abstract

In many applications it is important to perform image-based relighting. That is, to synthesize a scene in various lighting conditions without explicitely rerendering the image. This paper proposes an efficient compression method, which after a precomputing step allows an efficient re-rendering of the scene. The best results are achieved on scenes with a limited number of materials, but it may also be used on arbitrary scenes. In this work images of a static scene are generated and the method is exemplified using a dataset of ray-traced images.

CR Categories: I.3.3 [Computer graphics]: Picture/Image Generation; I.3.7 [Computer graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing and texture

Keywords: BRDF, PCA, image-based relighting

1 Introduction

Image-based relighting is important in many applications, for example in the entertainment industry and in flight-simulators. This technique is based on methods developed decades ago and the first renderings with reflection mapping were performed by [Blinn and Newell 1976]. In the years to follow the ideas were developed further by many people and were formalized by [Greene 1986].

A very large amount of work has been performed in the field of relighting. In [Kristensen et al. 2005] a general method for real-time relighting of scenes was presented. This method used the concepts of unstructered light clouds and clustered PCA to render scenes with moving lights and dynamic cameras. Another example is the method presented by [Wood et al. 2000], who used a surface light-field to generate images of shiny objects at arbitrary lighting conditions. Another method was presented by [Nimeroff et al. 1994], who used basis images for efficient re-rendering of a scene.

The PCA method has previously been used to render transparent objects (See [Matusik et al. 2002]). In contrast to earlier methods the approach presented in their paper computed the principal components for global reflection maps. In [Matusik et al. 2002] an approach similar to JPEG-compression was used. The reflection maps were divided into zones of 16x16 pixels and each zone was compressed to a small number of coefficients. Another paper by [Epstein et al. 1995] investigated the number of eigenimages required to generate the scene. In this case global eigencomponents computed from a data set of variably lit images were used. A related approach, the eigentexture method, was described by [Nishino et al. 1999].

Another approach, matrix radiance transfer, which also builds on PCA compression of BRDFs was presented by [Lehtinen and Kautz 2003]. In [Ho et al. 2003] a related approach was described. The PCA method was applied on a set of reference images with the same view, but with different illumination conditions. However, in contrast to the method presented in this paper PCA was applied block-wise. In [Shim and Chen 2005] a statistical approach, which can be used to compare various methods estimating surface reflection functions was described. A method related to the approach presented in this paper was described in [Sloan et al. 2002]. In that paper the reflection functions were represented using spherical harmonics. CPCA (Clustered principal component analysis) is another way to compress precomputed radiance transfer, which was used in [Sloan et al. 2003]. The algorithm was implemented on graphics hardware. An interesting method using CPCA can be found in [Liu et al. 2004]. It was used to perform relighting on large models.

In this paper a new reconstruction technique applied on reflection functions is described. The goal of this work has been to develop a simple method, which is fast and efficient. This approach builds on the PCA method and is especially efficient for scenes with a limited number of textures. For the method presented in this paper the viewpoint is fixed relative to the object and a set of photographs is captured while the light source is repositioned for each photograph. In addition this representation allows a very efficient reconstruction step due to the linear basis in the PCA method. After a precomputation step the scene can be relighted in arbitrary lighting conditions.

By using this method a reflection map can be characterized with a small number of coefficients. In this paper the method is applied to still images, but it would be very easy to generalize the method to arbitrary views by predicting the image appearance for several directions and interpolating the current view. The images used in this work are ray-traced, but it would also be possible to use captured

*email:Bjorn_A_Olsson@hotmail.com
†email:Anders.Hast@hig.se
‡email: andyn@itn.liu.se

images. In the next section the method is described in detail. The results and a discussion follows.

2 Method

The method proposed in this paper was applied to one data set of ray-traced images. After the precomputation and configuration phase it was possible to generate images of arbitrary lighting conditions without explicitely resynthesizing the scene. The PCA-method was used to compress the reflection functions to small coefficient vectors.

2.1 Principal Component Analysis

The principal component analysis method (PCA for short), also named the Hotelling transform (See for example Joliffe [Joliffe 2002] or Haykin [Haykin 1999]) is a general method to compress input data to smaller representations by computing the major components of the input-data. This technique is common in statistics as well as in image processing.

The PCA method is used to decompose images into a number of basis components. The pixels in an image are rearranged into a vector $\bar{x} = [x_1, x_2, \ldots, x_N]$ of length N. We assume that we have M such vectors. The transpose is denoted $\vec{x}^T$. The mean vector $\overline{m_x}$ is defined as $\overline{m_x} = E(\bar{x})$ and the covariance matrix is $\mathbf{C_{xx}} = E((\bar{x} - \overline{m_x}) \cdot (\bar{x} - \overline{m_x})^T)$, where $E(.)$ means the expectation value of a stochastic variable. The eigenvalue problem is then defined $\mathbf{C_{xx}V} = \mathbf{VD}$, where $\mathbf{V}$ is a matrix with the eigenvectors as columns and $\mathbf{D}$ a diagonal matrix with the eigenvalues along the main diagonal.

For the resolution needed the size of the covariance matrix becomes impractical. The size of the matrix can be decreased by using dimensionality reduction [Haykin 1999]. Let $\mathbf{Y}$ be the rectangular data matrix of size MxN. The matrix has the same height as the number of vectors (M). The eigenvalue problem, $\mathbf{C_{xx}V} = \mathbf{VD}$, can be written as

$$\mathbf{C_{xx}} = \frac{1}{N^2}\mathbf{Y^T Y} \Rightarrow \frac{1}{N^2}\mathbf{Y^T YV} = \mathbf{VD} \tag{1}$$

The size of the covariance matrix is reduced from a size of NxN elements to a size of MxM elements by multiplying with $\mathbf{Y}$ from the left and changing variable $\mathbf{W} = \mathbf{YV}$.

$$\frac{1}{N^2}\mathbf{YY^T YV} = \mathbf{YVD} \Rightarrow \frac{1}{N^2}\mathbf{YY^T W} = \mathbf{WD}. \tag{2}$$

We have now computed the eigenvectors in $\mathbf{W}$, which is a sub-space of $\mathbf{V}$. To transform them to the original system $\mathbf{W}$ is multiplied with $\mathbf{Y^T}$.

$$\mathbf{W} = \mathbf{YV} \Rightarrow \mathbf{Y^T W} = \mathbf{Y^T YV} = \mathbf{C_{xx}V} \Rightarrow \mathbf{Y^T W} = \mathbf{VD} \tag{3}$$

$\mathbf{VD}$ are the eigenvectors multiplied with the eigenvalues. This scaling-factor does not alter the result since the principal components are normalized before usage

$$\overline{\phi(k)} = \frac{\mathbf{V}(:,k)}{\sqrt{\mathbf{V}(:,k) \cdot \mathbf{V}(:,k)^T}}. \tag{4}$$

An image is transformed to a coefficient vector by

$$\overline{\kappa(k)} = (\bar{x} \cdot \overline{\phi(k)}^T), k = 1, \ldots, N \tag{5}$$

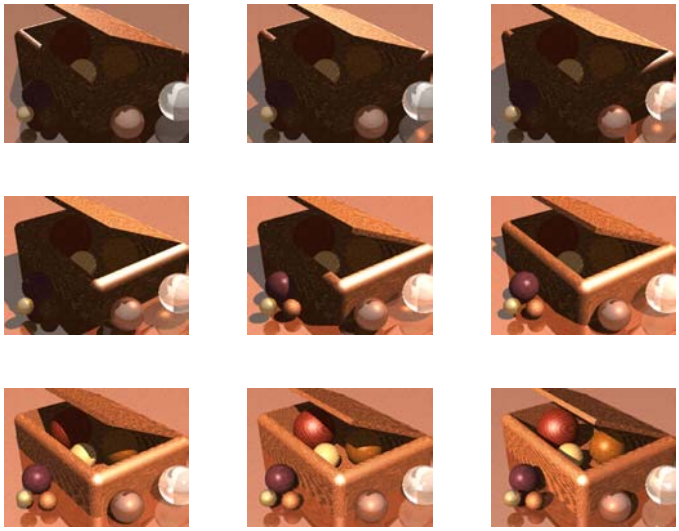

Figure 2: A number of original images for varying lighting conditons are exemplified in this figure.

This transformation is denoted $\bar{x} \overset{PCA}{\rightarrow} \overline{\kappa}$. The inverse transform $\overline{\kappa} \overset{PCA^{-1}}{\rightarrow} \bar{x}$ is defined by

$$\bar{x} = \sum_{k=1}^{N} \overline{\kappa(k)} \cdot \overline{\phi(k)} \tag{6}$$

In practical applications vector $\overline{\kappa}$ is often truncated and only the P most important coefficients are computed

$$\overline{\kappa_c(k)}, k = 1, \ldots, P \tag{7}$$

In this case the inverse transform will be inexact

$$\overline{\kappa_c} \overset{PCA^{-1}}{\rightarrow} \tilde{x}. \tag{8}$$

2.2 Precomputing and synthesizing

The method can be divided into the precomputing and synthesizing phases:

Precomputing Reflection maps, $\mathbf{R_{xy}}$, one for each pixel, are computed from the input images. These reflection maps have size QxQx3 elements. The PCA method is used to compress the reflection maps to small coefficient vectors, one vector for each pixel.

Synthesizing The scene is relighted for a specific lighting condition. This is performed by relighting each individual pixel by using the corresponding reflection function and the lightmap. By using the PCA approach it is possible to perform the computation much more efficiently.

2.3 Precomputing

The initial configuration phase can be divided into a number of steps:

Collect data set This method uses a database of images containing images of one identical scene lighted by a point source

with varying direction. The database is used to construct a method to relight the scene with arbitrary light sources. The data set can contain either captured or ray-traced images, but the best results are achieved for scenes with a limited number of surface textures. Since this work has concerned synthetic images, the colour channels are assumed to behave linearly.

Discretize the lighting directions Only the upper semi-sphere is considered in this work. It is discretized in a limited number of directions. A circular bitmap image with varying diameter is used as a model of the reflection map. Each pixel is associated with a corresponding direction. By using this technique interpolation artifacts are avoided. The directions are stored separately to be used in the generating process.

Compute reflection maps Each generated image is associated with a specific direction. For each pixel a separate reflection map is computed. This is performed by picking one pixel from each generated image and locating them in the associated positions. The procedure is exemplified in Figure 3, in which the pixel-information of the images is transformed to the reflection functions. The reflection matrices are stored in a 5D data structure (image_x, image_y, r_map_x, r_map_y, {rgb}).

Rotation of reflection maps All reflection maps are rotated along their major axes. The colour planes are not rotated individually, but instead the intensity of each pixel is computed by $I = R + G + B$. The major axis of the corresponding binary image is computed by thresholding the intensity levels. Pixels with $I(x,y) > threshold$ are included in the calculation.

The major axis θ is computed by the following method:

- $f_{xx} = 0$, $f_{yy} = 0$ and $f_{xy} = 0$.
  1. $\Delta x = x - m_x$, where x is the current x coordinate and m_x the mean x position.
  2. $\Delta y = y - m_y$, m_y is the mean y position and y the current y coordinate.
  3. $f_{xx} = f_{xx} + \Delta x \cdot \Delta x$
  4. $f_{yy} = f_{yy} + \Delta y \cdot \Delta y$
  5. $f_{xy} = f_{xy} + \Delta x \cdot \Delta y$

  Then perform the following calculations:
- $p = -(f_{yy} + f_{xx})$
- $q = f_{xx} \cdot f_{yy} - f_{xy} \cdot f_{xy}$
- $f = -\frac{p}{2} + \sqrt{\frac{p \cdot p}{4} - q}$
- The mean axis will be $\theta = \frac{q}{f}$

All reflection maps are rotated to have their major axes in the same direction. The rotation angles are stored in a separate matrix, which is used in the synthesizing phase.

Compute the eigencomponents In the next step the PCA method is used to compute a number of the most important eigencomponents, $\phi_{\mathbf{1}}, \ldots, \phi_{\mathbf{P}}$ from the reflection maps. A large number of reflection maps is needed to compute sharp components.

Compute coefficients The reflection maps are projected one by one onto the most important eigencomponents and the coefficients are stored. The result will be a coefficient matrix with one coefficient vector for each pixel of the image, $\kappa(\mathbf{1}, \ldots, \mathbf{P}; \mathbf{1}, \ldots, \mathbf{M})$, where P is the number of coefficients and M is the number of pixels.

Transform the lightmap to this specific representation The original lightmap defining the lighting conditions is rescaled to L_{xy} of size QxQ, where Q is the size of the reflection map.

Precompute the rotated eigencomponents: $\phi(1, \ldots, N, 1 : \sigma : 180)$ To make the computations more efficient, rotated versions of the eigencomponents are computed before the calculations. The angle is discretized in steps of size σ. In the calculations the eigencomponent closest to the angle is used. If the reflection functions are of a small size the error introduced will be limited.

Image representation The result will be a coefficient matrix with P coefficients for each pixel and a data structure with the rotated eigencomponents. In the beginning of the synthesizing phase the eigen-numbers are computed by using the appropriate lightmap.

2.4 Synthesizing

In the synthesizing phase the data volume in Figure 5 is used. One column consisting of a coefficient vector and a rotation vector is applied to construct one reflection function.

Collect lighting conditions Acquire an HDR-fisheye representation of the lighting conditions.

Rescaling Recompute the light representation to this specific discretization. The result will be a QxQx3 matrix, $\mathbf{L_{xy}}$. In the synthesizing phase the data volume in Figure 4 is used. One column consisting of a coefficient vector and a rotation vector is applied to construct one reflection function.

Algorithm In this section the principle behind the algorithm is presented. It is described in more detail in the upper row of Figure 4. One pixel of the resulting image is generated at a time:

1. Generate a reflection function $\mathbf{R_{xy}}$ from the corresponding coefficient vector $\overline{\kappa}$ for pixel ψ.

$$\mathbf{R_{xy}} = \sum_{k=1}^{P} \overline{\kappa(k)} \cdot \phi(\mathbf{k}), \tag{9}$$

   P is the number of coefficients and $\phi(\mathbf{k})$ the eigenvectors.

2. Rotate the reflection map angle θ.

$$\mathbf{R}^{\theta}_{\mathbf{xy}} = rot(\mathbf{R_{xy}}, \theta) \tag{10}$$

3. For each colour plane multiply the light representation with the reflection map element by element and sum the elements to three scalar values.

$$C(\psi) = \sum_{x=1}^{M} \sum_{y=1}^{M} (R^{\theta}_{xy}(:,:,\psi) \cdot L_{xy}(:,:,\psi)). \tag{11}$$

Efficient algorithm The algorithm presented in the previous section can be made more efficient by pre-computing the eigennumbers. This approach is described in more detail in the second row of Figure 4. One pixel is generated at a time.

To simplify the calculations the eigen-numbers, $\chi(k, \theta)$, are introduced. These are precomputed summations of the multi-

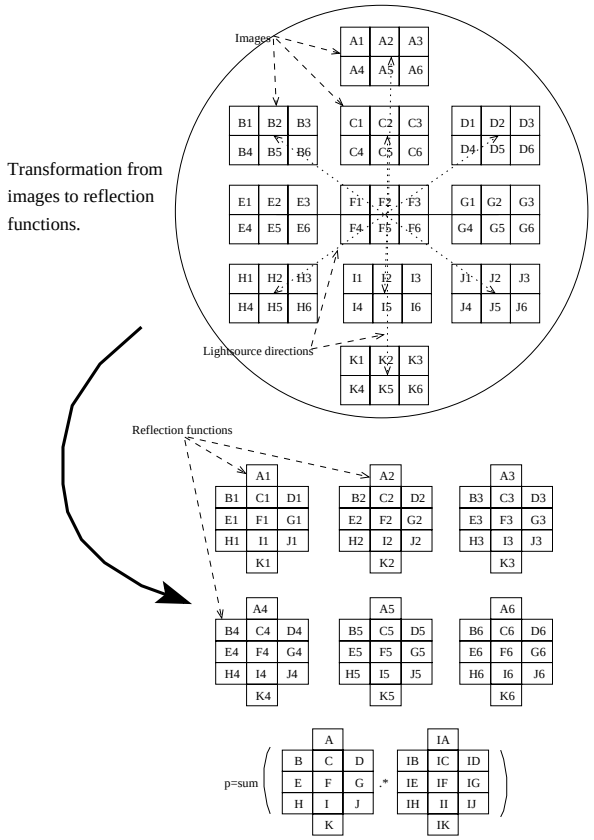

Figure 3: The scene is illuminated with a point light source located in the direction defined by the lightmap. In this figure 11 images with 6 pixels each are captured at varying locations of the lightmap. The pixel-information is then transformed to the reflection functions containing 11 pixels each, which characterize the reflection properties for a specific pixel. In the synthesizing phase the reflection function for a specific pixel is multiplied element by element with the lightmap. The resulting pixel value is equal to the sum of all elements.

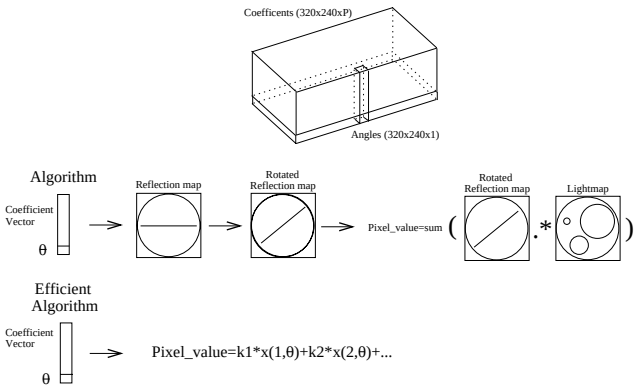

Figure 4: This figure explains the method. The data is represented in a volume, in which each column is used to construct one reflection function. In the first row the algorithm is explained and the second row is a description of the optimized algorithm.

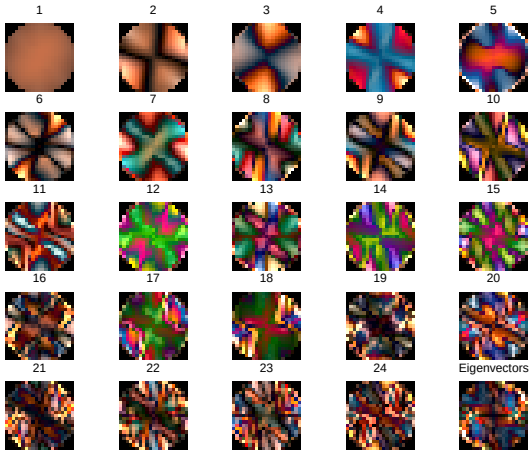

Figure 5: These are the 25 most important eigencomponents for the examined data set.

plications between the rotated eigen-components and the reflection functions

$$\chi(k,\theta) = \sum_{x=1}^{M}\sum_{y=1}^{M} \phi(\mathbf{k},\theta)\cdot\mathbf{R_{xy}}, k = 1:1:n, \theta = 0:\sigma:180. \tag{12}$$

- ψ is the pixel number.

$$\mathbf{R_{xy}}(\psi) = \sum_{k=1}^{N} \kappa(\psi,\mathbf{k})\cdot\phi(\mathbf{k},\theta) \Rightarrow \tag{13}$$

$$\overline{C(\psi)} = (\mathbf{R_{xy}}\cdot\mathbf{L_{xy}}) =$$
$$([\kappa(\mathbf{1},\psi)\cdot\phi(\mathbf{1},\theta)+\ldots+\kappa(\mathbf{P},\psi)\cdot\phi(\mathbf{P},\theta)]\cdot\mathbf{L_{xy}}) = \tag{14}$$

$$\kappa(\mathbf{1},\psi)\cdot[\phi(\mathbf{1},\theta)\cdot\mathbf{R_{xy}}]+\ldots+\kappa(\mathbf{P},\psi)\cdot[\phi(\mathbf{P},\theta)\cdot\mathbf{L_{xy}}] = \tag{15}$$

$$\kappa(\mathbf{1},\psi)\cdot\chi(1,\theta)+\ldots+\kappa(\mathbf{P},\psi)\cdot\chi(P,\theta)$$

- To generate one pixel value (P-1) additions and P multiplications are needed. The resulting equation will for each pixel and colour plane be:

$$C(\psi) = \sum_{k=1}^{P} \kappa(\mathbf{k},\psi)\cdot\chi(\mathbf{k},\theta) \tag{16}$$

3 Implementation and results

There are two main components to the system, the pre-computing phase, in which the data matrices are computed and the synthesizing, where images are generated using novel lighting conditions.

The first phase was implemented in Matlab and the second in C++. An image data set containing 374 images was generated by using MegaPOV (A version of POVray [Povray] allowing rendering of HDR images) and this data set was used to evaluate the

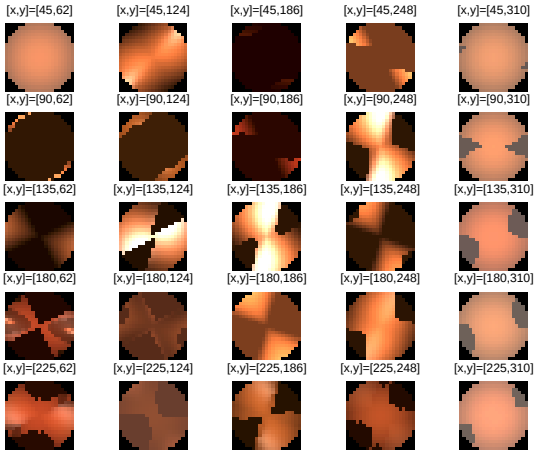

Figure 6: These images are examples of original reflection maps for the data set.

method. The data set contained images of a complicated scene, with a large amount of shadows. In this initial setup small images of size 320x240 pixels were used. Examples of input images can be seen in Figure 2. This dataset was rearranged to reflection functions (See Figure 6) of size 21x21 pixels, which were rotated along their major axes. The 25 most important eigencomponents (See Figure 5) were computed from the rotated reflection functions. It was chosen to compute the 25 most important eigencomponents from the rotated reflection functions, since the use of additional components resulted in little difference in the resulting images. All rotated reflection functions were compressed using the eigencomponents and the coefficients were stored in a data volume together with the rotation angle. To speed up the computations rotated versions of the eigencomponents were precomputed for each of the 360 degrees. The synthesis of an image was divided into two phases. In the first phase lighting coefficients were computed by multiplying the lightmap with the eigencomponents. (See the algorithm in section 2.4). In the second phase the pixel values were computed by using the lighting coefficients and the eigen coefficients. These computations were repeated for every frame.

Resulting images were computed for three well-known HDR fisheye images (See [Debevec and Malik 1997] for more information). These HDR fisheye images were transformed to the same representation as the reflection functions (The upper semi-sphere was used and this was rescaled to 21x21 pixels). For every lightprobe image the corresponding scene was generated using 10 and 25 coefficients and by using the corresponding uncompressed reflection functions (Images generated using 25 coefficients can be seen in Figure 1 and images generated using 10 coefficients and by using the corresponding original lightprobe images can be seen in Figure 7).

The frame-rate for varying numbers of coefficients can be seen in Table 1. As a measure of the image quality the peak signal-to-noise ratio was selected

$$PSNR = 20 * log_{10}(\frac{MAX_I}{\sqrt{MSE}}), \tag{17}$$

where

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \sum_{k=1}^{3} ||im_{orig}(i,j,k) - im_{\kappa}(i,j,k)||^2 \tag{18}$$

In these equations $m \cdot n$ is the size of the image. The reference image generated using the original reflection functions is im_{orig} and an image generated using κ coefficients is im_{κ}. MAX_I is the maximum intensity of im_{orig}.

κ	fps	$PSNR_{lp1}$	$PSNR_{lp2}$	$PSNR_{lp3}$
1	11.25	19.1	20.7	20.0
2	10	18.9	21.6	18.5
3	7.8	18.8	21.7	18.3
5	5.45	19.0	25.5	18.8
10	3.46	19.0	25.4	18.7
25	1.6	19.0	25.6	18.6

Table 1: A comparison of the resulting speed and image quality for various selections of coefficient numbers. The first column is the number of coefficients used, the second is the number of frames per second for the corresponding number of coefficients and the three following columns include the PSNR value for each of the lightprobes.

The errors for different choices of κ were computed for the selections of lightprobe images depicted in the third column of Figure 7. In Table 1 it can be seen that the PSNR increases for increasing numbers of coefficients for the second lightprobe, but for the others the PSNR is approximately constant. The difference in PSNR for 10 and 25 coefficients is very small. However, when examining the resulting images (Figure 1 and 7) it can be seen that the details of some of the spheres become better, when using additional components even if the PSNR value does not improve.

4 Discussion

In this paper we have presented a new approach for real-time relighting of a static scene. This method is general and can be used to relight a scene with arbitrary lightprobe images. It could for example be used to visualize a scene at arbitrary weather conditions by using synthetic sky images (See for example [Olsson et al. 2004] or [Olsson 2005]). The approach could be advantageous for example in scenes with a limited number of materials. The results show that it is feasible to represent BRDFs using principal components. For the data set used in this work 25 coefficients are sufficient. This can be seen by comparing the images in Figures 1 and 7. The images in Figure 1 are very similar to the images in the middle column in Figure 7, which were computed using uncompressed reflection functions. The major limitation is that semi-shadows are not always rendered correctly. The relatively small reflection functions of 21x21 pixels limit the possible shadows, but the limitation can be removed by using larger matrices to represent the reflection functions. A limitation of the method is that it demands detailed reflection functions to compute sharp components. The reflection functions are rotated along their major axes in order to to take the correlation between reflection functions into account.

This is a simple approach and can not directly be compared with more general, but also more complicated approaches as for example the method presented in [Kristensen et al. 2005]. The goal of this method is to relight a still image in novel lighting conditions, while the goal in their paper is to relight an arbitrary scene with unstructured light clouds using precomputed radiance transfer. The major advantage of this method is that each pixel of the resulting image can be computed with a fixed number of 25 multiplications and 24 additions, while the more advanced algorithms need a much larger number of operations in the generation process. On the other hand, the largest drawback is the precomputing step and the relatively large data matrices, which must be stored to be able to gen-

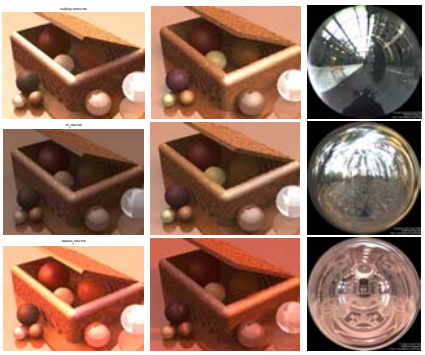

Figure 7: Each row of images corresponds to a specific lighting condition defined by an associated light probe image. The first image was generated using 10 coefficients, the second was generated using the original reflection functions and in the third image the corresponding lightprobe image can be seen.

erate images. The memory usage is

$$x_{size} * y_{size} * 3 * (Nbr_{coefficients} + 1), \quad (19)$$

which means that for an image of size 1024 x 1024 pixels, as much as 76 megabytes is needed to store the information.

The memory demand increases for larger choices of reflection functions, but the number of coefficients do not increase at the same rate, which means that after the precomputing step it should be possible to use much larger reflection functions and thus increase the image quality, without severely increasing the memory usage of the data matrices.

This method should be most appropriate for small images to make it possible to easily generate an image for various lighting conditions, for example the appearance of a car during the day.

5 Future Work

As a major limitation of this implementation is the computing time, our major efforts will be directed in decreasing the execution time. This can be performed by implementing parts of the algorithm in hardware. See for example [Owens et al.]. Another future work is to generalize the algorithm to 3D, by generating multiple views.

References

BLINN, J. F., AND NEWELL, M. E. 1976. Texture and reflection in computer generated images. In *Communications of the ACM*, vol. 19, 542–547.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH*, 369–378.

EPSTEIN, R., HALLINAN, P. W., AND YUILLE, A. L. 1995. 5±2 eigenimages suffice: An empirical investigation of low-dimensional lighting models. In *IEEE Workshop on physics-based vision*, 108–116.

GREENE, N. 1986. Environment mapping and other publications of world projections. In *IEEE Computer Graphics and Applications*, vol. 6.

HAYKIN, S. 1999. *Neural Networks*. Prentice Hall, New Jersey.

HO, P.-M., WONG, T.-T., AND LEUNG, C.-S. 2003. Compressing the illumination-adjustable images with principal component analysis. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, 59–64.

JOLIFFE, I. T. 2002. *Principal Component Analysis*. Springer-Verlag, New York.

KRISTENSEN, A. W., AKENINE-MÖLLER, T., AND JENSEN, H. W. 2005. Precomputed radiance transfer for real-time lighting design. In *Proceedings of ACM SIGGRAPH*. 1208–1215.

LEHTINEN, J., AND KAUTZ, J. 2003. Matrix radiance transfer. In *Proceedings of the 2003 symposium on interactive 3D graphics table of contents*, 59–64.

LIU, X., SLOAN, P.-P., SHUM, H.-Y., AND SNYDER, J. 2004. All-frequency precomputed radiance transfer for glossy objects. In *Eurographics Symposium on Rendering*.

MATUSIK, W., PFISTER, H., ZIEGLER, R., NGAN, A., AND MCMILLAN, L. 2002. Acquisition and rendering of transparent refractive objects. In *Proceedings of Eurographics Workshop on Rendering*, 267–278.

NIMEROFF, J. S., SIMONCELLI, E., AND DORSEY, J. 1994. Efficient re-rendering of naturally illuminated environments. In *Eurographics workshop on rendering*.

NISHINO, K., SATO, Y., AND IKEUCHI, K. 1999. Eigen-texture method: Appearance compression based on 3D model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 618–624.

OLSSON, B., YNNERMAN, A., AND LENZ, R. 2004. Visualizing weather with synthetic high dynamic range images. In *Proceedings of SPIE - IS & T Electronic Imaging. Visualization and Data Analysis*, R. F. Erbacher, J. C. Roberts, M. T. Gröhn, and K. Börner, Eds.

OLSSON, B. 2005. *Image based visualization methods for meteorological data*. Licentiate thesis, Department of Science and Technology, Linköping University. ISBN 9185297003.

OWENS, J. D., LUEBKE, D., GOVINDARAJU, N., HARRIS, M., KRUGER, J., LEFOHN, A. E., AND PURCELL, T. J. A survey of general-purpose computation on graphics hardware. In *Eurographics 2005, State of the Art Reports*, 21–51.

POVRAY. Persistence of vision raytracer (version 3.6). *Computer software, Retrieved from http://www.povray.org/download/*.

SHIM, K. H., AND CHEN, T. 2005. A statistical framework for image-based relighting. In *IEEE Conference on Acoustics, Speech, and Signal Processing (ICASSP)*.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency light environments. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques*. 527–536.

SLOAN, P.-P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered principal components for precomputed radiance transfer. In *ACM SIGGRAPH*. 382–391.

WOOD, D. N., AZUMA, D. I., ALDINGER, K., CURLESS, B., DUCHAMP, T., SALSIN, D. H., AND STUETZLE, W. 2000. Surface light fields for 3d photography. In *Proceedings of the 27th annual conference on computer graphics and interactive techniques*. 287–296.