

Compressing the Illumination-Adjustable Images With Principal Component Analysis

Pun-Mo Ho[†]

pmho1@cse.cuhk.edu.hk

Tien-Tsin Wong[†]

ttwong@acm.org

Chi-Sing Leung[‡]

eeleungc@cityu.edu.hk

[†]The Chinese University of Hong Kong

[‡]City University of Hong Kong

Acknowledgment

This research is supported by the Chinese University of Hong Kong under Direct Grant for Research (Project No. 2050262) and Research Grants Council of the Hong Kong Special Administrative Region under RGC Earmarked Grants (Project No. CityU 1116/02E).

Abstract

The ability to change illumination is a crucial factor in image-based modeling and rendering. Image-based relighting offers such capability. However, the trade-off is the enormous increase of storage requirement. In this paper, we propose a compression scheme that effectively reduces the data volume while maintaining the real-time relighting capability. The proposed method is based on principal component analysis (PCA). A block-wise PCA is used to practically process the huge input data. The output of PCA is a set of eigenimages and the corresponding relighting coefficients. By dropping those low-energy eigenimages, the data size is drastically reduced. To further compress the data, eigenimages left are compressed using transform coding and quantization while the relighting coefficients are compressed using uniform quantization. We also suggest the suitable target bit rate for each phase of the compression method in order to preserve the visual quality. Finally, we propose a real-time engine that relights images from the compressed data.

Keywords

Image-based modeling and rendering, image-based relighting, data compression, plenoptic illumination function

1 INTRODUCTION

Image-based relighting [1], [2] offers the image-based entities, such as light field [3], lumigraph [3], [4], concentric mosaics [5] and panorama [6], an ability to change the illumination. Since the representation is image-based and no geometry is used in the computation of reflected radiance, the rendering (*relighting*) time is independent of scene complexity and can be in real time. The real-time performance is critical to its application in computer games. However, the data size is enormous when the illumination information is included in the image-based representation. Therefore, data compression is a must.

There have been several techniques [3], [7], [8], [9] in compressing data for image-based navigation (representation allowing the change of viewpoint, such as light field and concentric mosaics). Utilizing the disparity maps, Magnor and Girod [7] proposed the MPEG-like codec for light field. Zhang and Li [10] proposed a multiple reference frame (MRF) prediction approach for compressing the lumigraph. It seems that compression methods for image-based navigation may also be applicable to image-based relighting. However, there is fundamental difference between the nature of image data from these two categories. Data for image-based navigation is closely related to geometry while data for image-based relighting is closely related to surface reflectance. Without considering the data nature, the compression algorithm cannot effectively reduce the data volume. Due to the nature of random data access in image-based relighting, traditional inter-frame predictive based methods are not suitable for compressing the relighting data. These methods have to decompress a large number of reference images for relighting with complex lighting configuration. Hence, real-time relighting cannot be achieved.

Unfortunately, there have been less techniques in compressing image-based representation for relighting. Wong *et al.* [1] compressed the relighting data using spherical harmonics. Higher compression ratio is obtained by exploiting the inter-pixel correlation [11], [12]. However, compressing illumination data with spherical harmonics may suffer from blurring of specular highlight and shadows due to its nature of global support. In this paper, we propose a compression scheme based on principal component analysis (PCA). There have been some previous works [13] [14] [15] [16] using PCA. Epstein *et al.* [14] empirically showed that 5 to 7 basis images are sufficient for representing scene/object containing specular surfaces. While most existing works mainly focus on recognition and/or representation, not much attention is paid on practicability, ability of random access, optimization of compression

ratio, and real-time decompression issues. In contrast, we address these issues in this paper.

We use a block-based PCA approach to divide the problem to a manageable size. Most energy of the original data is packed in the first few principal components. By dropping most of the low-energy components, the data size is drastically reduced. We then further compress the high-energy components. The proposed method compresses image-based relighting data to a small size (0.5 - 2.5 MB for our test cases) that can be rapidly transferred through Internet. Moreover, we discuss how to achieve real-time relighting from the compressed data. We make no assumption on the surface types and no restriction on the trajectory of the light source.

We first review our image-based relighting representation in Section 2. In Section 3, we present the proposed compression method in detail. The relighting is then described in Section 4. Finally, conclusions are drawn and future directions are discussed in Section 5.

2 IMAGE-BASED RELIGHTING

For the completeness of this paper, we give a brief description of the illumination-adjustable image representation previously proposed. The details can be found in [1][11]. The original *plenoptic function* [17] is a computational human vision model for evaluating low-level vision. By explicitly parameterizing the illumination component, we extended the plenoptic function to account for illumination [11]. We call this new formulation the *plenoptic illumination function*. The new function tells us what we see when the whole scene is illuminated by a directional light source with the lighting direction \vec{L} .

When the viewpoint is fixed, the plenoptic illumination function for a color channel (wavelength) is given by

$$I = P(x, y, \vec{L}) \quad (1)$$

where (x, y) describes the position of a pixel in the image and \vec{L} describes the direction of the light vector. The plenoptic illumination function tells us the appearance of a scene/object under the directional light source with direction \vec{L} . For a fixed lighting direction \vec{L} , the plenoptic illumination function reduces to an image. Note that we do not restrict the image to be planar perspective. It can also be panoramic (cylindrical or spherical). If we fix the pixel position, the function tells the appearance of the pixel under various lighting directions.

Sampling the plenoptic illumination function is actually a process of taking pictures. In this paper, we sample the illumination component \vec{L} on the grid points of spherical coordinate system $\vec{L} = (\theta, \phi)$. Hence each reference image corresponds to a grid point in the spherical domain. The *illumination-adjustable image* refers to an image set is indexed by (θ_i, ϕ_j) for $i = 0, \dots, p - 1$ and $j = 0, \dots, q - 1$. The set contains altogether $m = pq$ images.

Given a desired light vector which is not one of the samples, the desired image of the scene can be estimated by interpolating the samples. We shall discuss the interpolation in Section 4. Even though the sampled plenoptic illumination function only tells us how the environment looks like when it is illuminated by a directional light source with unit intensity, other illumination configurations can be simulated by making use of the linearity property of light if the depth information of the scene is available. Readers are referred to [1], [11] for details.

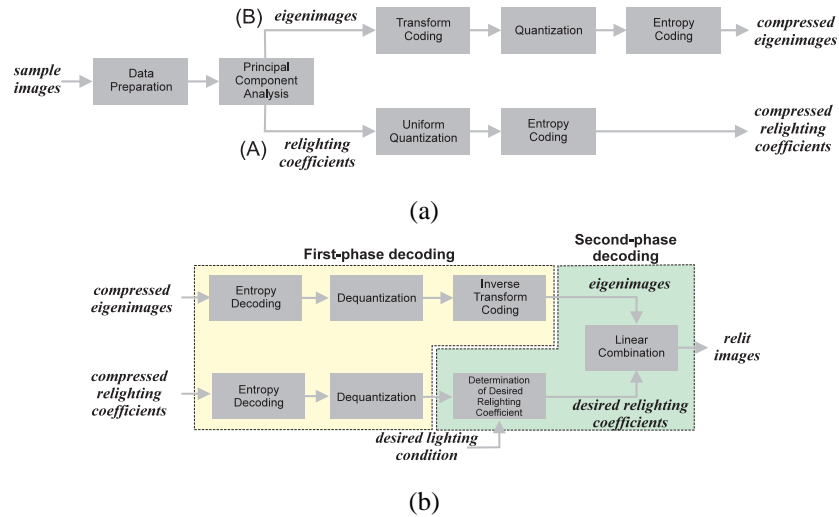


Fig. 1. Overview of (a) encoding and (b) decoding.



Fig. 2. Three samples from the data set ‘ding’, each illuminated with different lighting conditions.

3 COMPRESSION OF ILLUMINATION-ADJUSTABLE IMAGES

The compression process of illumination-adjustable image consists four main phases, namely *data preparation*, *principal component analysis*, *eigenimage coding* and *relighting coefficient coding*. Figure 1(a) illustrates the major steps. The decoding or relighting process is basically the inverse of the processes (Figure 1(b)). Note that, there is no need to reconstruct all reference images during the relighting. We only decompress the eigenimages and relighting coefficients. Given the desired lighting condition, the corresponding relighting coefficients are interpolated. Then the reconstruction (or *relighting*) can be easily done by linearly combining the eigenimages with the interpolated relighting coefficients as weights. We shall describe in details the relighting in Section 4.

3.1 Data Preparation

The goal of data preparation is to reorganize and preprocess the input data. So that the principal component analysis can effectively reduce the data dimensionality. The input is a set of reference images with the same view, but with different illumination condition. Figure 2 shows three samples from the data set ‘ding’, each illuminated by a single directional light source from different direction. Note that shadow is present in the sample images. Since there are strong correlation among them, we can apply PCA on them across the illumination domain. Hence, the reference images can be approximated by a number of eigenimages (principal components) and relighting coefficients.

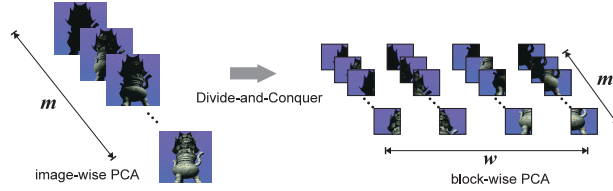


Fig. 3. A divide-and-conquer approach is used to make the computation tractable and introduce parallelism.

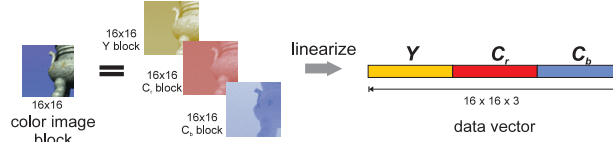


Fig. 4. A color image block is linearized to form a data vector. Its Y , C_r and C_b data are grouped inside the same vector.

3.1.1 Block Division

We can directly apply the PCA process on reference images. However, the size of images is prohibitively large for the PCA process. We propose a divide-and-conquer approach which subdivides the images into blocks. Multiple *block-wise PCAs* are then applied on the corresponding blocks. If each image is subdivided into w blocks, we perform w PCAs and each on a set of blocks as illustrated in Figure 3. With this block-based approach, the computation becomes tractable and the memory requirement is also reduced. Each set of blocks produces a number of eigenimage blocks and a set of relighting coefficients. Hence, this subdivision leads to storage overhead (each set of blocks generates an extra set of relighting coefficients) which in turn reduces the compression ratio. We will discuss the overhead shortly.

In our system, we choose a block size of 16×16 for PCA. Although the block size of 8×8 is widely used in transform coding like JPEG, there is too much storage overhead for PCA. Since we employ transform coding in the later stage, it is desirable to keep the block size to be multiple of 8×8 . A block size of 16×16 introduces less overhead while maintaining the compatibility with standard transform codec.

3.1.2 Color Model

We convert the pixel values from RGB to YC_rC_b color space [18]. Hence, we can allocate more bits for human-sensitive luminance component while less bits for chrominance components during compression in the later stage. Since there is high correlation between values in three channels (YC_rC_b), we group values from three channels and perform PCA on them as a whole (mixed channel PCA). Figure 4 illustrates how a color image block is linearized and forms a data vector. On the other hand, if we perform PCA separately on each channel (separated channel PCA), the storage requirement will increase because each color channel generates one set of relighting coefficients.

From our experiments, we found that mixed channel PCA not just reduces storage overhead (with less number of relighting coefficients), but also preserves image quality. Figure 5 compares the reconstruction errors between mixed channel PCA and separated channel PCA. The error due to mixed channel PCA is only a bit greater than that

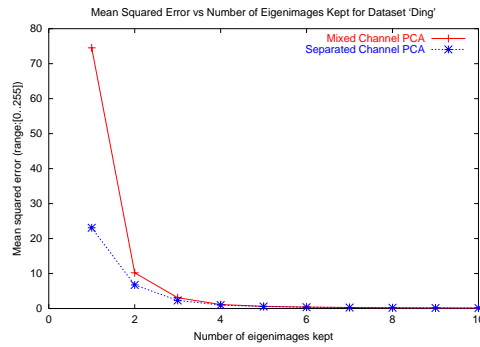


Fig. 5. Error comparison (MSE) between PCA on grouped data and PCA on separated channel data. Data set ‘ding’ is used in this experiment.

due to separated channel PCA when four or more eigenimages are used. Therefore it is more cost-effective to use mixed channel PCA. This phenomenon is consistent in different data sets.

3.1.3 Mean Extraction

We then compute a mean vector (mean image) over m input images. Each input image is subtracted by the mean image before the PCA process. This mean extraction offers a more accurate computational base for PCA. During the relighting, for each pixel position, the mean value is added to obtain the reconstructed radiance value.

3.2 Principal Component Analysis

3.2.1 Decomposition Process

We use the singular value decomposition (SVD) [19] to perform the PCA process. Consider a set of m color blocks, each block is linearized to a n -dimensional data vector including $Y C_R C_B$. All data vectors are stacked to form a $m \times n$ data matrix \mathbf{M} (Figure 6). The SVD factorizes \mathbf{M} into 3 matrices,

$$\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V}^T, \quad (2)$$

where \mathbf{U} and \mathbf{V}^T are unitary matrices with dimensions $m \times m$ and $m \times n$ respectively. The matrix \mathbf{S} (a $m \times m$ diagonal matrix) contains the singular values. Those elements are sorted in descending order, such that, $s_1 \geq s_2 \geq \dots \geq s_m \geq 0$. The singular values in \mathbf{S} may be large in magnitude while the values in \mathbf{U} and \mathbf{V}^T are relatively small. Storage and manipulation of both large and small values may harm the accuracy. It is not desirable to compress

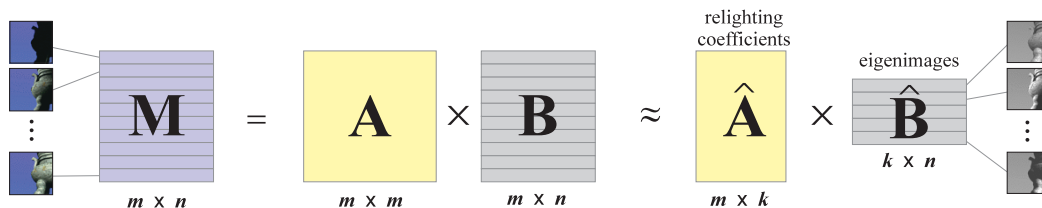


Fig. 6. Using PCA, we approximate the original data matrix \mathbf{M} by two smaller matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$.

data source containing both large and small values. By splitting the singular values into two halves (taking square roots) and multiply them to \mathbf{U} and \mathbf{V}^T , we evenly distribute the energy of singular values to obtain a $m \times m$ matrix \mathbf{A} and a $m \times n$ matrix \mathbf{B} , given by

$$\mathbf{A} = \mathbf{U}\sqrt{\mathbf{S}} \quad \mathbf{B} = \sqrt{\mathbf{S}}\mathbf{V}^T. \quad (3)$$

From our experiments, we find that most elements in \mathbf{A} and \mathbf{B} fall into the range of $[-1, 1]$. This nice property facilitates our hardware-assisted relighting in Section 4. Now, the matrix \mathbf{M} is rewritten as

$$\mathbf{M} = \mathbf{A}\mathbf{B}. \quad (4)$$

Figure 7 illustrates the reconstruction graphically. We call the rows in \mathbf{B} the *eigenimages* and the elements in \mathbf{A} the *relighting coefficients*. Every row of \mathbf{M} (image block) can be reconstructed by linearly combining all rows in \mathbf{B} . The corresponding weights (coefficients) are kept in a row in \mathbf{A} . The i -th row of \mathbf{A} contains the relighting coefficients for the i -th reference image. The j -th column of coefficients in \mathbf{A} are weighting factors for the j -th eigenimage.

Since SVD packs the energy into the first few eigen components, the matrix \mathbf{M} can be approximated by

$$\mathbf{M} = \mathbf{A}\mathbf{B} \approx \hat{\mathbf{A}}\hat{\mathbf{B}} \quad (5)$$

where the $m \times k$ matrix $\hat{\mathbf{A}}$ is formed by keeping only the first k columns of \mathbf{A} and the $k \times n$ matrix $\hat{\mathbf{B}}$ is formed by keeping the first k rows of \mathbf{B} (Figure 6). We use the Frobenius norm as an error measure, given by

$$e_k = \|\mathbf{M} - \hat{\mathbf{A}}\hat{\mathbf{B}}\|_F^2 = \sum_{i=k+1}^m s_i^2. \quad (6)$$

Dividing e_k by the number of elements in \mathbf{M} , we obtain the mean-squared-error (MSE) between the original and the approximation.

From another point of view, the PCA decomposes the plenoptic illumination function into two parts, lighting and spatial, given by

$$P(x, y, \vec{L}) = \sum_i^k f_i(\vec{L}) g_i(x, y), \quad (7)$$

where k is the number of eigenimages kept; and the functions $f_i(\vec{L})$'s and $g_i(x, y)$'s correspond to the relighting coefficients and eigenimages respectively. Interestingly, Weiss [20] decomposed the plenoptic illumination function in the log domain, given by

$$\log P(x, y, \vec{L}) = \alpha(x, y) + \beta(x, y, \vec{L}), \quad (8)$$

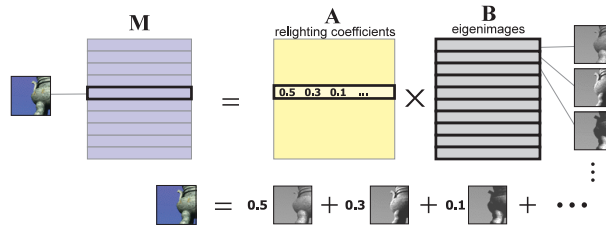


Fig. 7. Each row (image block) of \mathbf{M} is a linear combination of rows (eigenimages) in \mathbf{B} .

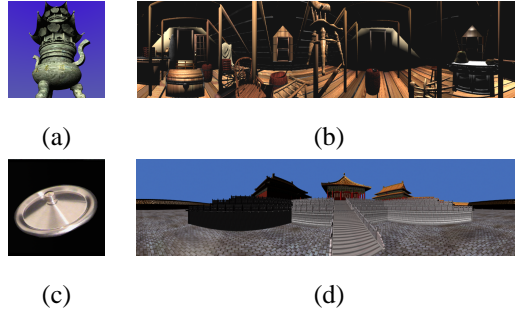


Fig. 8. Four data sets tested throughout the experiments: (a) ‘ding’, (b) ‘attic’, (c) ‘cover’ and (d) ‘forbidden’. Data set ‘cover’ contains real-world images while others are synthetic.

Data set	Resolution	Sampling rate ($\theta \times \phi$)	Real/Synthetic	With Shadow
ding	512×512	30×40	Synthetic	Yes
attic	1024×256	15×20	Synthetic	No
forbidden	1024×256	30×40	Synthetic	Yes
cover	256×256	45×90	Real	Yes

TABLE I

CHARACTERISTICS OF TESTED DATA SETS.

where $\alpha(x, y)$ is the reflectance component (the intrinsic image), and $\beta(x, y, \vec{L})$ is the illumination component which is a 4D function. This intrinsic image approach is mainly used in the view based template matching and shape-from-shading.

3.2.2 Evaluation

The number of eigenimages required depends on the reflection property and geometry of the scene. To determine the number of required eigenimages, we measure the peak signal-to-noise ratio (PSNR) of the reconstructed images. Four data sets (Figure 8) are tested, they are ‘ding’, ‘attic’, ‘forbidden’ and ‘cover’. The first three sets are synthetic and the last one contains real-world images. While the data set ‘attic’ contains no shadow, all other data sets contain shadow. Table I shows the characteristics of each data set. From Figure 9, the PNSR increases rapidly when the number of eigenimages k is below 9 or 10. The increase of PSNR slows down after that point. For example, the first 9 eigenimages of ‘attic’ give more than 40 dB. Other data sets have similar behavior.

3.3 Eigenimage Coding

After PCA, we obtain a set of eigenimages (matrix $\hat{\mathbf{B}}$) and the corresponding relighting coefficients (matrix $\hat{\mathbf{A}}$). Since the properties of these two sets of values are different, we compress them differently. We first discuss the compression of $\hat{\mathbf{B}}_i$'s for $i = 0, \dots, w - 1$. The subscript i denotes which block position we are referring.

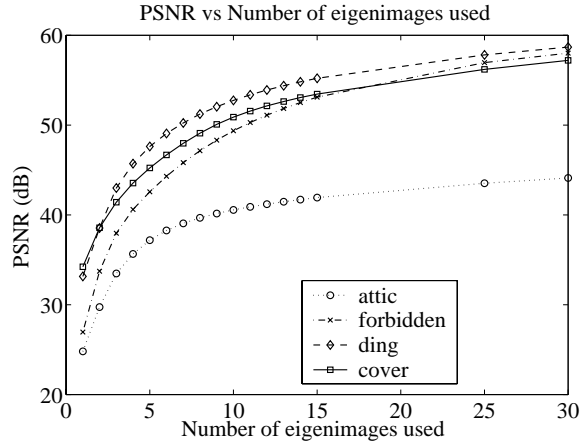


Fig. 9. PSNR vs number of eigenimages used for reconstruction.

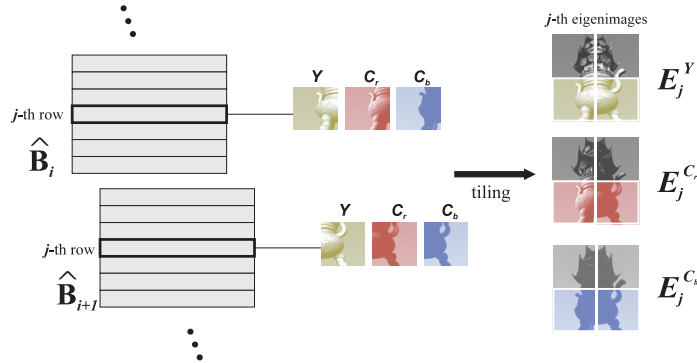


Fig. 10. Matrix $\hat{\mathbf{B}}$ is rebinned to form the tiled eigenimages.

3.3.1 Discrete Cosine Transform

The j -th row in $\hat{\mathbf{B}}_i$ gives three 16×16 blocks (Y , C_r and C_b blocks) in the j -th eigenimage. We then tile the blocks according to their position in the original image and form three tiled eigenimages. Figure 10 illustrates such tiling. Note that we separate the Y , C_r and C_b during tiling. Therefore we have altogether $3k$ tiled eigenimages after tiling. The tiled eigenimages are denoted as E_j^c for $c = Y, C_r$ or C_b and $j = 0, \dots, k - 1$.

Each tiled eigenimage is itself an image. Figure 11 shows the first 4 eigenimages from the data set ‘ding’. Since we perform block-wise PCA, it is not surprising that boundary is observable in the tiled eigenimages. It seems that low-order eigenimages (high energy) are smoother than high-order eigenimages (low energy). This behavior is similar to natural images where low-frequency signals have more energy than high-frequency signals. We then compress the tiled eigenimages using discrete cosine transform (DCT). We apply 8×8 block-based DCT on the tiled eigenimages to avoid crossing the boundary of 16×16 tiles.

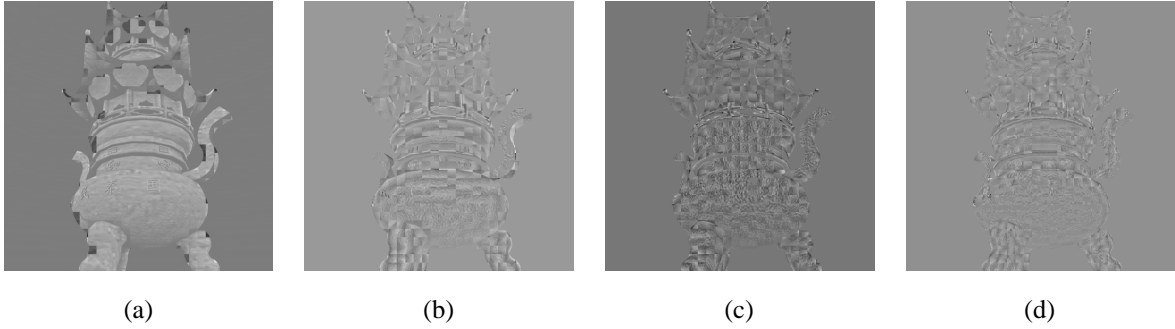


Fig. 11. The first 4 tiled eigenimages (Y channel) of 'ding'.

3.3.2 Bit Allocation and Quantization

To quantize the DCT coefficients, we apply non-uniform quantization. An overall target bit rate R_B is specified for the whole eigenimage encoding process. The bit rate is then sub-allocated to three color channels according to the commonly used ratio: $R^Y : R^{C_r} : R^{C_b} = 2 : 1 : 1$. Same bit rate is then assigned to all tiled eigenimages in the same channel (in despite of their orders). The rationale is that we believe it is visually important to preserve high-frequency details in high-order eigenimages even though they contribute less energy.

For each tiled eigenimage, 64 data sources (one for each DCT coefficient) are formed. The bit rate R_μ for each source [21] is given by

$$R_\mu = R_B + \frac{1}{2} \log_2(\alpha_\mu \sigma_\mu^2) - \frac{1}{2H} \left\{ \sum_{\nu=0}^{H-1} \log_2(\alpha_\nu \sigma_\nu^2) \right\}. \quad (9)$$

where $H = 64$ is the number of data sources; σ_μ^2 is the variance of the source; α_μ is a user-defined constant. Constant α_μ depends on the density of source as well as the type of encoding method used. The DC component of DCT is assumed to have a Gaussian distribution as described in [22]. All other sources are modeled as Laplacian sources. Constant α_μ is chosen to be 2.7 and 4.5 for Gaussian and Laplacian sources respectively [21]. After we obtain the bit rates, each source is quantized by the generalized Lloyd algorithm [23].

We then perform arithmetic coding [24] on the quantized indices to further reduce the data size. The usual reduction of this entropy coding is around 5% - 6%. The gain of the entropy coding is low because the generalized Lloyd algorithm we used is a source optimized quantization.

3.3.3 Evaluation

To evaluate the performance of our eigenimage coding, we measure the reconstruction error of the eigenimages versus the target bit rate R_B . Figure 12 shows the statistics of four data sets. We also evaluate the coding visually. Figure 13 compares the reconstructed images from the forbidden city data set in a side-by-side manner. When a low bit rate is used, the reconstruction exhibits blocky artifact and image details are lost. Since we apply DCT-based coding to eigenimages, the visual artifact is similar to the artifact of JPEG images at low bit rate. The quality improves as the bit rate increases. There is no significant artifact when the bit rate is raised to 3.0. From the above figure and statistics, 3.0 bit is a cost-effective choice for eigenimage coding as it returns more than 35 dB in each

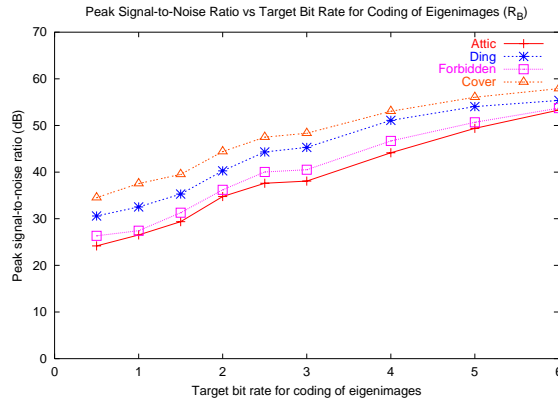


Fig. 12. PSNR versus target bit rate for eigenimage coding (R_B).

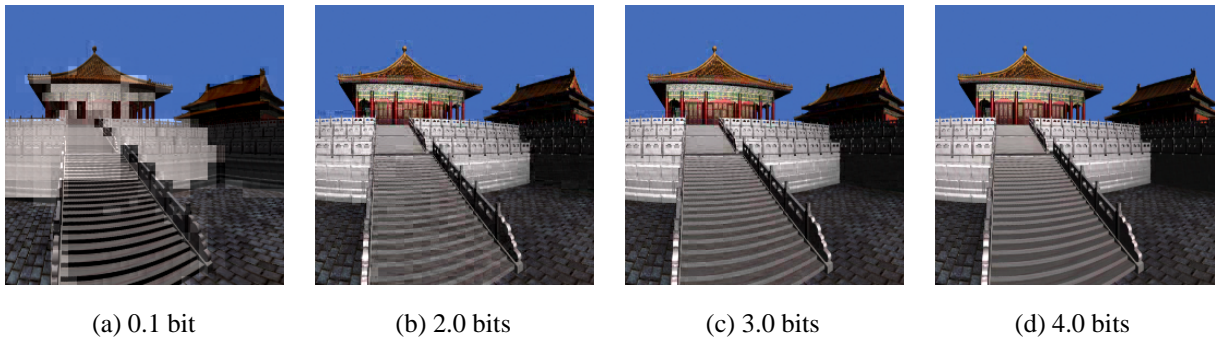


Fig. 13. Visual artifacts introduced by encoding eigenimages with different target bit rates R_B .

test case.

3.3.4 Mean Image Coding

The encoding method for the three mean images, one for each channel, is similar to the encoding method used in eigenimages. We subdivide the mean images into 8×8 blocks. Afterwards, the similar DCT and quantization in compressing eigenimages are applied. Just like the case of eigenimage coding, we find that there is no significant artifact when the bit rate reaches 3-4 bits. For example in the ‘ding’ data set, the PSNR between the original and the reconstructed mean images is more than 50 dB when the bit rate is 3 bits. This is sufficient for high-quality rendering.

3.4 Relighting Coefficient Coding

Unlike eigenimages, the relighting coefficients $\hat{\mathbf{A}}_i$ for $i = 0, \dots, w - 1$ in different block positions, do not resemble an image. The total number of coefficients is much larger than the total number of pixels in eigenimages. It may not be practical to compress them using non-uniform quantization as the training of codebook is time-consuming. Instead, we encode them using uniform quantization.

We find that the variances of values from the same column of different $\hat{\mathbf{A}}_i$ (different block position) for $i = 0, \dots, w - 1$ are similar. Hence we group them into the same data source. Figure 14 illustrates how data sources

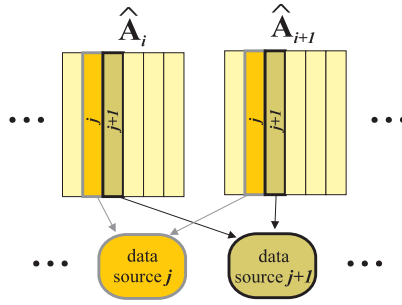


Fig. 14. Each data source is formed by grouping elements from the same columns in different $\hat{\mathbf{A}}_i$.

are formed. Since there are k columns in $\hat{\mathbf{A}}_i$, there are k data sources.

3.4.1 Quantization and Bit Allocation

The bit allocation scheme (Equation 9) in Section 3.3.2 is used for each data source S_μ with notation R_B replaced by R_A (target bit rate for relighting coefficient coding). We have examined the histograms of those k sources. All of them exhibit a Laplacian distribution. Hence we set constant α_μ to be 4.5. The step size Δ_μ for the μ -th source is given by

$$\Delta_\mu^2 = \frac{|\max_\mu|^2}{2R_\mu} \quad (10)$$

where \max_μ is the maximum value of the μ -th source. Afterwards, the coefficients are quantized using a uniform quantizer with step size Δ_μ . The quantized indices are further coded using arithmetic coding. The rough reduction due to arithmetic coding is around 35%. The gain of entropy coding is better than that of entropy coding in eigenimage encoding is because here the data are quantized uniformly and may not be source optimized.

3.4.2 Evaluation

We measure the reconstruction error of the relighting coefficients versus different target bit rate R_A (Figure 15). Figure 16 shows the reconstructed images from the data set of panoramic forbidden city. Unlike the visual artifact in the eigenimage encoding, image details are not lost too much. Instead, the major visual artifact introduced at low bit rate is the tone inconsistency among the neighboring blocks. This is because relighting coefficients are mainly responsible for information in illumination domain. Low bit rate in coding relighting coefficients in some sense is analogous to the undersampling of illumination dimension. The tone inconsistency becomes unobservable when the target bit rate R_A is raised to 4.0. From the above figure and statistics, 4.0 bit is a good choice among the tested data sets as it gives more than 30 dB and reduces the tone inconsistency.

3.5 Overall Evaluation

Just like video compression, the PCA approach for relighting also compresses multiple images. Hence, it is more appropriate to compare our method to video compression methods. However, there is fundamental difference between our image-based relighting application and video playback. For video playback, the frames are reconstructed

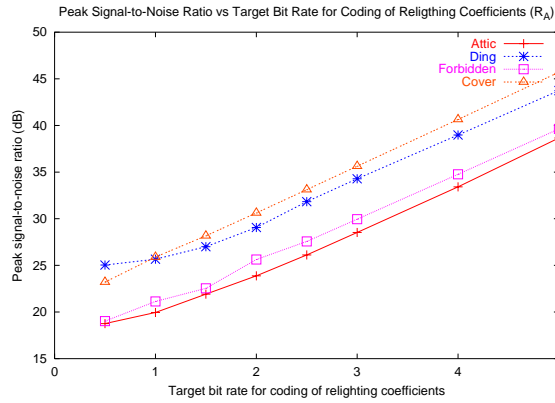


Fig. 15. PSNR versus target bit rate for coding of relighting coefficients (R_A).

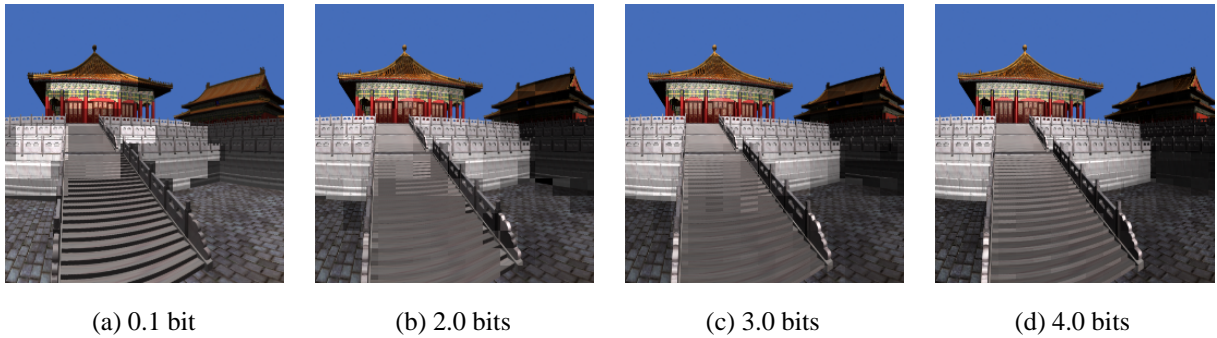


Fig. 16. Visual artifacts introduced by encoding relighting coefficients with different target bit rates R_A .

in specific orders, either forward or backward. On the other hand, in our image-based relighting application, the image is relit according to the unconstrained lighting configuration specified by user. Therefore, we need to support *random access* of compressed images in *real time*. Besides, we also compare the PCA approach with the spherical harmonic encoding previously developed by us [12].

We use the setting suggested in previous sections ($k = 9$, $R_A = 4.0$ and $R_B = 3.0$) to compress the four data sets. The bit rate for encoding mean images is set to 3. The compression ratios and reconstruction errors are then measured and recorded. Next, the same four data sets are compressed using two video coding methods, Motion JPEG (M-JPEG) and MPEG. To have a fair comparison with the MPEG approach, we need to order input reference images such that two consecutive images (frames) are coherent. To do so, reference images with the same elevation θ form a group of frames. Within the same group, reference images are ordered according to their values of ϕ . Then, groups of frames are ordered according to the value of θ . We then determine the compression ratios of M-JPEG and MPEG given the same image quality (in terms of PSNR).

Since we cannot directly control the desired image quality during video compression, we can only estimate the compression ratio in the following manner. For each video coding method and each data set, we compress the data set using different target bit rates. The corresponding reconstruction errors are measured. Then a graph of reconstruction error against the target bit rate is plotted and the compression ratio giving the ‘same’ image quality

Data set	PSNR (dB)	M-JPEG	MPEG	SH	The proposed PCA
ding	36.9	50.5	45.0	171	538.9 (1.67 MB)
attic	31.0	42.5	57.4	105	97.8 (2.30 MB)
cover	39.1	42.0	43.0	433	576.0 (0.44 MB)
forbidden	33.2	22.9	54.7	150	373.4 (2.41 MB)

TABLE II

COMPARISON OF COMPRESSION RATIOS OF DIFFERENT ENCODING METHODS

is estimated from this graph. We tabulated the compression ratios of M-JPEG and MPEG in Table II.

In every case, the proposed method out-performs the two video coding methods. It is expected that M-JPEG is the worst because it does not exploit the coherence among images. But our method also out-performs MPEG. The result may due to the fact that MPEG can only exploit coherence among the two consecutive frames (1D) while our method utilizes coherence in 2D. In general, the PCA approach is better than the spherical harmonic approach (column ‘SH’ in Table II). Spherical harmonic uses a fixed basis to capture the similarity in the illumination domain while the PCA approach uses a data dependent basis (eigenimages). One disadvantage of the PCA approach is that its implementation is more complex due to the complicated arrangement of relighting coefficients in memory.

The sizes of compressed files generated by our method (shown in the round brackets of the last column in Table II) are around 0.5 - 2.5 MB only. Obviously, these compressed files can be rapidly transferred through Internet nowadays. The proposed method not just effectively compresses data but also facilitates the user retrieval pattern (2D) in our relighting application.

4 RELIGHTING

The relighting is revealed in Figure 1(b). It can be divided into two major phases. In the first phase, the compressed eigenimages, mean images and relighting coefficients are decompressed into memory. In the second phase, the relighting engine interpolates the relighting coefficients and synthesizes the desired image by linearly combining the eigenimages, according to the user specified lighting condition.

Recall that the block-wise eigenimages are tiled to form an image-wise eigenimages before the transform coding. Hence there is no need to tile the decompressed eigenimages as they are already in an image form. The only process needed is to convert them from YC_rC_b to RGB color space. As the reconstruction is a linear process (linear combination of eigenimages), the color transform can be simply applied to the eigenimages instead of the reconstructed images.

Second-phase decoding mainly deals with the user request during run-time. Figure 17 depicts the reconstruction and interpolation process. For each light vector (θ, ϕ) specified by the user, at most four neighboring images on the grid points are reconstructed. Each of them is reconstructed by linearly combining the eigenimages with the corresponding relighting coefficients as weights. Then these reconstructed images are bi-linearly interpolated to synthesize the desired image. In practice, there is no need to reconstruct these four images. As the interpolation is

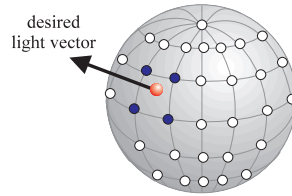


Fig. 17. The relighting due to one light source requires the reconstruction and interpolation of at most four neighboring samples.

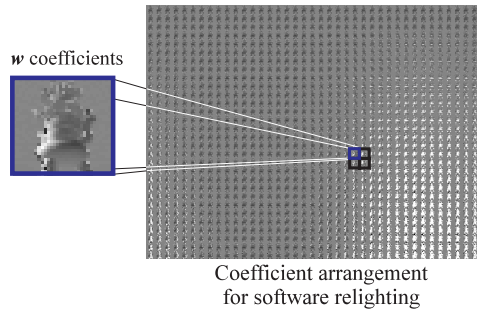


Fig. 18. Coefficient arrangements for software relighting.

linear, we can simply bi-linearly interpolate the four sets of relighting coefficients and synthesize the desired image by linearly combining the eigenimages with the interpolated coefficients as weights. The mean image is then added to shift the values back to the original range (Section 3.1.3). If there are more than one directional light sources used for relighting, the above process is repeated for each light source and the result is accumulated (summed) to generate the final desired image.

In order to efficiently relight the image, we arrange the relighting coefficients in the memory so that coefficients being accessed in the same relighting pass are grouped together. When interpolating the relighting coefficients, w relighting coefficients (one for each block, there are w blocks in an image) of the same order are accessed simultaneously. Hence, we group those w coefficients together in order to speed up the memory access. Figure 18 shows the arrangement of relighting coefficients for the ding data set. Due to the space limit, only the coefficients corresponding to the first eigenimage are shown. It is a matrix of sub-matrices of w coefficients. Since there may be at most four sets of relighting coefficients being looked up in one relighting pass, four neighboring sub-matrices (highlighted as boxes in Figure 18) may be accessed at the same time. For pure software implementation, the relighting time for a novel image is about 0.6 to 2 seconds on a Pentium III 800 MHz machine.

With the advances of current graphics hardware, we are able to perform parallel operations on consumer-level graphics boards, such as nVidia GeForce4 or ATI Radeon 8500. These graphics boards are equipped with graphics processing unit (GPU) to execute user-defined *shader* (machine-code like program) [25]. The architecture is basically a SIMD design, a single shader is executed on multiple data (such as pixels or vertices).

This motivates us to utilize the commodity hardware for relighting purpose. Since the major process during relighting is the linear combination of eigenimages, this can be straightforwardly implemented by storing each eigenimage in the texture unit of the hardware. The SIMD-based hardware is efficient in performing image-wise

Data set	frame rate (fps)	PSNR (dB)
ding	18.9	27.9
attic	18.9	28.9
cover	76.9	27.9
forbidden	18.9	28.7

TABLE III

FRAME RATE AND ERROR MEASUREMENT OF HARDWARE-ACCELERATED RELIGHTING.

operation such as multiplication, addition and subtraction. Pixels in the texture can be manipulated (multiplied, added or subtracted) in parallel. Due to the limitation of our specific hardware, only four texture units can be used in one shader pass. The whole reconstruction requires multiple passes. Nevertheless, this hardware-accelerated relighting achieves real-time performance as evidenced by the second column of Table III. All statistics are taken on a Pentium III 800 MHz equipped with GeForce 3 graphics accelerator. Note that our current implementation is still a prototype and the program code is not optimized.

The precision of our graphics hardware is limited to 8-bit per pixel and all computations must be taken within the range of [0,1]. Even though we distribute energy evenly to matrices $\hat{\mathbf{A}}$ and $\hat{\mathbf{B}}$ during SVD (Section 3.2), error still exists. We measure the error solely due to this hardware acceleration process and tabulate them in the third column of Table III. At the time we conducted the experiments, the graphics hardwares available only support 8-bit computation and very limited number of textures. Nowadays 32-bit floating point computation and larger number of textures are already supported by consumer-level graphics hardwares. Hence the relighting can be further sped up and there is no loss in PSNR due to the hardware computation.

5 CONCLUSION

In this paper, we proposed a block-wise PCA-based method for compressing image-based relighting data. Since the data volume is enormous, we divide-and-conquer the problem in order to make it manageable. The eigenimages and relighting coefficients are compressed separately according to their data natures. A high compression ratio is achieved (Table II) while maintaining the visual quality. It out-performs standard video coding method as it exploits the data correlation specific to image-based relighting. To facilitate real-time relighting, we utilize consumer-level graphics hardware to synthesize the desired images. The real-time performance also demonstrates the advantage of image-based approach. Currently we store equal number of eigenimages for each block. The compression ratio can be further increased if the number of eigenimages adapts to the image content. For instance, constant background may require less number of eigenimages.

ACKNOWLEDGMENT

This research is supported by the Chinese University of Hong Kong under Direct Grant for Research (Project No. 2050262) and Research Grants Council of the Hong Kong Special Administrative Region under RGC Earmarked

Grants (Project No. CityU 1116/02E).

REFERENCES

- [1] T.-T. Wong, P.-A. Heng, S.-H. Or, and W.-Y. Ng, "Image-based rendering with controllable illumination," in *Eighth Eurographics Workshop on Rendering*, June 1997, pp. 13–22.
- [2] T.-T. Wong, C.-W. Fu, and P.-A. Heng, "Interactive relighting of panoramas," *IEEE Computer Graphics & Applications*, vol. 21, no. 2, pp. 32–41, March-April 2001.
- [3] M. Levoy and P. Hanrahan, "Light field rendering," in *SIGGRAPH*, August 1996, pp. 31–42.
- [4] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *SIGGRAPH*, August 1996, pp. 43–54.
- [5] H.-Y. Shum and L.-W. He, "Rendering with concentric mosaics," in *SIGGRAPH*, August 1999, pp. 299–306.
- [6] S. E. Chen, "QuickTime VR - an image-based approach to virtual environment navigation," in *Computer Graphics Proceedings, Annual Conference Series, SIGGRAPH'95*, August 1995, pp. 29–38.
- [7] M. Magnor and B. Girod, "Data compression for light-field rendering," in *IEEE transactions on CSVT*, April 2000, vol. 10, pp. 338–343.
- [8] C. Bajaj, I. Ihm, and S. Park, "3D RGB image compression for interactive applications," *ACM Transactions on Graphics*, vol. 20, pp. 10–38, January 2001.
- [9] H.-Y. Shum, K.-T. Ng, and S.-C. Chan, "Virtual reality using the concentric mosaic: construction, rendering and data compression," in *Proceedings of ICIP 2000*, 2000, pp. 644–647.
- [10] C. Zhang and J. Li, "Compression of lumigraph with multiple reference frame (MRF) prediction and just-in-time rendering," in *IEEE Conference on Data Compression*, 2000, pp. 253–262.
- [11] T.-T. Wong, C.-W. Fu, P.-A. Heng, and C.-S. Leung, "The plenoptic illumination function," *IEEE Transactions on Multimedia*, vol. 4, no. 3, September 2002.
- [12] T.-T. Wong and C.-S. Leung, "Compression of illumination-adjustable images," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 11, pp. 1107–1118, November 2003.
- [13] P. N. Belhumeur and D. J. Kriegman, "What is the set of images of an object under all possible lighting conditions?," in *IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 270–277.
- [14] R. Epstein, P. W. Hallinan, and A. L. Yuille, "5+/-2 eigenimages suffice: An empirical investigation of low-dimensional lighting models," in *Proceedings of IEEE Workshop on Physics-Based Modeling in Computer Vision*, Cambridge, Massachusetts, June 1995, pp. 108–116.
- [15] Z. Zhang, "Modeling geometric structure and illumination variation of a scene from real images," in *International Conference on Computer Vision*, January 1998.
- [16] K. Nishino, Y. Sato, and K. Ikeuchi, "Eigen-texture method: Appearance compression based on 3D model," in *IEEE Conference on Computer Vision and Pattern Recognition*, June 1999, vol. 1, pp. 618–624.
- [17] E. H. Adelson and J. R. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, Michael S. Landy and J. Anthony Movshon, Eds., chapter 1, pp. 3–20. MIT Press, 1991.
- [18] E. Hamilton, "JPEG file interchange format, version 1.02," <http://www.jpeg.org/public/jfif.pdf>, September 1992.
- [19] G. W. Stewart, "On the early history of the singular value decomposition," Tech. Rep. CS-TR-2855, Department of Computer Science, University of Maryland, College Park, 1992.
- [20] Y. Weiss, "Deriving intrinsic image from image sequences," in *Proceedings of ICCV'01*, 2001.
- [21] A. N. Akansu and M. J. T. Smith, *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets*, Academic Press, 1992.
- [22] R. C. Reininger and J. D. Gibson, "Distribution of the two-dimensional dct coefficients for images," *IEEE Transactions on Communications*, vol. COM-31, pp. 835–839, 1983.
- [23] R. M. Gray, *Source Coding Theory*, Kluwer Academic, 1990.
- [24] A. M. Tekalp, *Digital Video Processing*, Prentice-Hall, 1998.
- [25] E. Lindholm, M. J. Kilgard, and H. Moreton, "A user-programmable vertex engine," in *Proceedings of SIGGRAPH 2001*, August 2001, pp. 149–158.