Compact Representation of Reflectance Fields using Clustered Sparse Residual Factorization

Hideshi Yamada* Pieter Peers† Paul Debevec†
Sony Corporation* University of Southern California†
Institute for Creative Technologies

Abstract

We present a novel compression method for fixed viewpoint reflectance fields, captured for example by a Light Stage. Our compressed representation consists of a *global* approximation that exploits the similarities between the reflectance functions of different pixels, and a *local* approximation that encodes the per-pixel residual with the global approximation. Key to our method is a clustered sparse residual factorization. This sparse residual factorization ensures that the per-pixel residual matrix is as sparse as possible, enabling a compact local approximation. Finally, we demonstrate that the presented compact representation is well suited for high-quality real-time rendering.

1 Introduction

Image-based relighting enables creating detailed relit images of a scene under arbitrary distant illumination [Debevec et al. 2000]. The light transport through a scene is described by a *reflectance field*, and is captured during the data acquisition phase. Under the assumption that the viewpoint is fixed, and the incident illumination is restricted to distant illumination only, this reflectance field is a $4D$ function: parameterized over $2D$ outgoing illumination directions (i.e., camera pixels), and $2D$ incident illumination directions (i.e., number of light sources on the Light Stage). Computing a relit image consists of taking an inner product between the incident illumination and each pixel's reflectance function.

Image-based relighting owes its popularity to its simple, yet effective, nature. Lighting conditions can be changed at any time after acquisition yielding photo-realistic results. Compared to a full light transport simulation this requires relatively little computation, that can be done at nearly interactive rates. Complex light transport effects are essentially obtained for free with image-based relighting. Due an increased interest in image-based relighting, the desire for higher resolution reflectance fields (in both camera as lighting space) has grown. This also increases storage and computational requirements. Transmission and relighting speeds are becoming a limiting factor when using these high resolution reflectance fields.

In this paper we describe and investigate a novel compression method that is able to reduce the storage requirements by utilizing a different factorization method. In addition to reducing storage requirements, our method similarly reduces the time required to compute a relit image. Key to our method is a *sparse residual factorization*. This method is able to separate a collection of reflectance functions into a few basis functions, and at the same time ensures that the difference between each reflectance function expressed in the new basis and the original reflectance function contains as much as possible elements of (near) zero magnitude (i.e., has a sparse residual). Similar to [Sloan et al. 2003] we use a clustered approach using our sparse residual factorization to obtain a *global* approximation of the reflectance field. The per-pixel residual is afterwards compressed by keeping the n largest elements.

This paper makes the following contributions:

- A factorization method that minimizes the sparseness of the residual of the factorization.
- A new technique for representing reflectance fields that attains good compression ratios for both lossless and lossy compression.
- A relighting system that runs at interactive speeds on a CPU-only implementation, and at real-time speeds on consumer graphics hardware.

This paper is organized as follows. Section 2 overviews related work. In Section 3, the different key components of the presented compression method are detailed: sparse residual factorization (Section 3.2), clustered factorization (Section 3.3), and the compression of the residual (Section 3.4). Next, we discuss our real-time system in Section 4, followed by a discussion of the obtained results in Section 5. Finally, Section 6 concludes this paper with some thoughts on avenues for future research.

2 Related Work

We overview work related to the methods presented in this paper: factorization, precomputed radiance transfer, and the compression of reflectance fields.

Factorization. Matrix factorization has been used before for compactly representing high dimensional datasets. Different techniques such as principal component analysis (PCA) [Kautz and McCool 1999; Furukawa et al. 2002; Vasilescu and Terzopoulos 2004], homomorphic factorization [Latta and Kolb 2002; Suykens et al. 2003], independent component analysis (ICA) [Tsumura et al. 2003], non-negative matrix factorization (NMF) [Lawrence et al. 2004; Peers et al. 2006], clustered PCA [Sloan et al. 2003], and sparse alternating constraint least squares (SACLS) [Lawrence et al. 2006] have been successfully applied to various problems in computer graphics. The presented method is most similar to Sloan *et al.* [Sloan et al. 2003] and Lawrence *et al.* [Lawrence et al. 2006]. As in Sloan *et al.*, we also use a clustered approach, but using a different factorization method. Similar to Lawrence *et al.*, we also enforce sparsity. However, the difference is that we enforce sparsity on the residual, while they enforce sparsity on one of the factors. Additionally, we use a different sparseness constraint.

Precomputed Radiance Transfer. Precomputed radiance transfer is related to this paper in the sense that one of its goals is to compactly represent the light transport through a virtual scene such that it can be rendered at interactive rates using graphics hardware. Although the presented method is specifically geared towards compressing reflectance fields for image-based relighting, it is still very related to precomputed radiance transfer. A large number of precomputed radiance transfer systems first represent the light transport in a spherical harmonics basis [Sloan et al. 2002; Lehtinen and Kautz 2003] which is well suited to express the low-frequency content in the transport matrix. High-frequency content, however, is omitted or filtered out. Our acquired reflectance fields can contain both low and high-frequency content. All-frequency precomputed radiance transfer methods [Ng et al. 2003; Ng et al. 2004; Wang

*Work performed while visiting USC/ICT in 2007

et al. 2004a] use sophisticated compression methods such as non-linear wavelet approximations. In general, these methods do not exploit the similarities between surface points/BRDFs/pixels.

Compression of Reflectance Fields. Compression of reflectance fields has been addressed in a number of separate publications.

Debevec *et al.* [Debevec et al. 2000] briefly mention that by compressing the incident illumination and the reflectance functions using JPEG compression, computational costs and storage requirements are reduced by a factor 20. However, no detailed statistics are provided. Masselus *et al.* [Masselus et al. 2004] study the upsampling and compression of reflectance functions in detail. They apply a non-linear wavelet approximation of each reflectance function separately using different kinds of wavelets. A $1:34$ compression ratio is reported yielding only a 1% error on the reflectance functions itself. However, due to the lack of high resolution reflectance functions, these results were obtained on upsampled reflectance functions (from 1280 samples to 256×64 samples). The effect of this upsampling has not been investigated in detail. Both of the previous methods focus on compressing the reflectance field in the incident illumination domain.

Einarsson *et al.* [Einarsson et al. 2006] compressed the reflectance field in image-space, arguing that due to their very low incident illumination resolution, it is more optimal to exploit the spatial coherency. A $1:16$ compression ratio is reported for both JPEG and a wavelet based compression (using the Daubechies D4 wavelet). In the first case, the linear radiance images are compacted into 8 bits using a gamma-correction. Due to this non-linear storage, each of the JPEG images needs to be decompressed before a relit image can be computed. This results in a slight increase in computational cost.

Wong and Leung [Wong and Leung 2003] use a spherical harmonics representation of the reflectance functions. Inter-pixel relations are exploited using an image-space wavelet compression. Although this method attains good compression ratios, the use of spherical harmonics (25 coefficients) severely limits the ability to represent specular reflections accurately. Leung *et al.* [Leung et al. 2006] argue that radial basis functions are a better choice for compressing reflectance fields. Their analysis shows that the quality is slightly less than using spherical harmonics. Again, accurately compressing reflectance functions containing high frequency content is difficult. Wang *et al.* [Wang et al. 2004b] follow a similar approach, except that they represent the reflectance functions using spherical wavelets. For low compression ratios, their method is superior to using spherical harmonics and radial basis functions. However, at high compression ratios spherical harmonics yield less error.

Finally, Ho *et al.* [Ho et al. 2005] use a factorization method to attain storage reduction. A principal component analysis is performed on 16×16 image blocks. The subdivision in blocks ensures that the computations are tractable and help to better exploit spatial coherence. The obtained eigen-images are further compressed using the discrete cosine transform. The authors report compression ratios between $1:100$ and $1:500$. However, since only a low number of principal components are kept (between 5 and 10), it is not clear if this method is suited for reflectance functions with both low and high frequency content.

Unlike our technique, none of these methods exploit both spatial and incident illumination coherency for improving both compression *and* all-frequency relighting computations.

3 Compact Representation

In this section we derive the necessary theory for our compact factored representation. First a general overview is given, followed by a detailed study and derivation of the sparse residual factorization algorithm that lies at the core of our representation. Next, the clustering algorithm is described, followed by a short description of the residual compression method.

In this paper we will use the following notations: Scalars are denoted as: s, a vector as: $\mathbf{v}$, and a matrix as: $\mathbf{A}$. Functions are denoted as: $function(...)$. The i-th element of a vector $\mathbf{v}$ is denoted by $\mathbf{v}_i$. The element on the k-th column, and l-th row of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}_{l,k}$. The vector given by the l-th row a of matrix is denoted by: $\mathbf{A}_l$.

3.1 Overview

Similarly to [Sloan et al. 2003], we create a global approximation of the reflectance field using a clustered factorization (Subsection 3.3):

$$\mathbf{R}_{p,i} \approx \sum_{t}^{terms} \mathbf{W}_{p,t}\mathbf{F}_{t,i}^{c(p)}, \tag{1}$$

where $\mathbf{R}$ represents the reflectance field, p goes over n camera pixels, and i goes over m incident light directions. $\mathbf{F}^{c(p)}$ is a matrix containing the basis functions for the cluster $c(p)$, and is parameterized over terms t and incident light directions i. $\mathbf{W}$ is a matrix that contains for each pixel p the weights corresponding to the basis $\mathbf{F}^{c(p)}$, and is parameterized over pixels p, and terms t.

For many general reflectance fields this global approximation will almost never be exact, unless a prohibitively large number of clusters and terms is used. In particular, specular reflections and shadows can pose problems. These effects are localized, and thus require very small clusters or many terms (approximately one per camera pixel) to be represented accurately. Due to this local character, a better solution is to encode these effects locally per pixel. This can be done by storing the (partial) residual of each pixel. The residual $\mathbf{E}$ of a reflectance function is the difference between its approximation and the original reflectance function:

$$\mathbf{R}_{p,i} = \sum_{t}^{terms} \left(\mathbf{W}_{p,t}\mathbf{F}_{t,i}^{c(p)}\right) + \mathbf{E}_{p,i}. \tag{2}$$

From this equation, it is clear that the dimensionality of the residual $\mathbf{E}$ is the same as the full reflectance field $\mathbf{R}$. As such, this global-local separation will not necessarily lead to a compact representation. Crucial for obtaining a reduction in storage requirements is that this residual is easily compressible using, for example, a non-linear approximation. To achieve a good non-linear approximation, the residual should contain many zero (or insignificant) elements. To ensure this, we develop a novel factorization method that ensures that the residual is as sparse as possible.

In many cases, it is required that the residual is sparse in a specific basis, such as a wavelet of a spherical harmonics basis. Transforming both sides of equation (2) in a basis ψ, we obtain:

$$(\mathbf{R}\psi)_{p,i} = \sum_{t}^{terms} \left(\mathbf{W}_{p,t}(\mathbf{F}^{c(p)}\psi)_{t,i}\right) + (\mathbf{E}\psi)_{p,i}. \tag{3}$$

$\mathbf{R}\psi$ is the original reflectance field expressed in the basis ψ. Likewise, $\mathbf{F}^{c(p)}\psi$ is the projection of each basis $\mathbf{F}^{c(p)}$ into the basis ψ, and $\mathbf{E}\psi$ is the projection of the residual matrix $\mathbf{E}$. From this equation it is clear that by factorizing the reflectance field expressed in the desired basis $\mathbf{R}\psi$, the resulting residual is also expressed in this

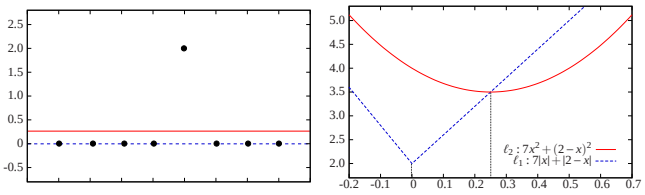

Figure 1: *Illustration of the difference between the ℓ_1-norm and the ℓ_2-norm. Left: Eight measurements are shown as dots through which the optimal vertical line is fitted to minimize the error with respect to the ℓ_2-norm (red) and the ℓ_1-norm (blue). The error function is obviously sparser (i.e., yields a zero error at many measurements) for the ℓ_1-norm than for the ℓ_2-norm. Right: the error functions plots. The ℓ_1-norm reaches its minimum at 0, while the ℓ_2-norm reaches its minimum at 0.25.*

basis, and thus enforcing sparseness of the residual on the factorization of the transformed reflectance field, yields a residual that is sparse in the desired basis.

Because the basis transformation can be applied beforehand to the reflectance field **R**, and is transparent to the actual factorization process, we will drop the basis transformation and implicitly assume that the reflectance field is implicitly expressed in the desired basis.

The remainder of this section is organized as follows. In subsection 3.2, we first discuss how to compute a factorization of the reflectance field **R** in **W** and **F** matrices, ignoring clustering (i.e., we assume that there is only one cluster). Next, in subsection 3.3 the clustering procedure in described. Finally, the compression of the residual **E** is discussed (subsection 3.4).

3.2 Sparse Residual Factorization

In this section we discuss how to compute a factorization of a matrix, such that the residual is as sparse as possible. Formally, an $n \times m$ matrix **R** is decomposed into matrices **W** and **F** (with size $n \times t$, and $t \times m$ respectively), such that the residual **E** is sparse:

$$\mathbf{E} \;\; max.\, sparse, \;\; s.t. \;\; \mathbf{R} = \mathbf{WF} + \mathbf{E}. \tag{4}$$

Compression is achieved if the number of terms t is smaller than $\frac{n \times m}{n+m}$ (excluding the residual). Sparseness is measured by counting the number of non-zero elements. This can formally be denoted using the ℓ_0-norm. Thus, we can recast Equation (4) as:

$$\min \|\mathbf{E}\|_0, \;\; s.t. \;\; \mathbf{R} = \mathbf{WF} + \mathbf{E}. \tag{5}$$

This problem, however, is computationally intractable except for small problem sizes. Fortunately, we can achieve a similar behavior as the ℓ_0-norm by using the ℓ_1-norm instead. An ℓ_1-minimization has the property that it tends to prefer a sparse solution on the condition that the solution is sufficiently sparse [Chen et al. 1999]:

$$\min \|\mathbf{E}\|_1, \;\; s.t. \;\; \mathbf{R} = \mathbf{WF} + \mathbf{E}, . \tag{6}$$

Figure 1 shows an example that compares the ℓ_1-norm and the well-known ℓ_2-norm in order to give further insight in the properties of the ℓ_1-norm. An ℓ_2 minimization tends to minimize the energy over the whole residual, while an ℓ_1 minimization tends to prefer elements of smaller magnitude, and in the limit yields a sparser residual.

A similar factorization is used in [Ke and Kanade 2005] to obtain a stable decomposition in the presence of outliers. It is interesting to note that although the goal is different, a similar factorization is obtained. To understand the relation between reducing the effect of outliers and maximizing the sparseness of the residual consider the following. Maximizing the sparseness of the residual means that for a t-dimensional approximation of the rows of **R**, as many rows as possible are *exactly* representable in this approximation. The rows that do not exactly fit in this t-dimensional space are in fact outliers. In other words, the non-zero elements in the residual identify outliers.

To solve the minimization in Equation (6), and to compute **W** and **F** we use an alternating iterative approach similar to [Lawrence et al. 2006]. At each iteration step either **W** or **F** is kept constant while the other is optimized. Solving Equation (6) with one of the factors fixed, is equal to solving a constraint ℓ_1-minimization problem. Furthermore, note that for a variable **W** and a fixed **F**, that each row of **W** can be computed independently from the other rows. We can therefore solve Equation (6) for each row U_i of **W** independently:

$$\min \|\mathbf{E}_i\|_1, \;\; s.t. \;\; \mathbf{R}_i = \mathbf{W}_i\mathbf{F} + \mathbf{E}_i, \tag{7}$$

where $\mathbf{E}_i$ is the i-th row of the residual **E**, and $\mathbf{R}_i$ is the i-th row of **R**. A similar formula exists for the columns of **F** in case **W** is fixed and **F** is variable.

[Ke and Kanade 2005] identify two possible ways of solving this constraint ℓ_1-minimization problem: using linear programming, and using quadratic programming. They opt for using a quadratic programming solution. However, a detailed comparison of both methods is missing. We will therefore discuss and compare both solutions in detail.

Linear Programming. Constraint ℓ_1-minimization problems can be efficiently solved using Linear Programming. A number of efficient, general, algorithms have been developed to solve these kinds of problems (e.g., the simplex algorithm and interior point methods), and a large selection of software libraries are available that implement these methods are available. We will therefore describe the *linear program* without assuming a specific algorithm or library.

More specifically, we adapt the linear program described in [Abdelmalek 1980] to fit our needs. In the so-called standard form of linear programming [Dantzig 1963; Gill et al. 1991] the following constraint optimization is solved in terms of a variable $\mathbf{x} \in \Re^z$:

$$\min \mathbf{c}^T\mathbf{x} \;\; s.t. \;\; \mathbf{Ax} = \mathbf{b}, \;\; \mathbf{x} \geq 0, \tag{8}$$

where $\mathbf{c}^T\mathbf{x}$ is the objective function, $\mathbf{Ax} = \mathbf{b}$ a collection of equality constraints, and $\mathbf{x} \geq 0$ a set of bounds. Reformulating Equation (7) now yields the following translations:

$$z \Leftrightarrow t + 2n \; ; \quad \mathbf{x} \Leftrightarrow (\mathbf{W}_i, \mathbf{k}, \mathbf{l}) \; ; \quad \mathbf{c} \Leftrightarrow (\mathbf{0}, \mathbf{1}, \mathbf{1}) \; ;$$
$$\mathbf{A} \Leftrightarrow (\mathbf{F}, +\mathbf{I}, -\mathbf{I}) \; ; \quad \mathbf{b} \Leftrightarrow \mathbf{R}_i,$$

where z is the dimension of the vector of variables **x**. Note that $\mathbf{W}_i$ and $\mathbf{R}_i$ have a length of t and n respectively. **k** and **l** represent the positive and negative part of each element of (a row in) the residual matrix (i.e., the residual row $\mathbf{E}_i = \mathbf{k} - \mathbf{l}$), and thus the ℓ_1-norm of the residual is the sum of both these parts $\mathbf{k} + \mathbf{l}$ (hence the **1**'s in **c**). Both vectors have a length of n. Because we are searching for the best **W** to minimize the residual, we introduce the shadow variables $\mathbf{W}_i$ in **x**. These shadow variables do not contribute to the error functional $\mathbf{c}^T\mathbf{x}$, and thus they receive a zero weight in **c**. Finally, we need to enforce that $\mathbf{R}_i - \mathbf{W}_i\mathbf{F} = \mathbf{k} - \mathbf{l}$, which is achieved by the particular forms of **R** $(= \mathbf{W}_i\mathbf{F} + \mathbf{k} - \mathbf{l})$ and **b** $(= \mathbf{R}_i)$. A similar linear program can be constructed for the columns of **F**. Note that if a positive residual is desired, the term **l** can be dropped.

We implemented the above algorithm using the GNU Linear Programming Kit (`http://www.gnu.org/software/glpk/`). We initialize either **W** or **F** to random values, and optimize the

other according to the linear program. Next, we switch the roles of **W** and **F**, and solve the other factor. We repeat this until the ℓ_1-error on the residual does not decrease anymore. Convergence is usually reached in less than 10 iterations. Note that it is possible to solve for the matrix **W** using a single linear program, instead of a program for each row $\mathbf{W}_i$ as shown above. However, linear programming has an exponential time complexity, and thus keeping the program as small as possible is essential for obtaining a solution in a reasonable time. Furthermore, the algorithm is not guaranteed to converge to a global minimum, but in general when the residual can be made reasonably sparse, the global minimum is likely to be found.

Quadratic Programming. Constraint linear problems can be solved using quadratic programming by using the Huber M-estimator [Li and Swetits 1998; Huber 1991] instead of the ℓ_1-norm:

$$\rho(t) = \begin{cases} \frac{1}{2}t^2 & \text{if } |t| \leq \gamma \\ \gamma|t| - \frac{1}{2}\gamma^2 & \text{if } |t| > \gamma \end{cases} \tag{9}$$

where γ is some positive scalar. $\rho(t)$ approximates an ℓ_1-norm when $\gamma \to 0$. This estimator can be written as a minimum of a convex quadratic program [Mangasarian and Musicant 2000]:

$$\rho(t) = \min_x \frac{1}{2}x^2 + \gamma|t - x|. \tag{10}$$

By replacing the ℓ_1-norm in Equation (7) we get:

$$\min \rho(\mathbf{E}_i), \;\; s.t. \;\; \mathbf{R}_i = \mathbf{W}_i\mathbf{F} + \mathbf{E}_i. \tag{11}$$

This can be rewritten as [Mangasarian and Musicant 2000]:

$$\min \frac{1}{2}\|\mathbf{E}_i\|_2^2 + \gamma\|\mathbf{W}_i\mathbf{F} - \mathbf{R}_i - \mathbf{E}_i\|_1, \tag{12}$$

This problem can be efficiently solved using quadratic programming. The standard form of a quadratic program is given by:

$$\min \frac{1}{2}\mathbf{x}^T\mathbf{C}\mathbf{x} + \mathbf{c}_0^T\mathbf{x}, \;\; s.t. \;\; \mathbf{A}\mathbf{x} = \mathbf{b}, \;\; \mathbf{D}\mathbf{x} \leq \mathbf{t}, \tag{13}$$

where $\mathbf{x} \in \Re^z$ is the minimization variable, **C** and $\mathbf{c}_0$ determine the cost function, **A** and **b** are a set of equality constraints, and **D** and **t** are a set of inequality constraints. Equation (12) now yields the following translations:

$$\begin{gathered} z \Leftrightarrow t + 2n \; ; \quad \mathbf{x} \Leftrightarrow (\mathbf{W}_i, \mathbf{E}_i, \mathbf{k}) \; ; \\ \mathbf{C} \Leftrightarrow \mathbf{K}^T\mathbf{K}, \;\; \mathbf{K} = (0, \mathbf{I}, 0) \; ; \quad \mathbf{c}_0 \Leftrightarrow (\mathbf{0}, \mathbf{0}, \gamma) \; ; \\ \mathbf{D} \Leftrightarrow \begin{pmatrix} -\mathbf{F} & +\mathbf{I} & +\mathbf{I} \\ +\mathbf{F} & -\mathbf{I} & +\mathbf{I} \end{pmatrix} \; ; \quad \mathbf{t} \Leftrightarrow (+\mathbf{E}_i, -\mathbf{E}_i), \end{gathered}$$

where z is the dimension of the vector of variables **x**. Note that $\mathbf{W}_i$ and $\mathbf{R}_i$ have a length of t and n respectively. **k** represent the ℓ_1-norm on $\mathbf{W}_i\mathbf{F} - \mathbf{R}_i$ (i.e., the second term in equation (12)), and has a length of n. According to equation (12), the quadratic term only depends on the residual $\mathbf{E}_i$. This is ensured by the definition of **C**. The linear term in equation (12) only depends on **k** weighted by γ. This is mirrored by the definition of $\mathbf{c}_0$. Note that similarly as in the linear program a shadow variable, $\mathbf{W}_i$, is introduced with a similar goal. Due to the introduction of **k**, no equality constraints (**A**, and **b**) are necessary. The inequality constraints ensure that when $\mathbf{k} \to 0$ that $\mathbf{R}_i \to \mathbf{W}_i\mathbf{F} + \mathbf{E}_i$. The advantage of formulating the program like this, instead of following a similar approach as the linear program, is that the dimensionality z is less. Because the computation time is directly proportional to z, this yields a faster quadratic program.

We implemented the above algorithm using QuadProg++, however any other library can also be used. As before, we initialize either **W** or **F** to random values, and optimize the other according to the linear program. Next, we switch the roles of **W** and **F**, and solve the other factor.

Comparison. We compare both the linear and quadratic program in terms of accuracy and computational cost.

To test accuracy, we factorize a matrix $\mathbf{R} = \mathbf{W}\mathbf{F} + \mathbf{E}$, that is generated by selecting random vectors **W** and **F** (the number of terms is preselected), and writing at random position in **E** a random value. By controlling the number of randomly written values in **E**, we can control the sparseness of the expected residual after factorization of **R** in t terms. This allows us to compare the accuracy of both methods. To compare the computational cost of both algorithms, we generate matrices in a similar manner, but we now keep the sparseness fixed to 10% percent, and increase the size of both **W** and **F** (both have an equal size and thus **R** is a square matrix). We performed this test for a different number of terms.

For both tests, the linear program outperforms the quadratic program in terms of speed and accuracy, expect for small matrices (i.e., less than 100 elements) where the computational cost and accuracy is similar for both methods. Furthermore, the linear program scales better in terms of problem size and sparseness than the quadratic program. This result is surprising, because quadratic programming is the preferred solution of [Ke and Kanade 2005]. Therefore, we also tested the same quadratic and linear program using different libraries (i.e., in Octave and Matlab) to ensure that the obtained results are not skewed by library specific issues. However, similar results were obtained. In the remainder of this paper, we will therefore perform sparse residual factorization using linear programming.

3.3 Clustering

To create the global approximation we use a clustered factorization approach. The general idea is to alternate between assigning pixels to the clusters that yield the best approximation, and finding the best approximation for a given cluster. We bootstrap the clustered factorization algorithm by assigning each pixel to a random cluster. Next, we repeat the following two steps until convergence is reached:

1. **Compute a factorization per cluster.** We apply one of the factorization algorithms of Subsection 3.2. This yields for each cluster k and pixels $q \in \{p : c(p) = k\}$ an approximation $\mathbf{R}_{q,i} \approx \sum_t \mathbf{W}_{q,t}\mathbf{F}^k_{t,i}$.

2. **Reassign each pixel to the optimal cluster.** For each pixel p we reassign it to the cluster that yields the best approximation given the basis vectors $H^k_{t,i}$:

$$c(p) = \arg\min_k \|\mathbf{E}^k_{p,i}\|_1, \tag{14}$$

   where $\mathbf{E}^k$ is the minimal residual (in an ℓ_1 sense) obtained by solving $\mathbf{R}_{p,i} \approx \sum_t \mathbf{W}_{p,t}\mathbf{F}^k_{t,i}$ for **W** (i.e., keeping $\mathbf{F}^k$ fixed).

Although this algorithm gives a good clustered factorized approximation of the reflectance field, it requires a significant amount of computation time. As noted before, the time-complexity of the linear program used to compute the factorization is exponentially proportional to the size of the matrix **R**. This size is determined by the number of pixels in the cluster (i.e., the height of the reflectance matrix **R**), and the length of each reflectance function (i.e., the number of sampled light source positions).

In order to speed up the clustered factorization, we first compute the clusters on a low resolution (in camera pixels) reflectance field. This reduces the height of the reflectance field matrix **R**. The idea is that neighboring clustering will share many properties, and therefore, the important terms in their factorization will be very similar. This scheme is similar in spirit to randomly selecting rows in large matrices to factorize basis vectors. However, in this case a priori knowledge of inter-pixel relations (i.e., 2D spatial neighborhoods) are exploited, yielding a more coherent subsampling. Once a low resolution clustered factorization is obtained, we assign each pixel in the full resolution reflectance field to the most optimal cluster from the low resolution reflectance field. Finally, an optional final sparse factorization can be performed to further fine-tune the basis functions $\mathbf{F}^k$ per cluster.

Assigning a cluster to each pixel of the full-resolution reflectance field is still very time consuming, because the optimal weights (in an ℓ_1-sense) have to be computed for each cluster. Ideally we would like to reduce the number of clusters that need to be verified by excluding most of them a priori. This is achieved by only considering clusters assigned to a small neighborhood around the pixel's projected position in the low resolution version. The idea is that there is a reasonable amount of spatial coherence between the low and full-resolution versions of the reflectance field.

3.4 Compressing the Residual

The obtained residual from the previous factorization method is sparse, making it well suited for further compression. Due to approximation errors, a large number of the elements in the residual are not exactly zero, but near zero. Therefore, as a first step, we set all elements below some small threshold to zero. To further control the compression ratio, we keep the n largest elements for each pixel's residual.

When storage is of less concern, but rendering quality is of prime interest, we select a large value for n. During rendering, we dynamically choose the number of residual elements ($< n$) depending on the desired frame rate or rendering quality.

4 Efficient Relighting

Techniques to render dimensionality-reduced reflectance functions have been explored extensively [Sloan et al. 2002; Sloan et al. 2003; Ng et al. 2004]. Similar to [Sloan et al. 2003], we precompute with each incident illumination change, for each cluster k the inner products of the basis functions $\mathbf{F}^k$ and the incident illumination $\mathbf{l}$, resulting in a vector of length t (the number of terms). During rendering, a weighted sum per pixel is computed with these t precomputed values $\mathbf{F}^k\mathbf{l}$ and the pixel's weights $\mathbf{W}$. Finally, the inner product of the compressed residual matrix $\mathbf{E}$ and the incident illumination $\mathbf{l}$ is added:

$$\mathbf{R}_p = \sum_{t}^{terms} \left(\mathbf{W}_{p,t} \left(\mathbf{F}_t^{c(p)} \mathbf{l} \right) \right) + \mathbf{E}_p \mathbf{l}. \tag{15}$$

The incident illumination vector $\mathbf{l}$ is extracted at runtime from a high resolution environment map using an angular Voronoi diagram based sampling method [Masselus et al. 2002]. Implementing the above relighting system on a CPU-only system is straightforward. A hardware accelerated GPU implementation is a little bit more complex. In the following subsections, we detail some GPU specific implementation aspects.

4.1 Global Approximation Relighting Computation

A number of publications have studied matrix-matrix multiplication, and matrix-vector multiplication implementations on GPUs. Fatahalian *et al.* [Fatahalian et al. 2004] did an in-depth-study on the efficiency of various matrix multiplication algorithms on a variety of graphics hardware. Changhao *et al.* [Jiang and Snir 2005] developed automatic performance tuning strategies. In our application, we opt for using a 2x2 scheme, called NV Single [Fatahalian et al. 2004], because of its ease of implementation and computational efficiency.

The per cluster multiplication of $\mathbf{F}^k\mathbf{l}$ is a straightforward matrix-vector multiplication. We store 2x2 blocks of the matrix $\mathbf{F}^k$ in four component texels according to NV Single.

To compute $\mathbf{W}(\mathbf{F}^k\mathbf{l})$, we note that each row of matrix $\mathbf{W}$ is potentially assigned to a different cluster. It is therefore quite reasonable to access $\mathbf{W}$ per row unit. Consequently, each row of the matrix $\mathbf{W}$ is stored and packed in four component texels. Due to this packing, a single fragment program can obtain its pixel's intensity value by a simple inner product computation.

4.2 Sparse Residual Multiplication

Multiplying the incident illumination vector with the sparse residual matrix is the most costly part in our relighting algorithm. It is therefore very important to do this as efficiently as possible. Similar problems have been studied extensively [Krüger and Westermann 2003; Bolz et al. 2003]. A compressed sparse row format is commonly used (as described in [Mellor-Crummey and Garvin 2004]). To compute the sparse matrix-vector multiplication in parallel fragment programs, we store each of the non-zero coefficients and an identification index, packed into a single texture. Next, we generate an indirect texture that for each pixel, that refers to the first non-zero residual coefficient in the previously defined texture. Computing the inner product at runtime is now relatively straightforward.

5 Results and Discussions

We verified our compression method on a number of publicly available reflectance fields (i.e., `http://gl.ict.usc.edu/Data/LightStage/`). In the discussion of the clustered sparse residual factorization algorithm of Section 3, we treated the reflectance functions as monochromatic. However, real reflectance fields usually consist of three color channels. In our implementation we treat each color channels of a reflectance function as an independent monochromatic reflectance function. Thus, each reflectance function's color channel will have an associated cluster index.

Furthermore, to ensure that each cluster does not contain too many or too few reflectance functions to achieve a successful factorization, a minimum and maximum cluster size is enforced. Clusters that are too small are merged to the second most optimal cluster. Clusters that are too large are split into two equal-sized clusters (increasing the number of clusters).

The computation costs of our method are asymmetric: decompression costs are very low, while compression requires significant effort. On average the compression timings ranged from 24 to 72 hours depending on the resolutions, the number of clusters and the number of terms.

Visual Quality Analysis. Figure 2 shows two scenes containing a helmet and a plant, relit with the St. Peter's Basilica and Grace

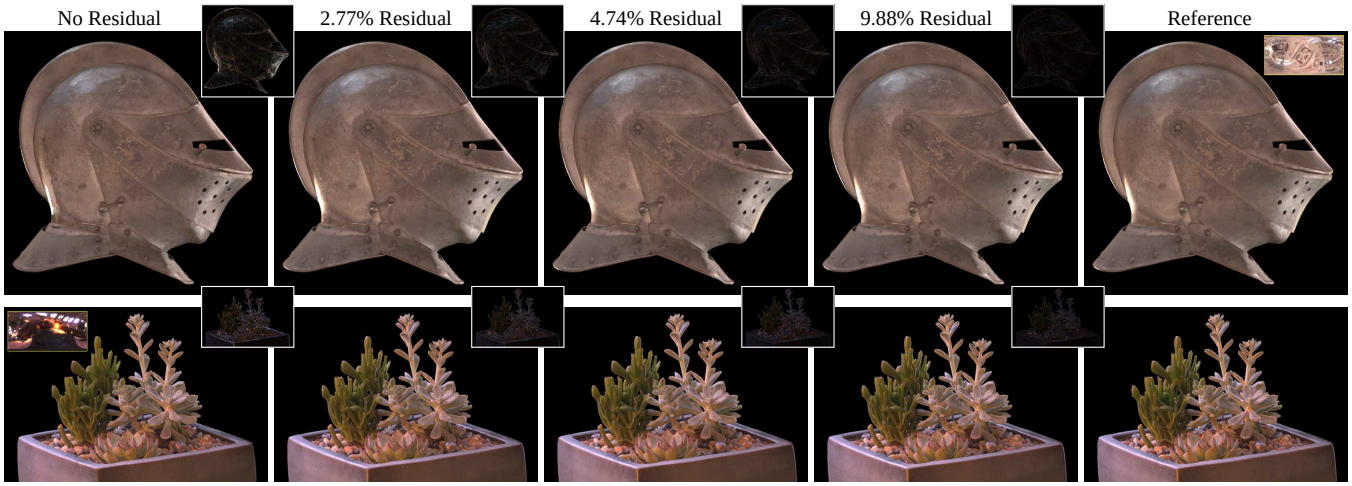

Figure 2: *Illustrating the visual effect on relit images of the helmet and plant dataset with increasing number of residual coefficients. Both datasets are factorized using* 128 *clusters, and* 4 *terms. The number of residual terms used (from left to right) are:* 0%, 2.77%, 4.74%, 9.88%, *and* 100%. *The differences with the reference image (i.e.,* 100% *residual) are shown at the right top of each image.*

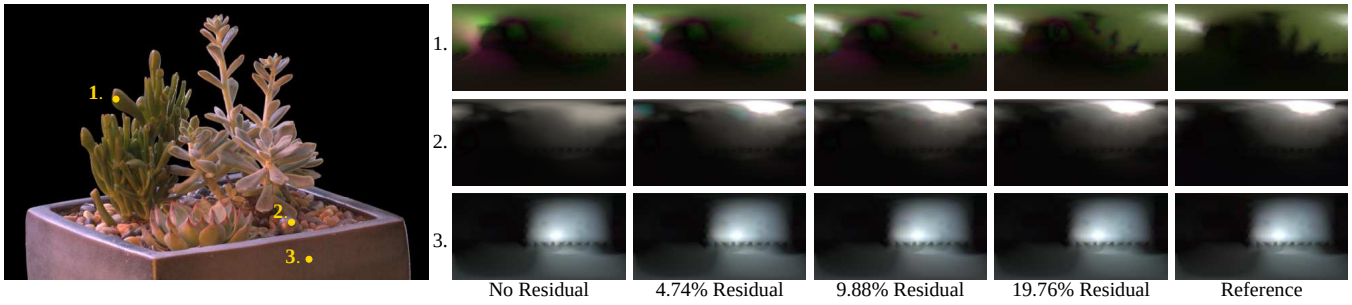

Figure 3: *The effect of increasing the number of residual terms on three selected reflectance functions. Marked on the relit of the image of the plant dataset (*32 *clusters, and* 4 *terms) on the left are the three pixels for which the reflectance functions are shown on the right. Each reflectance function is shown using* 0%, 4.74%, 9.88%, 19.76%, *and* 100% *of the residual.*

Cathedral environment maps respectively. We compactly represented these reflectance fields using approximately 128 clusters and 4 terms. Each scene is relit and shown with different numbers of residual elements included. The leftmost image shows the scene with no residual added (i.e., reconstructed using $\mathbf{W}$ and $\mathbf{F}^k$ only), while the rightmost shows a reference image (i.e., using the full residual $\mathbf{E}$). The differences of each image with respect to the reference image are shown in the details in the right top of each relit image. The difference between the relit image without any residual and the reference image is significant; in particular the visually important highlights are missing. However, when adding even a few residual elements (3% – 5%), the visual difference decreases dramatically. At 10% of the residual, the difference is difficult to distinguish from measurement noise.

Figure 3 illustrates the effect of increasing the number of residual elements on a number of selected reflectance functions of the plant scene (32 clusters, and 4 terms). The reflectance functions are shown respectively with 0%, 4.74%, 9.88%, 19.76%, and 100% of the residual. As can be seen, only a few large elements in the residual contribute significantly to the reflectance function. Note how specular highlights are added to the reflectance functions using only a few residual coefficients. Furthermore, note that we used a very low number of clusters and terms to better illustrate these effects.

Quantitative Error Analysis. To further investigate the effect of the number of clusters, terms, and residuals, we plot the error as a function of these parameters. There are a number of error metrics possible. In [Wong and Leung 2003; Wang et al. 2004b; Wang et al. 2004a; Ho et al. 2005] the average *peak signal to noise ratio* (PSNR) is used to characterize the error. This error metric is commonly used in the compression community. However, this metric is not suited to characterize the error on compressed reflectance fields. First, the notion of *peak signal* is not well defined when using multiple high dynamic range reflectance functions, since each function can have a different peak. Furthermore, a specular peak in a reflectance function will mask much of the error in the diffuse component. An alternative to directly computing the average PSNR on the reflectance field is to compute the average PSNR on a large selection of generated relit images. However, this approach also suffers from the same problems as noted before. Masselus *et al.* [Masselus et al. 2004] use two different metrics to characterize the error on the reconstruction and on the compression of reflectance functions. For the compression they use the Sobolev H^1-norm, and for the reconstruction they compute the average relative error per reflectance function. To better differentiate between the effects of the different reflectance types (i.e., occluded vs. unoccluded, and specular vs. diffuse) they compute the average over pixels with a similar reflectance type. We will follow the latter approach, and compute the

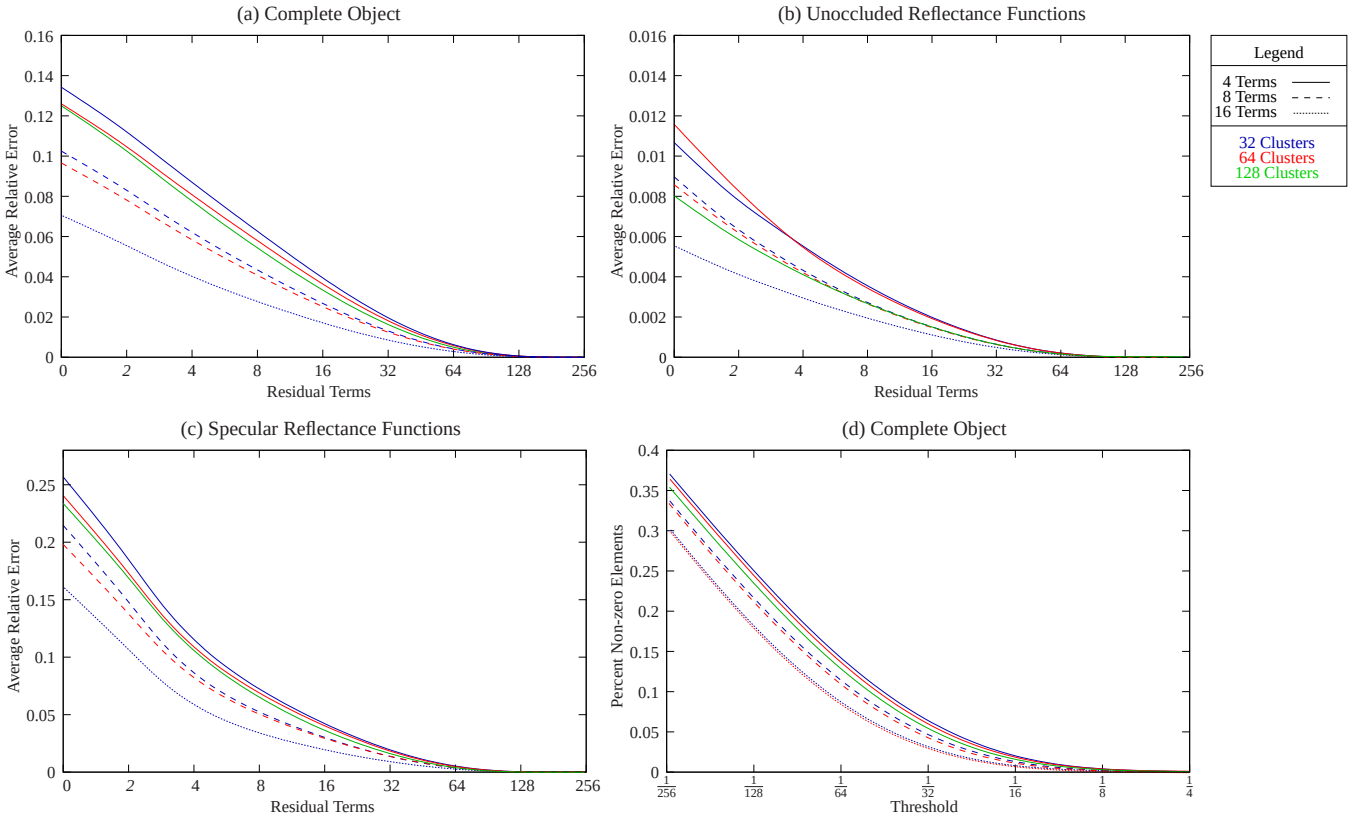

Figure 4: *Error and sparseness plots of the plant dataset. The colors indicate the number of clusters. The line pattern denotes the number of terms. (a-c) The average relative error of all reflectance functions of the object, unoccluded reflectance functions, and specular reflectance functions. (d) The average sparseness with respect to a noise threshold on all the reflectance functions of the object.*

relative error as:

$$error = \frac{1}{n}\sum_{p}\left(\frac{\sum_i \left(\mathbf{R}_{p,i} - \widetilde{\mathbf{R}}_{p,i}\right)^2}{\sum_i \mathbf{R}_{p,i}^2}\right), \tag{16}$$

where $\widetilde{\mathbf{R}}_{p,i}$ is the compressed approximation of the reflectance field $\mathbf{R}_{p,i}$. Figures 4 (a-c) show the average relative error plots for the plant scene for *unoccluded*, *specular* and *all the object's* reflectance functions. Note that the horizontal axis uses a logarithmic scale.

A first observation that can be made from these error plots is that the influence on the error of the number of clusters is less than that of the number of terms. However, in terms of the compression ratio, increasing the number of clusters is far less expensive (i.e., you only need to store the additional basis functions H^k once), while increasing the number of terms is much more expensive because you need to store additional weights $\mathbf{W}$ for *each* pixel in the reflectance field.

A second observation is that for unoccluded diffuse surfaces (Figure 4 (b)) the error is already low without including any residual terms. Furthermore, for reflectance functions with a predominantly specular behavior, the error is much higher, and drops very fast with increasing the number of residual elements. The reason for this is that the residual in the specular case consists largely of localized effects that change rapidly between pixels. The low number of clusters is unable to capture these differences, unless the number of clusters is increased significantly (to approximately the number of pixels).

A third observation is that it is generally better to store additional residual terms, then to increase the number of terms. For example, when looking at Figure 4 (c), we see that with 32 clusters, 16 terms, and no residual, the error is approximately 16%. The error on 32 clusters, 4 terms, and 12 residual terms (that has approximately the same storage cost per pixel) is just above 6%, and the error on 32 clusters, 8 terms, and 8 residual terms is just below 6%. The reason for this is that the first few residual terms basically encode specular highlights and other high frequency phenomena. These high frequency features change significantly (spatially) among different reflectance functions in the same cluster, and are, therefore, difficult to encode with only a few basis functions.

A final observation is that from a certain point on, it becomes more interesting to add additional terms instead of extra residual elements. For example, in the case of the specular reflectance functions (Figure 4 (c)), the error for 32 cluster, 8 terms, and 8 residual elements is about equal to the error of 32 clusters, 4 terms, and 16 residual elements (this requires more data to be stored). In general the error-plot decreases more slowly as the number of residual elements increases, and thus at a certain point, the gain by jumping to a different error-plot (with more terms) provides an advantage. The reason for this is that the residual terms of smaller magnitude encode mainly low frequency differences, and these can be represented more efficiently by additional terms.

In Figure 4 (d) the average sparseness of the plant scene in terms of a noise threshold is shown. All elements with a magnitude less than the noise threshold are considered to be noise and set to zero. Even with a very low threshold of $\frac{1}{256}$, only about $\frac{1}{3}$ of the coefficients are non-zero. Although this is a lossy compression, it is nearly lossless

in that our threshold is below the camera noise threshold. Note that the sparseness is almost independent of the number of clusters and only slightly dependent on the number of terms. This is consistent with the first observation made with respect to the average relative error.

Comparative Analysis. To further investigate the performance of our method, we compare it to clustered PCA [Sloan et al. 2003]. Furthermore, we also investigate the effect of including residual terms in a clustered PCA approximation.

Figure 5 (left) shows the ratio of the relative average error of our method versus clustered PCA. We vary for a fixed number of clusters and terms, the amount residual terms for both methods. While clustered PCA has not been developed for including residual terms, it clearly shows the strength of our method. In general, our methods starts to outperform around 16 additional residual terms. As expected, without adding any additional residual terms, clustered PCA is superior since it specifically minimizes the error in this case. Additionally, we also show the same reflectance function with increasing number of residual terms as in figure 3 for clustered PCA.

If beforehand the maximum number of required terms is known, then a different comparison with a clustered PCA approach can be made. In this case, a clustered-PCA approximation with the maximum number of terms can be computed. During visualization, a subset of largest vectors of each cluster with the desired size (less than the maximum number of terms) can be selected for visualization. This has similar runtime (CPU and memory) requirements compared to the proposed clustered sparse matrix factorization approach where the residual terms + sparse terms equal the number of PCA terms. We computed the average relative error for a clustered PCA approximation of the plant scene computed with a maximum of 64 terms, and using a subset of 16, versus a cluster sparse matrix factorization approximation with 4, 8, and 16 terms, with respectively 12, 8 and 0 residual terms. Both are computed with 64 clusters. Figure 6 shows the relative error images. Clustered PCA significantly outperforms, in terms of average error, our method when no residual terms are included. However, when adding only a few residual terms, our method clearly outperforms clustered PCA.

To better understand the difference for individual reflectance functions, we generated the false color images (figure 6 bottom) for these cases. A blue color indicates a lower error for the PCA approach, while a red color indicates a lower error for the sparse matrix factorization approach. The intensity of the false color denotes the magnitude of the difference in error between both approaches. As can be seen, there are regions where each method excels. In general we found that a clustering based on a sparse matrix factorization yields better results for specular materials, and pure diffuse materials. In these cases the residual encodes mostly very localized differences to the factorized approximation. However, for glossy materials a clustering based on PCA outperform our method. In these cases, the differences with the cluster approximation covers a larger solid angle. These differences cannot be efficiently encoded by a few large coefficients.

Visualization Performance. Table 1 lists for all scenes the GPU rendering speed and error as a function of the amount of residual that is added to the approximation. All measurements are performed using an nVIDIA GeForce 8800GTX with 768Mb onboard memory. The camera resolutions of the reflectance fields are 640×422 for the plant dataset, 640×640 for the helmet dataset, 512×512 for the kneeling knight dataset, and 640×854 for the fighting knight dataset. The lighting resolution is 253 lighting directions. As can be seen, the number of clusters has very little effect on rendering speed. Increasing the number of terms though, reduces the frame rate. Nevertheless, even with 30% of the full residual and errors well below 1%, we are still able to achieve 50fps. Our system is also suited for CPU interactive rendering (around 8fps) with moderate residual levels ($\approx 3-5\%$). The videos in the supplemental material contain animated sequences of these datasets rendered using our GPU implementation.

Discussion. The total compression ratio is easily computed as follows: $ratio \approx \frac{(clusters \times terms)}{n} + \frac{(terms+residual)}{m}$. The first term can usually be neglected due to the large size of n (i.e., the number of pixels). For example, for the plant dataset, using 128 clusters, 4 terms, and 12 residual terms ($\approx 5\%$) a ratio of $\frac{16}{253} \approx 6.3\%$ is obtained. These compression ratios are less than some of the ratios reported in previous work. However, almost all of these methods work on much higher resolution (in lighting directions) datasets, making it easier to achieve higher compression ratios due to the fact that there is more data to correlate. Furthermore, it should be noted that not all methods are able to represent all-frequency reflectance data accurately. Additionally, in many cases the exact error is not known, and thus a qualitative comparison is not possible.

Although we only tested our method on reflectance fields of moderate incident lighting resolution, we expect that the compression ratios will improve when using higher incident resolution reflectance fields. We expect that the efficiency of the factored representation is sub-linearly dependent on the incident illumination resolution, since it mainly represents the low and mid frequency content, that is largely resolution independent. The compressed residual, on the other hand, represent mostly high frequency content. This high frequency content is largely undersampled in low and mid resolution reflectance fields, and we expect that these localized high frequency features will also scale sub-linearly in size.

Finally, we note that it is straightforward to extend our method to obtain a residual that is sparse in a different basis, for example a wavelet basis. Wavelets are well known to exploit spatial coherencies and improve compression ratios. To obtain a sparse residual in any basis, we first express each row (i.e., reflectance function) in that basis. Next, we perform the factorization as described in Section 3.2. The obtained result will be that the basis functions $\mathbf{F}_k$ and the residual are expressed in this basis. We determined empirically that using a Haar wavelet basis improved sparseness by 3% to 5%. However, this resulted in our case in a *reduction* in compression ratios for a given level of quality. Because wavelets are a hierarchical basis, the opportunity to leave out wavelet coefficients increases at each additional wavelet level. Low frequency wavelets are usually significant and cannot be left out easily. Thus, due to the fact that in our case the incident illumination resolution is low, we do not gain anything. We expect that when higher resolution reflectance fields are used that wavelets will provide a significant advantage. An additional advantage of a wavelet represented residual is that localized differences over more then one lighting direction can be more efficiently represented.

6 Conclusions

In this paper we presented a novel compression method for fixed viewpoint reflectance fields. This method is based on a clustered sparse residual factorization. The newly developed sparse residual factorization method ensures that the residual of the original reflectance function and the factorized approximation contains as many as possible (near) zero elements. The obtained residual is therefore very easy to represent sparsely, and enables to us trade off storage requirements versus accuracy. Our method shows good compression ratios for both nearly lossless and lossy compression. Additionally, we show that our method is well suited for real-time hardware assisted rendering.

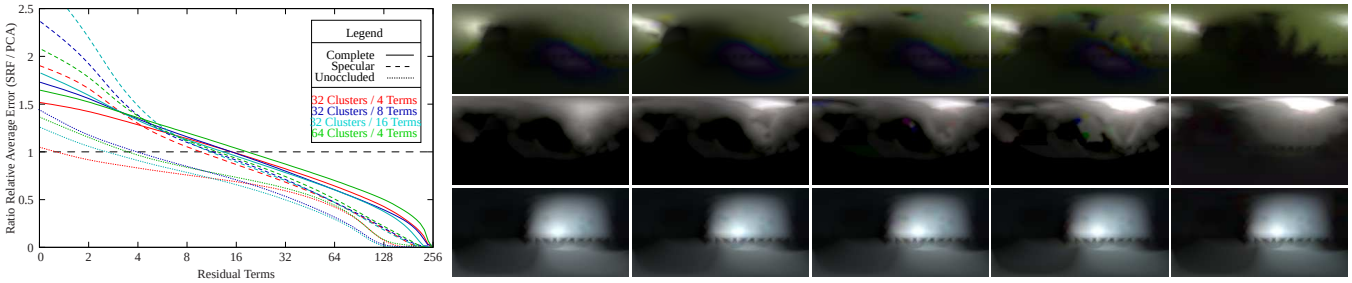

Figure 5: *A comparison of relative average error on our method versus clustered PCA for a varying number of residual terms.*

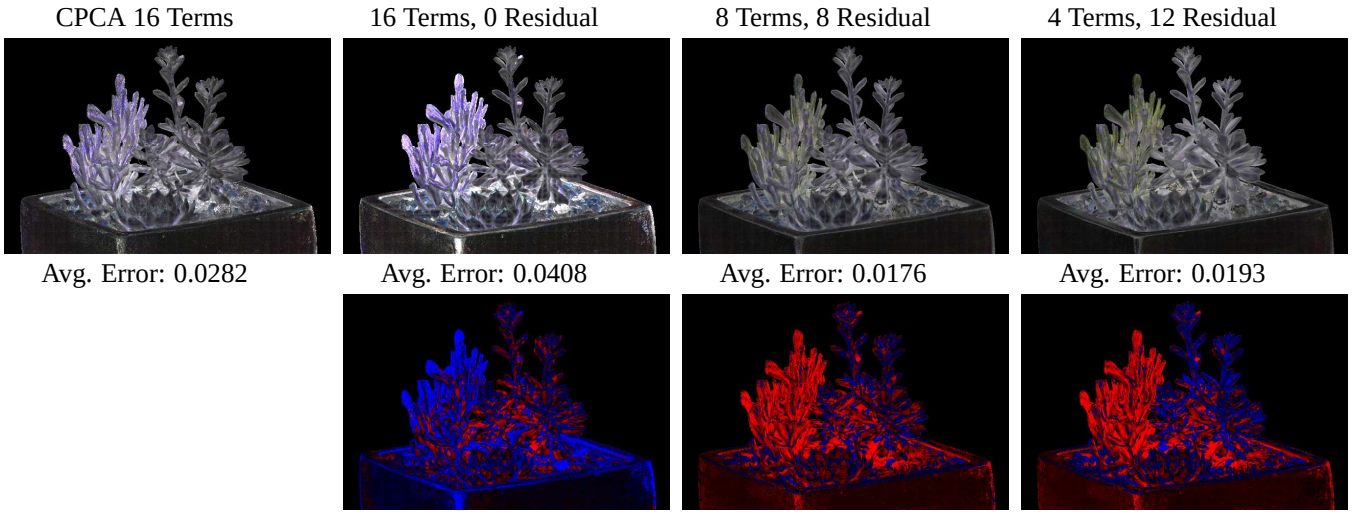

Figure 6: *A comparison of clustered PCA (64 terms, 64 clusters) using a subset of the 16 most important factors, compared to clustered sparse matrix factorization using 4, 8, and 16 terms and respectively 12, 8, and 0 residual terms. The bottom row shows the difference between the average relative error of the clustered PCA approximation versus the clustered sparse matrix factorization approximation. A red color indicates a larger error on the clustered PCA approximation, while a blue color indicates a larger error on the clustered sparse matrix factorization approximation.*

For future work we would like to investigate supplemental compression methods similar to [Wong and Leung 2003], by compressing the obtained weights and residual elements in the camera-domain using wavelet or DCT compression methods. This will most likely reduce storage cost even further. Second, alternative algorithms for computing the factorization instead of linear and quadratic programming should be investigated. Currently we only looked at general purpose algorithms, such as linear and quadratic programming, for factorization. These general purpose algorithms are very versatile, but the computational costs are considerable. Specialized algorithm will most likely yield a more optimal factorization algorithm. Finally, we would like to investigate the application of our sparse matrix factorization method to other problems.

Acknowledgments

We would like to thank Tomas Pereira, Tim Hawkins and Saskia Mordijck for their valuable input, and Bill Swartout, Randolph Hall, and Max Nikias for their support and assistance with this work. This work was sponsored by the University of Southern California Office of the Provost and the U.S. Army Research, Development, and Engineering Command (RDECOM). The content of the information does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

References

ABDELMALEK, N. N. 1980. Algorithm 551: A Fortran subroutine for the ℓ_1 solution of overdetermined systems of linear equations [F4]. *ACM Transactions on Mathematical Software 6*, 2, 228–230.

BOLZ, J., FARMER, I., GRINSPUN, E., AND SCHRÖDER, P. 2003. Sparse matrix solvers on the GPU: Conjugate gradients and multigrid. *ACM Transactions on Graphics 22*, 3 (July), 917–924.

CHEN, S. S., DONOHO, D. L., AND SAUNDERS, M. A. 1999. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing 20*, 1, 33–61.

DANTZIG, G. 1963. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ.

DEBEVEC, P., HAWKINS, T., TCHOU, C., DUIKER, H.-P., SAROKIN, W., AND SAGAR, M. 2000. Acquiring the reflectance field of a human face. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 145–156.

EINARSSON, P., CHABERT, C.-F., JONES, A., MA, W.-C., LAMOND, B., HAWKINS, T., BOLAS, M., SYLWAN, S., AND DE-

Dataset	No. Clus.	No. Terms	0%		3%		5%		10%		20%		30%	
			fps	err	fps	err	fps	err	fps	err	fps	err	fps	err
Plant	32	4	310.7	0.188	256.0	0.069	188.8	0.049	123.3	0.026	75.3	0.010	53.2	0.004
Plant	32	8	299.6	0.151	244.4	0.050	182.3	0.034	121.8	0.017	73.7	0.006	52.5	0.002
Plant	32	16	259.5	0.112	206.4	0.031	168.4	0.021	115.1	0.011	71.3	0.004	50.2	0.002
Plant	64	4	307.7	0.178	256.1	0.064	188.9	0.046	123.5	0.024	75.2	0.009	53.3	0.004
Plant	64	8	299.1	0.144	244.3	0.046	182.3	0.032	121.2	0.017	73.8	0.005	52.5	0.003
Plant	64	16	259.4	0.106	205.8	0.030	166.7	0.020	114.1	0.011	70.9	0.004	51.0	0.002
Plant	128	4	307.7	0.167	256.1	0.059	189.0	0.041	123.4	0.021	75.3	0.008	53.2	0.003
Fight. Knight	128	4	285.7	0.173	216.9	0.045	178.3	0.030	125.5	0.015	78.0	0.005	57.2	0.002
Kneel. Knight	128	4	310.7	0.148	284.4	0.064	223.8	0.048	157.5	0.028	100.9	0.011	76.8	0.005
Helmet (side)	128	4	306.2	0.064	174.4	0.010	131.1	0.007	80.2	0.003	45.7	0.001	31.6	0.000

Table 1: *A summary of the rendering rates and average relative errors of the different datasets in function of the amount of residual used.*

BEVEC, P. 2006. Relighting human locomotion with flowed reflectance fields. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, 183–194.

FATAHALIAN, K., SUGERMAN, J., AND HANRAHAN, P. 2004. Understanding the efficiency of GPU algorithms for matrix-matrix multiplication. In *Graphics Hardware 2004*, 133–138.

FURUKAWA, R., KAWASAKI, H., IKEUCHI, K., AND SAKAUCHI, M. 2002. Appearance based object modeling using texture database: Acquisition compression and rendering. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, 257–266.

GILL, P., MURRAY, W., AND WRIGHT, M. 1991. *Numerical Linear Algebra and Optimization*. Addison Wesley, Redwood City, CA.

HO, P., WONG, T., AND LEUNG, C. 2005. Compressing the illumination-adjustable images with principal component analysis. *IEEE Transactions on Circuits and Systems for Video Technology 15*, 3 (Mar.), 355–364.

HUBER, P. 1991. *Robust Statictics*. John Wiley, New York.

JIANG, C., AND SNIR, M. 2005. Automatic tuning matrix multiplication performance on graphics hardware. In *PACT '05: Proceedings of the 14th International Conference on Parallel Architectures and Compilation Techniques*, 185–196.

KAUTZ, J., AND MCCOOL, M. D. 1999. Interactive rendering with arbitrary BRDFs using separable approximations. In *SIGGRAPH '99: Conference abstracts and applications*, 253.

KE, Q., AND KANADE, T. 2005. Robust l1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*.

KRÜGER, J., AND WESTERMANN, R. 2003. Linear algebra operators for GPU implementation of numerical algorithms. *ACM Transactions on Graphics 22*, 3 (July), 908–916.

LATTA, L., AND KOLB, A. 2002. Homomorphic factorization of BRDF-based lighting computation. *ACM Transactions on Graphics 21*, 3 (July), 509–516.

LAWRENCE, J., RUSINKIEWICZ, S., AND RAMAMOORTHI, R. 2004. Efficient BRDF importance sampling using a factored representation. *ACM Transactions on Graphics 23*, 3 (Aug.), 496–505.

LAWRENCE, J., BEN-ARTZI, A., DECORO, C., MATUSIK, W., PFISTER, H., RAMAMOORTHI, R., AND RUSINKIEWICZ, S. 2006. Inverse shade trees for non-parametric material representation and editing. *ACM Transactions on Graphics 25*, 3 (July), 735–745.

LEHTINEN, J., AND KAUTZ, J. 2003. Matrix radiance transfer. In *2003 ACM Symposium on Interactive 3D Graphics*, 59–64.

LEUNG, C., WONG, T., LAM, P., AND CHOY, K. 2006. An RBF-based compression method for image-based relighting. *IEEE Transactions on Image Processing 15*, 4 (Apr.), 1031–1041.

LI, W., AND SWETITS, J. 1998. The linear ℓ_1 estimator and the Huber M-estimator. *SIAM Journal on Optimization*, 8, 457–475.

MANGASARIAN, O. L., AND MUSICANT, D. R. 2000. Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 9, 950–955.

MASSELUS, V., DUTRÉ, P., AND ANRYS, F. 2002. The free-form light stage. In *EGWR '02: Proceedings of the 13th Eurographics workshop on Rendering*, 247–256.

MASSELUS, V., PEERS, P., DUTRÉ, P., AND WILLEMS, Y. D. 2004. Smooth reconstruction and compact representation of reflectance functions for image-based relighting. In *EGSR '04: Proceedings of the 15th Eurographics workshop on Rendering*, 287–298.

MELLOR-CRUMMEY, J., AND GARVIN, J. 2004. Optimizing sparse matrix-vector product computations using unroll and jam. *International Journal of High Performance Computing Applications 18*, 2 (May), 225–236.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics 22*, 3 (July), 376–381.

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

PEERS, P., VOM BERGE, K., MATUSIK, W., RAMAMOORTHI, R., LAWRENCE, J., RUSINKIEWICZ, S., AND DUTRÉ, P. 2006. A compact factored representation of heterogeneous subsurface scattering. *ACM Transactions on Graphics 25*, 3 (July), 746–753.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics 21*, 3 (July), 527–536.

SLOAN, P.-P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered principal components for precomputed radiance transfer. *ACM Transactions on Graphics 22*, 3 (July), 382–391.

SUYKENS, F., VOM BERGE, K., LAGAE, A., AND DUTRÉ, P. 2003. Interactive rendering with bidirectional texture functions. *Computer Graphics Forum 22*, 3 (Sept.), 463–472.

TSUMURA, N., OJIMA, N., SATO, K., SHIRAISHI, M., SHIMIZU, H., NABESHIMA, H., AKAZAKI, S., HORI, K., AND MIYAKE, Y. 2003. Image-based skin color and texture analysis/synthesis by extracting hemoglobin and melanin information in the skin. *ACM Transactions on Graphics 22*, 3 (July), 770–779.

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2004. Tensortextures: multilinear image-based rendering. *ACM Transactions on Graphics 23*, 3 (Aug.), 336–342.

WANG, R., TRAN, J., AND LUEBKE, D. 2004. All-frequency relighting of non-diffuse objects using separable BRDF approximation. In *Rendering Techniques 2004: 15th Eurographics Workshop on Rendering*, 345–354.

WANG, Z., LEUNG, C.-S., ZHU, Y.-S., AND WONG, T.-T. 2004. Data compression with spherical wavelets and wavelets for the image-based relighting. *Computer Vision and Image Understanding 96*, 3 (Dec.), 327–344.

WONG, T., AND LEUNG, C. 2003. Compression of illumination-adjustable images. *IEEE Transactions on Circuits and Systems for Video Technology 13*, 11 (Nov.), 1107–1118.