An RBF-based Compression Method for Image-based Relighting

Chi-Sing Leung[1], Tien-Tsin Wong[2], Ping-Man Lam[1] and Kwok-Hung Choy[1]

[1] Department of Electronic Engineering,

The City University of Hong Kong, Kowloon Tong, Hong Kong

email: eeleungc@cityu.edu.hk

[2] Department of Computer Science and Engineering,

The Chinese University of Hong Kong, Shatin, Hong Kong

email: ttwong@cse.cuhk.edu.hk

Abstract

In image-based relighting, the data is a set of pre-recorded reference images sampled under different lighting directions. In other words, a pixel is associated with a number of sampled radiance values. Since a novel image under a novel illumination condition is rendered from these reference images, the quality of rendered images depends on the number of reference images used. This paper presents a two-level compression method. In the first level, the plenoptic property of a pixel is approximated by a spherical radial basis function (SRBF) network. That means that the spherical plenoptic function of each pixel is represented by a number of SRBF weights. In the second level, we apply a wavelet-based method to compress these SRBF weights. To reduce the visual artifact due to quantization noise, we develop a constrained method for estimating the SRBF weights. In terms of distortion, our proposed approach is superior to JPEG, JPEG2000, and MPEG. Compared with the spherical harmonics approach, our approach has a lower complexity, while the visual quality is comparable. Such cost-effectiveness in rendering is crucial in computer game applications. The real-time rendering method for our SRBF representation is also discussed.

Keywords: Image-based relighting, plenoptic illumination function, spherical radial basis function, constrained least square fitting, wavelet transform.

1 INTRODUCTION

Image–based relighting (IBR) [1][2][3][4] offers image-based entities, such as light field [5], lumigraph [5][6], concentric mosaics [7] and panorama [8], an ability to change the illumination. In [1], several research issues in capturing real-world relighting data were discussed. Since no geometry is used during rendering, IBR is able to achieve real-time performance which is crucial for computer game applications. Wong and Leung [9][10] proposed the plenoptic illumination function (PIF) model to describe the radiance property of a scene under various illumination conditions. In IBR, there is a set of pre-recorded images. Each pre-recorded image captures a scene or an object illuminated under a known lighting direction. These pre-recorded images are called *reference images*. Therefore, a PIF is a 4D function, with 2D in spatial pixel location (rectangular structure) and 2D in lighting dimension (spherical structure). In other words, each pixel is associated with a spherical function of recorded radiances, known as apparent bidirectional reflectance distribution function (ABRDF). With reference images, novel images of the scene under novel illumination conditions can be rendered (*relit*).

The quality of relit images depends on the number of reference images used. Typical relighting data requires hundreds of megabytes of storage. Hence an effective compression method is a must. Another concern is the relighting speed. In order to support real-time illumination control, the compression scheme must be designed to favor the *direct relighting* from the compressed data, without reconstructing unnecessary reference images. This requires the *random-accessibility* to the required radiance values which are stored in compressed format. We called this kind of compression schemes *rendering-friendly*. Not all compression methods are rendering-friendly. For example, run-length encoding is not rendering-friendly because it requires the decompression of unwanted data in front of the required one.

Compressing reference images with JPEG [11] or JPEG2000 [12] is not effective because these compression methods do not exploit the strong correlation among reference images. Using video compression methods, such as MPEG [13], to encode reference images is also not effective. This is because most current video compression methods encode the data in a 3D manner only. One may suggest to use some multi–dimensional compression methods to handle IBR data. However, most existing methods are designed to handle data organized in a rectangular structure. They do not match the nature of IBR data. Moreover, standard image or video compression methods are not rendering-friendly.

The ABRDF of a pixel can be represented in the spherical harmonics (SH) domain [2][14]. The resultant SH coefficients from all pixels can be further compressed. However, higher frequency SH basis functions are defined in more complex mathematical forms. They consist of associated Legendre polynomials which are recursively defined. For certain kinds of light sources, such as point light sources, the evaluations of SH basis functions are different for different pixels. That means relighting with point light sources could be very slow. Hence in many computer graphics literatures [14][15][16], only the first 16 or 25 SH basis functions are considered. Fast evaluation can be achieved by table look-up. Wong *et al.* [17] made use of cube-maps of graphics processing unit (GPU) to achieve real-time per-pixel evaluation of SH basis functions. However, modern GPUs have a limitation on the number of active textures (e.g. 16 on nVidia GeforceFX 6800) and memory size. These cube-maps consume memory, as well as, the scarce count of active textures.

This paper proposes a compression scheme based on spherical radial basis function (SRBF) networks [18][19][20]. The homogeneity and simplicity of kernel functions used in SRBF networks make the relighting process more efficient and practical. Our objectives are to reduce the resource requirement in the rendering process and to speed up the process. Previous works using radial basis function (RBF) networks focused on the problem whose input space is defined in a Cartesian coordinate system [21][22]. Some works extended the RBF model to the spherical space [19][20].

Our SRBF compression method is a two–level compression process. In the first level, we use a constrained least square algorithm [23] to estimate SRBF weights from reference images. After that, each pixel is associated with a number of SRBF weights. Secondly, we further compress the estimated SRBF weights using a wavelet–based method. With the constrained least square solution, the relighting process is much less sensitive to the quantization noise introduced by the second-level compression.

During rendering, the wavelet compressed SRBF weights are first decompressed and then loaded into graphics memory. Given a novel illumination condition, we can directly obtain the approximated radiance values from the decompressed SRBF weights without reconstructing all reference images. Hence users can control the illumination condition in real-time. In addition, unlike the SH approach in which its basis functions are complicated, kernel functions used in SRBF networks are very simple. Hence, both the software or hardware rendering are simple and efficient.

Simulation results show that our approach is better than JPEG, JPEG2000, and MPEG. In terms of distortion, our approach is comparable to the SH one. In terms of memory consumption, our approach consumes less memory than the SH approach. Moreover, the rendering speed of our SRBF approach is much faster than that of the SH approach.

The rest of this paper is organized as follow. We first give a review on existing compression methods for image-based representation in Section 2. The IBR representation and our system are briefly discussed in Sections 3 and 4, respectively. Section 5 describes the first-level compression. In Section 6, we present the second-level compression process which further compresses the estimated SRBF weights. Section 7 describes the rendering process on modern GPUs. Simulation results and comparisons are presented in Section 8. Finally, discussions on the extension of our approach to 3D scenery and conclusions are presented in Section 9.

2 Related Work

Several techniques in image–based modelling and rendering were proposed in recent years. However, not much attention has been paid on the compression issue of image–based rendering data. We roughly classify previous works into two main categories, namely compression for navigation and compression for relighting.

2.1 Navigation

There are several techniques [5][24][25] in compressing data for image–based navigation (changing viewpoint), such as light field and concentric mosaics. For example, Magnor and Girod [24] proposed a MPEG-like codec for light field. Zhang and Li [26] proposed a multiple reference frame (MRF) prediction approach for compressing lumigraphs. Peter and Straber [27] used a 4D wavelet compression method. Bajaj *et al.* [25] proposed a general multi–dimensional compression scheme for light field. Unlike the 4D nature of light field/lumigraph, concentric mosaics [7] are 3D data. A standard MPEG4 codec can be used to compress concentric mosaics. Leung and Chen [28] further proposed a MPEG-like motion compensation method for compressing concentric mosaics.

2.2 Relighting

IBR extends the image-based representation along the illumination dimension. Hence, it drastically increases the data volume. For example, a color data set (spatial resolution is 512×512) with 1,200 lighting samples occupies more than 900MB of storage. It seems that standard compression methods [11][13][29][30] for image and video are applicable to IBR data. However, due to the nature of random data access in IBR, traditional predictive based approaches or multi-dimensional approaches are not suitable. For complex lighting conditions, relighting the whole image requires access to a large number of reference images. Therefore, we need a *rendering-friendly* coding method that compresses IBR data on graphics memory (not just the secondary storage). The method should be able to decompress the data that only needed.

The eigen-image approach is one possible way to compress illumination data [31]. It is previously used in computer vision for the purpose of face recognition under various illumination conditions. Several variants [32] [33] [34] [31] [35] [36] have been proposed recently. The desired images can be synthesized from eigen–images and eigen–coefficients. However, objects in the scene are usually assumed to be *Lambertian*. Although empirical experiment [31] showed that 3 to 7 eigen–images may be good enough to represent specular surfaces, the correctness of generated images actually depends on the scene geometry. Specular highlight and shadow are still difficult to be correctly represented.

Another way to represent the behavior of illumination is the SH approach [2][16]. In computer graphics, it has been widely used for representing various spherical illumination properties, such as bidirectional reflection distribution function (BRDF) [37][38][14][39], distant environment [37], and precomputed radiance transfer (PRT) [15]. Earlier works in global illumination [38] have already made use of the SH basis for representing the BRDF of different surfaces. Recent development of PRT further improves the representation for incorporating self-shadowing and inter-reflection [39][40][15].

However, SH basis functions are defined in a mathematically complex way. For relighting of complex illumination configurations, these SH basis functions have to be evaluated several times for each pixel. The rendering time could be very long when large number of SH basis functions are used. Wong *et al.* [17] used hardware texture look-up (table look-up) on consumer-level graphics hardware to achieve real-time evaluation of these SH basis functions. However, this approach consumes a lot of resource on graphics hardware. If large number of SH basis functions are used, this approach may use up all memory on graphics hardware. In this situation, switching of active textures (tables) is needed and then slows down the performance. Motivated by the mathematical complexity of SH basis functions and the need of real-time relighting in games applications, we want to find a simpler and more efficient basis for approximating the spherical illumination information. The coding method utilizing this basis must also be rendering-friendly.

3 Computational Model

3.1 The Plenoptic Illumination Function

The PIF model [2], [4], [41] describes the appearance of a scene illuminated by a directional light source with different directions. To simplify our discussion, let us consider fixed view images. The PIF

$$P(x, y, \theta, \phi) = P(x, y, \vec{L}) \tag{1}$$

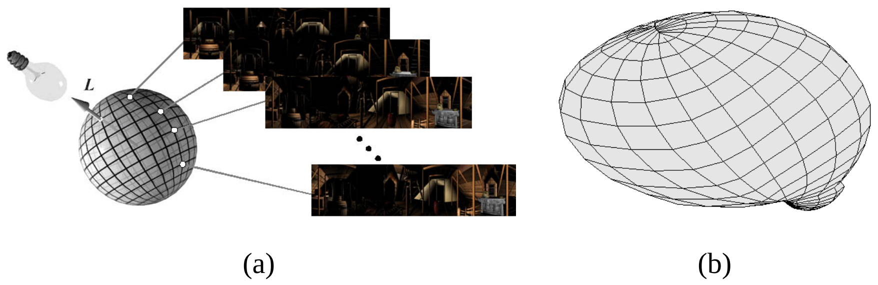


Fig. 1. (a) Sampling the lighting directions on the regular spherical grid. Each lighting direction is associated with a reference image; (b) Sampled radiance values of a pixel.

is a function of pixel position (x, y) in the view–plane and the lighting direction,

$$\vec{L} = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta\cos\phi \\ \sin\theta\sin\phi \end{pmatrix}. \tag{2}$$

The two parameters (θ, ϕ) correspond to the elevation and azimuth, respectively. The range of elevation is $[0, \pi]$ while that of azimuth is $[0, 2\pi]$. The PIF tells us the radiance value of a pixel located at (x, y) in the view–plane when the scene is illuminated by a directional light source with direction$\vec{L}$.

For a fixed position (x, y), the PIF is a spherical function of $\vec{L}$. In practice, we do not have the true continuous PIF. Instead, we can only obtain a set of sampled radiance values under a number of sampled light vectors $\vec{L}_i$. Sampling a PIF is actually a process of taking pictures. Every sampled light vector is associated with a reference image. That means, there is a set of reference images, $\{\text{Im}(\vec{L}_1), \cdots, \text{Im}(\vec{L}_M)\}$, captured at different lighting directions. For example, one can sample the PIF on a spherical grid (Figure 1).

3.2 Relighting from Reference Images

To relight a scene under a directional light source with a novel light vector$\vec{L}$, we first find out the n closest sampling light vectors to the novel light vector. Let $S_{\vec{L}}$ be the index set of the n sampled light vectors. The relit image is given by

$$\text{Im}(\vec{L}) = \sum_{i' \in S_{\vec{L}}} \alpha_{i',\vec{L}} \, \text{Im}(\vec{L}_{i'}) \tag{3}$$

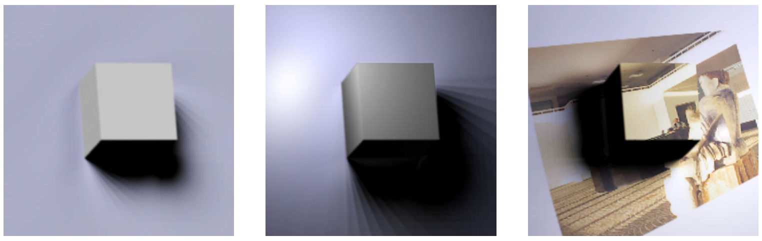
where $\alpha_{i',\vec{L}}$ is the weighting factor. The number of elements in $S_{\vec{L}}$, n, and the weighting factor, $\alpha_{i',\vec{L}}$, depend on the interpolation method used.

For complex illumination conditions, such as point light source and slide projector source, every pixel is associated with a distinct light vector. The light vector $\vec{L}_{x,y}$ of a pixel at (x, y) is also a function of

(x, y). The relighting equation at each pixel position (x, y) is given by

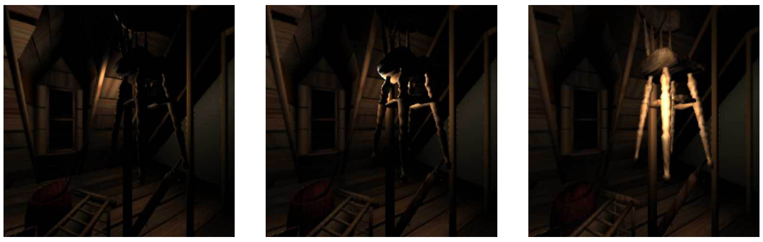
$$\mathrm{Im}(x, y, \vec{L}_{x,y}) = \sum_{i' \in S_{\vec{L}_{x,y}}} \alpha_{i',\vec{L}_{x,y}} \,\mathrm{Im}(x, y, \vec{L}_{i'}) \tag{4}$$

where $\mathrm{Im}(x, y, \vec{L}_{i'})$ denotes the pixel value at position (x, y) in the reference image $\mathrm{Im}(\vec{L}_{i'})$. Equation (4) tells us that different pixels in the relit image may be interpolated from different sets of reference images. As shown in Figures 2 and 3, we can synthesize not only a novel image illuminated by a novel directional light source but also an image illuminated by a non-directional light source, such as point light source. Interested readers are referred to [2][4] for relighting a scene under non–directional light sources.



(a) (b) (c)

Fig. 2. Relit images under (a) a directional light source, (b) a point light source, or (c) a slide projector. The behavior of a point source is similar to that of a light bulb.



(a) (b) (c)

Fig. 3. Relit images of a moving point light source that passes under the hanging chair.

Directly storing all reference images is very inefficient. For example, if the sampling rates along elevation and azimuth are equal to 30 and 40, respectively ($r_\theta \times r_\phi = 30 \times 40$), each pixel occupies about 3 KB storage space. A color data set with 512 $\times$ 512 resolution hence requires around 900 MB storage. Hence, a compression scheme with an efficient rendering ability is a must.

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

TABLE I

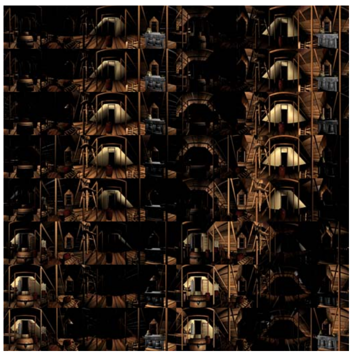
CHARACTERISTICS OF TESTED DATA SETS.

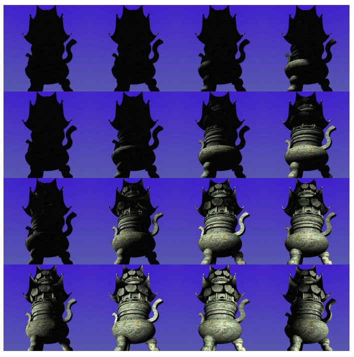
4 System Overview

4.1 Data Preparation

The input to our system is a set of reference images captured or synthesized under the illuminating directional light source with different lighting directions. Each of them corresponds to a distinct light vector $\vec{L}_i$ sampled on a unit sphere (Figure 1(a)). Each pixel hence is associated with a spherical function of radiance values. Figure 1(b) plots the spherical radiance function of a pixel. In fact, we make no assumption on the sampling pattern. The pattern can be a regular grid, random sampling, quasi Monte-Carlo sampling [42], or even scattered sampling. Lin *et al.* derived a theoretical sampling bound for relighting [43].

In our tested data, we used the regular grid to sample the illumination. So that, we can compare our result with that of the SH representation whose input is also regularly sampled. Throughout this paper, we use three synthetic data sets, *attic*, *ding*, and *forbid*. Table I summarizes the characteristics of each data set. Figure 4 shows some of the reference images. All images are rendered using 3D Studio Max. It should be noticed that our approach can apply to real data collected using various capture systems [44][45].


(a) *attic*


(b) *ding*

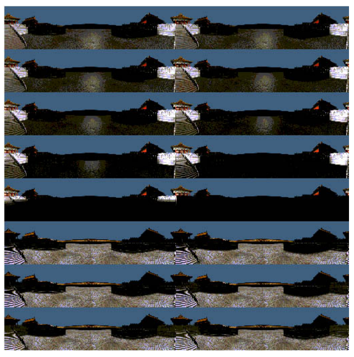

(c) *forbid*

Fig. 4. Some reference images in the three synthetic data sets.

4.2 Encoding

The proposed approach is a two-level compression method shown in Figure 5. In the first-level compression (Figure 5(a)), we use a SRBF network to approximate the spherical function associated with each pixel. The training data of each pixel are its sampled radiance values. Hence, for each pixel, we estimate its SRBF weights (weight vector). The details of the SRBF network model and SRBF weight estimation method are discussed in Section 5.

To exploit the spatial correlation in the second-level compression, we rebin those weight vectors, so that the first weights in all vectors are grouped to form a map, called *relighting map*. The same rebinning process is applied to all other weights to form k relighting maps (right hand side of Figure 5(a)). Those relighting maps represent the illumination property in the form of textures. As each relighting map is an image of real values (can be positive, zero or negative), they are further compressed by a wavelet-based method (Section 6), as illustrated in Figure 5(b).

4.3 Relighting

The rendering process (Figure 6) consists of two phases. In the first phase, the compressed relighting maps are decompressed and loaded into graphics hardware as textures. In the second phase, the novel image under a desired illumination condition is directly relit from these relighting maps without reconstructing any reference images. The relighting process is carried out in the pixel level (per-pixel). The per-pixel relighting process will be discussed in Section 7.

5 SRBF Networks for Approximating Spherical Functions

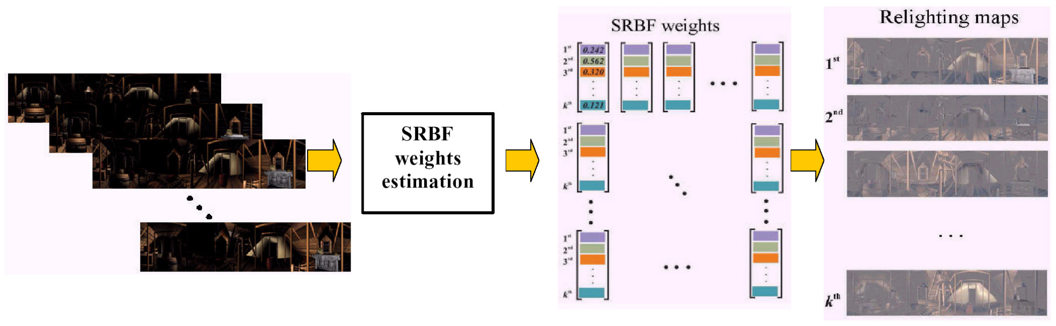
5.1 SRBF Networks

RBF networks have been used for function approximation and pattern recognition in various previous works [21][22][46][47][48]. A single output SRBF network [19][20] (Figure 7) consists of three layers: an input layer, a hidden layer with k SRBF nodes, and an output layer with one linear neuron. The network output is given by

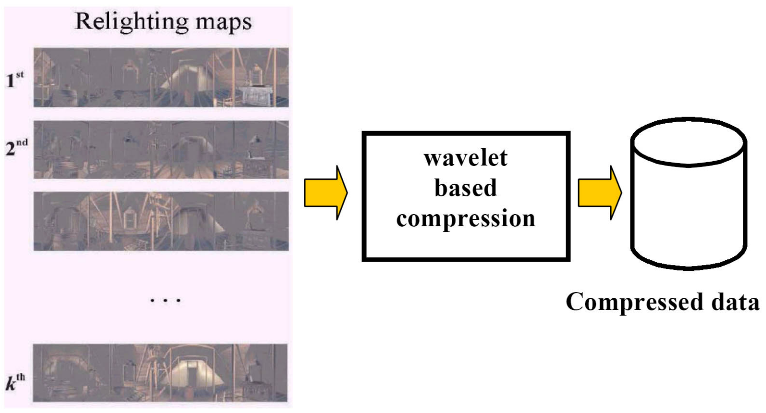
$$F(\vec{L}) = \sum_{i=1}^{k} w_i G_i(\vec{L})\,. \tag{5}$$

The output of the i-th SRBF node is given by

$$G_i(\vec{L}) = \exp\left\{ -\frac{d(\vec{L},\vec{C}_i)^2}{2\Delta^2} \right\}\,, \tag{6}$$



(a)



(b)

Fig. 5. Our encoding scheme consists of (a) the first-level compression, and (b) the second-level compression.

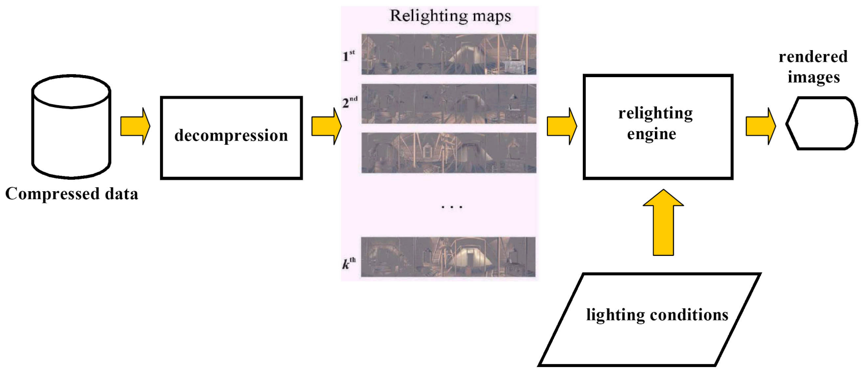


Fig. 6. The relighting process.

where Δ is the spread of the SRBF, unit vectors $\vec{C}_i$'s specify the centers of SRBF nodes (points on a spherical surface), and the distance function $d(\vec{L}, \vec{C}_i)$ is the distance between two unit vectors on a sphere, given by

$$d(\vec{L}, \vec{C}_i) = \cos^{-1}(\vec{L}^T \vec{C}_i)\,. \tag{7}$$

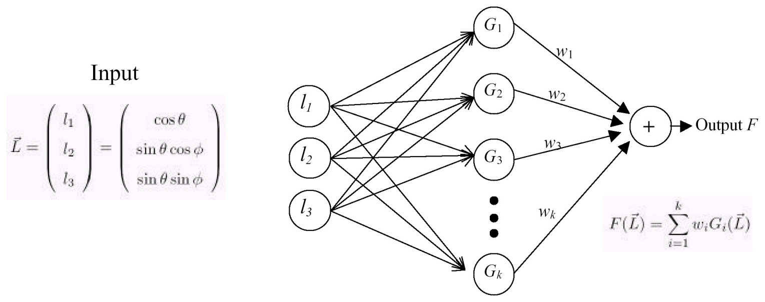

Input
$\vec{L} = \begin{pmatrix} l_1 \\ l_2 \\ l_3 \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta\cos\phi \\ \sin\theta\sin\phi \end{pmatrix}$
l_1
l_2
l_3
G_1
G_2
G_3
G_k
w_1
w_2
w_3
w_k
+
Output F
$F(\vec{L}) = \sum_{i=1}^{k} w_i G_i(\vec{L})$

Fig. 7. A single output SRBF network. Note that in our application, each pixel is associated a SRBF network.

For a fixed position (x, y), the PIF is a spherical function of $\vec{L}$, denoted as $P(\vec{L})$. We use a SRBF network to approximate $P(\vec{L})$, given by

$$P(\vec{L}) \approx F(\vec{L}) = \sum_{i=1}^{k} w_i G_i(\vec{L})\,. \tag{8}$$

Figure 8 illustrates the physical meaning of such approximation. Each SRBF node is responsible for capturing a region of the spherical function (range of the lighting directions). Of course, the more SRBF nodes we use, the better the approximation is (Figure 9). In our test cases (in Section 8.1), 25-40 SRBF nodes are sufficient for approximation. To build a SRBF network, we first select a set of centers$\vec{C}_i$'s and the spread parameter Δ. Secondly, we estimate the SRBF weights for each pixel.

In the classical RBF model (the input domain is defined over a rectangular system), if the input dimension is low [49][50], we can use a systematic method to estimate the centers and the spread based on some *a priori* information. The basic concept in [49][50] is that RBF centers should be uniformly distributed in the input space and the spread is the distance between two closest centers. With this approach, the estimation of RBF weights becomes a linear optimization problem that can be solved efficiently.

Another approach is to train those centers and the spread parameter based on the input-output training samples [46][47][48][51][52][53]. However, this training approach is not suitable for IBR. It may not take too long time to solve a single nonlinear problem on modern computers. However, there are massive number of pixels in IBR and we need to construct a SRBF network for each pixel. For example, if the

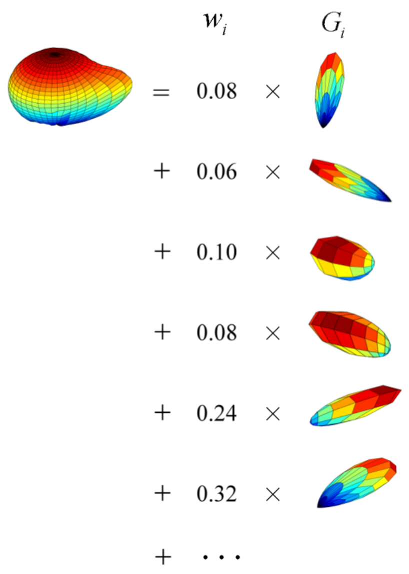
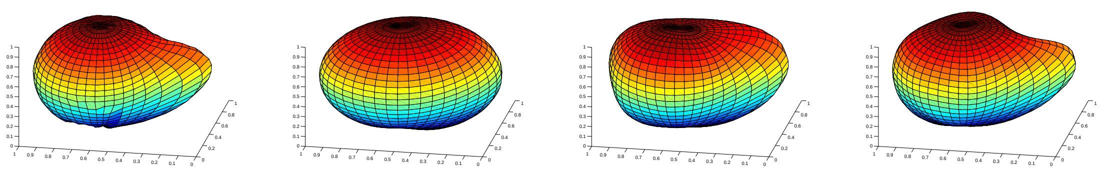


Fig. 8. Approximation of a spherical function by a SRBF network.



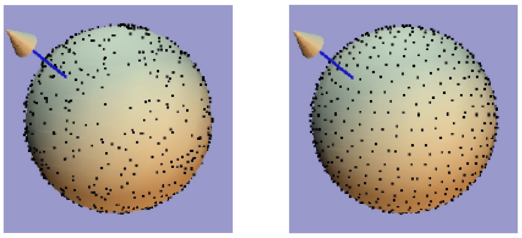
(a) original (b) 10 SRBF nodes (c) 25 SRBF nodes (d) 35 SRBF nodes

Fig. 9. The approximated PIF using different number of SRBF nodes.

spatial resolution is 512×512, we need to solve 262,144 nonlinear problems. More seriously, if the training approach is used, different pixels may have different centers and spreads. Sometimes, different pixels may even have different number of SRBF nodes. That means we need to store the centers and spreads for each pixel and hence introduce too much overhead. Furthermore, the second-level compression cannot be efficiently applied to the relighting maps and the relighting process will become too complex.

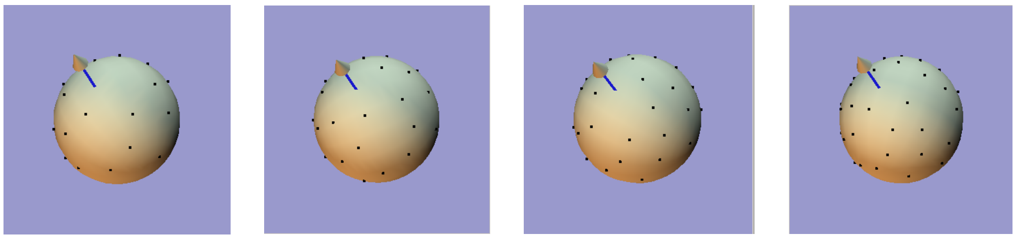
In this paper, we adapt, instead of directly apply, the approach used in [49], since the input space of a SRBF network is spherical. Similar to [49], we first generate a set of uniformly distributed SRBF centers on the unit sphere. The regular spherical grid cannot provide a uniformly distributed point set. The density of SRBF centers near the equator is much smaller than the density near the two poles. One may suggest to randomly distribute the SRBF centers (Figure 10(a)). However, some SRBF centers are too close to each other.

Wong *et al.* [42] used a quasi Monte-Carlo method to generate some pseudo uniformly distributed



(a) Random points (b) Hammersley points

Fig. 10. Random versus Hammersley points on the sphere.



(a) 20 points (b) 25 points (c) 30 points (d) 35 points

Fig. 11. Hammersley points used in this paper. Those points (SRBF centers) are distributed on the upper hemisphere. Note that they are SRBF centers not sampling lighting directions.

points, namely Hammersley points, on a sphere. As shown in Figure 10(b), Hammersley points are more uniformly distributed. In fact, the distribution of Hammersley points have been proved to be more uniform [54]. The SRBF centers used in this paper are shown in Figure 11.

We also set the spread parameters Δ of all SRBF nodes to be the same and equal to the average distance from a center to its closest neighboring center. Therefore, we can ensure two adjacent SRBF centers are not too close. Also, two adjacent SRBFs have certain overlap. With this setting, every pixel has the **same set of centers** and the **same spread**. This can minimize the storage overhead. Also, we only need to keep k SRBF weights for each pixel. The same-order SRBF weights from different pixels form a meaningful relighting map. Hence the second-level compression which exploits the spatial correlation can effectively compress relighting maps.

5.2 *Estimation of SRBF Weights*

5.2.1 The Unconstrained Solution

Once the centers and the spread parameter are selected, we can estimate the SRBF weights for each pixel. Consider a pixel, given M sampled radiance values $\vec{\beta} = \left[P(\vec{L}_1), \cdots, P(\vec{L}_M)\right]^T$, the SRBF

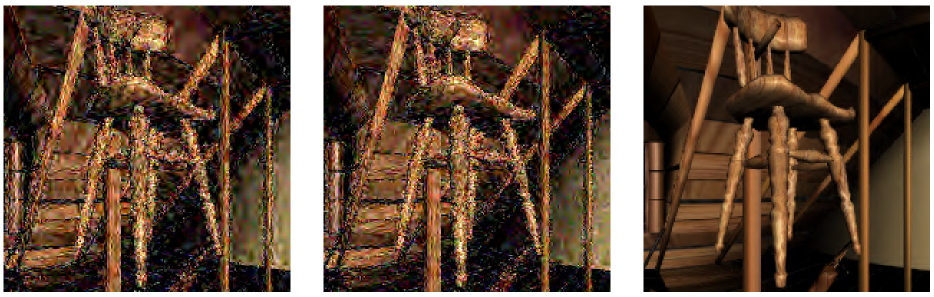


Fig. 12. Relit images from the compressed SRBF relighting maps of the data set *attic*. (a) The unconstrained least square solution with the same bit rate assigned for each relighting map. (b) The unconstrained least square solution with different bit rates assigned to different relighting maps. (c) The constrained least square solution.

weights $\vec{w} = [w_1, \cdots, w_k]^T$ is the least square solution [55][56] of the following linear equation

$$\vec{\beta} = \mathbf{G}\,\vec{w}\,, \tag{9}$$

where

$$\mathbf{G} = \begin{bmatrix} G_1(\vec{L}_1) & \cdots & G_k(\vec{L}_1) \\ \vdots & \ddots & \vdots \\ G_1(\vec{L}_M) & \cdots & G_k(\vec{L}_M) \end{bmatrix}.$$

We can use unconstrained least square methods [55] to obtain the solution, given by

$$\vec{w} = (\mathbf{G}^T\mathbf{G})^{-1}\,\mathbf{G}^T\vec{\beta}\,. \tag{10}$$

However, with this unconstrained solution, relit images are very sensitive to the quantization noise in the relighting maps. Such noise is introduced by the second-level compression which further compresses these relighting maps. Figure 12(a) shows a typical relit image from the compressed relighting maps. Annoying visual artifact due to noise is observed. The weights in the maps are obtained by the least square method and the same bit rate is assigned to all relighting maps. One may suggest to assign different bit rates to different relighting maps according to their variances. However, this approach does not reduce the artifact as evidenced in Figure 12(b). If we constrain the magnitude of SRBF weights during the least square fitting, we can effectively noise-proof the rendering result as shown in Figure 12(c). Even the same compression ratio is applied, there is no visual artifact observable.

5.2.2 Noise in Relit Images

To suppress the artifact in relit images, we first study how the quantization noise in SRBF weights w_i's affects the rendering quality. We model the noise as an additive noise with multiplicative effect. The quantized weight $\hat{w}_i$ is modeled as

$$\hat{w}_i = w_i + \delta_i w_i \qquad \forall i \tag{11}$$

where δ_i's are independently identical random variables with zero mean and variance σ_δ^2. Its probability density function is symmetric. In (11), the second term on the right hand side represents the quantization noise on SRBF weights. We assume that the quantization noise is proportional to the magnitude of the weights w_i's. This assumption is reasonable as it matches several standard quantization schemes. For example, in the low-precision floating point representation, the precision error is proportional to the magnitude of the number. Another example is the uniform quantization. From the rate distortion theory [57], the quantization error of a data source under the uniform quantization is proportional to the variance, as well as, the magnitude of the data source.

From (11), it is not difficult to see that the average extra error E_q on the spherical function due to the quantization of SRBF weights is given by

$$E_q = E\left[\left(\sum_{i=1}^{n} \delta_i w_i \int_\Omega G_i(\vec{L})\, d\vec{L}\right)^2\right], \tag{12}$$

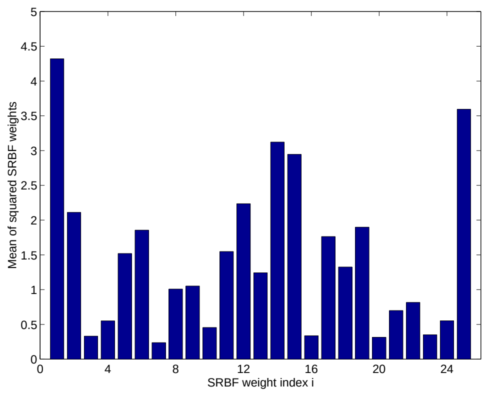
where Ω denotes the spherical surface and $E[\cdot]$ is the expectation operator. Since δ_i's are independently identical random variables with zero mean and variance σ_δ^2, the average extra error can be rewritten as

$$E_q = \sum_{i=1}^{k} \sigma_\delta^2 w_i^2 \int_\Omega G_i^2(\vec{L})\, d\vec{L}\,. \tag{13}$$

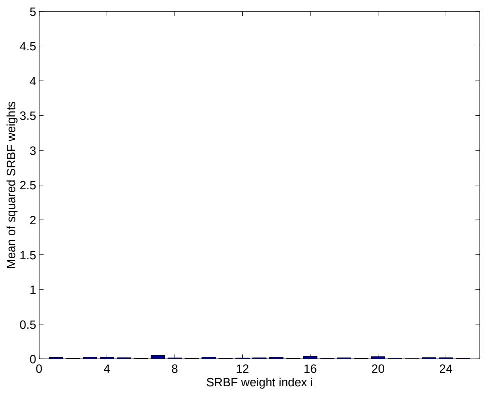
The variance σ_δ^2 is controlled by the bit rate in the second-level compression on w_i's. From (13), the quantization error of the PIF is proportional to w_i^2's. Hence, controlling the magnitude of the estimated SRBF weights is very important even if the input sampled data is noise-free.

5.2.3 Magnitude of SRBF Weights

Figure 13 compares the average magnitudes of SRBF weights obtained by the unconstrained and constrained least square methods. It plots the average of squared SRBF weights over all pixels against the index of SRBF weights. The magnitude of SRBF weights obtained by the unconstrained least square method is very large. The range of values is 0.2377-4.3204. Hence the relit images (Figure 12(a)-(b)) are very noisy. On other hand, the value range of the constrained solution is 0.0033-0.0499. Since the minimum value of the unconstrained solution is even greater than the maximum value of the constrained solution, the quality of relighting from unconstrained weights cannot be improved (Figure 12(b)) by allocating bits according to the variances.



(a)



(b)

Fig. 13. Average values of squared SRBF weights over all pixels in the data set *attic*. (a) Unconstrained least square solution. (b) Constrained least square solution.

5.2.4 Magnitude Constraint

Since we use the same spread Δ for all SRBF nodes, $\int_\Omega G_i^2(\vec{L})\,d\vec{L}$'s are equal to a constant ρ for all i. Therefore, (13) can be rewritten as

$$E_q = \sum_{i=1}^{k} \sigma_\delta^2 w_i^2 \rho = \sigma_\delta^2 \rho \vec{w}^T \vec{w}\,. \tag{14}$$

In (14), the parameter σ_δ^2 can be considered as the percentage error in SRBF weights. The constraint we impose is that given σ_δ^2 percentage error in weights, the extra error E_q due to the quantization noise should be less than or equal to σ_δ^2 times the energy of PIF, given by

$$\begin{aligned} \sigma_\delta^2 \rho \vec{w}^T \vec{w} &\leq \sigma_\delta^2 \times \text{Energy} & (15)\\ \vec{w}^T \vec{w} &\leq N_o, & (16) \end{aligned}$$

where $N_o = \frac{\text{Energy}}{\rho}$ and the energy can be estimated from the sampled values $P(\vec{L}_i)$'s. The above constraint means that the squared sum of weights should be bounded by N_o.

5.2.5 The Constrained Solution

Now, we need to minimize the constrained least square cost function, given by

$$J_2(\vec{w}) = ||\vec{\beta} - \mathbf{G}\,\vec{w}||_2^2 + \lambda(\vec{w}^T \vec{w} - N_o), \tag{17}$$

where $\lambda > 0$. The constrained solution is

$$\vec{w} = \left(\mathbf{G}^T\mathbf{G} + \lambda I_{k\times k}\right)^{-1} \mathbf{G}^T \vec{\beta}\ , \tag{18}$$

where $I_{k\times k}$ is a $k \times k$ identity matrix, and λ should satisfy the following inequality

$$\vec{\beta}^T \mathbf{G} \left(\mathbf{G}^T\mathbf{G} + \lambda I_{k\times k}\right)^{-2} \mathbf{G}^T \vec{\beta} \leq N_o \ . \tag{19}$$

We can apply the singular value decomposition (SVD) to $\mathbf{G}^T\mathbf{G}$:

$$\mathbf{G}^T\mathbf{G} = USU^T \ , \tag{20}$$

where U is a unitary matrix and S is a diagonal matrix containing non-negative singular values. Afterwards, (19) can be rewritten as

$$\vec{\beta}^T \mathbf{G}' \ (S + \lambda I_{k\times k})^{-2} \, \mathbf{G}'^T \vec{\beta} \leq N_o \, , \tag{21}$$

where $\mathbf{G}' = \mathbf{G}U$. Note that every pixel is associated with the same $\mathbf{G}'$, as well as, the same S and U, which can be precomputed. Our problem is to *iteratively* search the smallest value of λ such that (21) is satisfied. We can assign an initial value of λ and then use a simple binary search to estimate a suitable value of λ based on (21). Afterwards, we can determine the SRBF weights using (18). The calculation in (21) is very simple because the matrix S is a diagonal matrix. Since only the first few diagonal elements of S are important, the calculation in (21) can be further sped up.

To control the magnitude of SRBF weights, we may also consider the concept of regularizers [56][58][59]. In fact, we have tested the performance of the regularization approach. Its performance is quite similar to that of our constrained least square approach. Both approaches can successfully control the magnitude of SRBF weights. However, our problem size is enormous as we need to solve hundreds of thousands fitting, the regularization approach may not be fast enough for practical applications. Instead, the proposed method is a fast solution which can solve hundreds of thousands fitting in around 10 minutes on modern PCs (Pentium IV 1GHz CPU).

6 The Second-Level Compression: Relighting Map and Wavelet Coding

After the estimation, the PIF of a pixel is represented by k SRBF weights:

$$P(x, y, \vec{L}) \approx \sum_{i=1}^{k} w_i(x, y) G_i(\vec{L}) \tag{22}$$

where $w_i(x, y)$'s are SRBF weights of a pixel at position (x, y). They form a weight vector $\vec{w}(x, y)$. It is natural to compress weight vectors using standard vector quantization (VQ) [60][61]. However, the training time of VQ is prohibitively long when the input data size is large, just like our case.

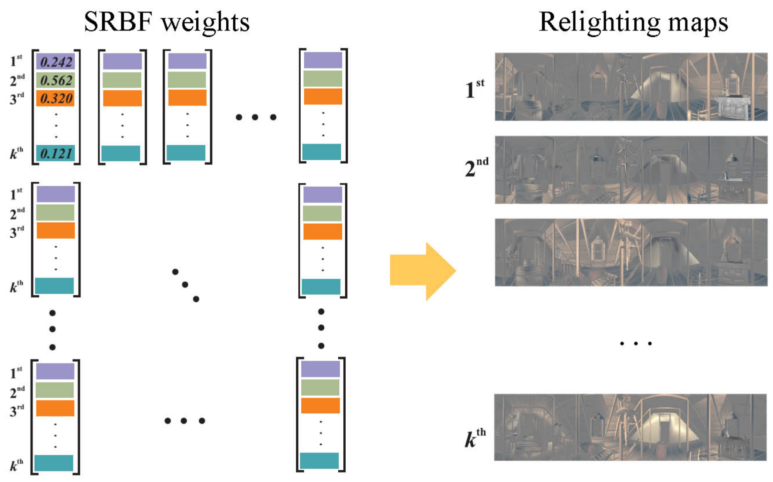


Fig. 14. Picking SRBF weights and forming relighting maps.

Instead, we pick the i-th SRBF weights from all pixels and form the i-th relighting map, denoted as $\mathbf{W}_i$. Each weight is placed at its corresponding pixel position. The result is a set of k relighting maps. If color images are considered, a relighting map consists of three color channels. Figure 14 illustrates the formation of relighting maps for the data set *attic*. Interestingly, each relighting map is in fact a floating-point image (the right hand side of Figure 14). Hence it is possible to compress these relighting maps with image compression methods. In this paper, we propose to use wavelets.

The wavelet technique used for compressing relighting maps is very similar to the new image coding standard of JPEG2000 [12][62]. The major difference is that we now encode the floating-point relighting maps instead of 8-bit integer natural images. We apply the symmetric wavelet transform (SWT) with 9/7 Daubechies coefficients to every relighting map. In this paper, we modify the EPWIC codec [63], which is originally designed for natural images, to compress floating-point images. The use of EPWIC codec is to demonstrate that relighting maps can be effectively compressed by wavelet–based methods. We expect that more advanced wavelet embedded methods may produce even better result. The target bit rate of each relighting map is set to be the same. This is because each relighting map corresponds to one SRBF. As each SRBF is responsible for representing the lighting property over a certain range of lighting directions, there is no way to differentiate the bit rates for different relighting maps unless we know *in prior* the exact trajectory of the light source during relighting. Hence we set an equal target bit rate for all relighting maps.

Before we pass those k color relighting maps to the wavelet codec, we first convert all relighting maps from the RGB to YUV color space [64]. So that we can allocate more bits for human-sensitive luminance component while less bits for human-insensitive chrominance components during compression. The bit ratio for YUV channels is 2:1:1. After compression, we measure the overall bit rate defined as

$$\text{Overall bit rate} = \frac{\text{Compressed file size}}{\text{Number of reference images } \times \text{ Number of pixels}}. \tag{23}$$

7 Relighting

As mentioned in Sections 1 and 2, standard image compression methods are not suitable to handle an IBR data set. If all reference images are held in uncompressed format in the computer memory, they occupies too much memory space. If they are held in the compressed format, the rendering speed may not be fast enough for complex illumination conditions because we need to decompress a large number of reference images.

With our proposed two-level compression scheme, users can control lighting condition of a scene interactively. The process consists of two major phases. In the first phase, those k relighting maps are decompressed and loaded into the memory. In the second phase, once the user specifies a lighting condition, the image can be directly relit from those relighting maps, based on (22). In this section, we briefly discuss the relighting process for the directional and point light sources.

7.1 Directional Light Source

Relighting a scene is actually a reconstruction process that computes (22) for every pixel. For the directional light source, every pixel is illuminated by the same light vector. That is, all SRBFs in (22) are fed with the same $\vec{L} = \{\theta, \phi\}$ and then k SRBF basis values, $\{G_1(\vec{L}), \cdots, G_k(\vec{L})\}$, are calculated. Note that the evaluations of these k basis functions are the same for all pixels. Therefore, the relighting process (Figure 15) can be formulated as a linear combination of relighting maps $\{\mathbf{W}_1, \cdots, \mathbf{W}_k\}$ with weighting factors $\{G_1(\vec{L}), \cdots, G_k(\vec{L})\}$. Mathematically, the relit image is given by

$$\text{Im}(\vec{L}) = \sum_{i=1}^{k} G_i(\vec{L})\mathbf{W}_i. \tag{24}$$

Nowadays, many consumer-level GPUs (such as nVidia GeForceFX and ATI 9800) are designed to linearly combine images. GPUs are SIMD-based parallel CPUs. They can execute user-defined shader programs which are written in C-like language, such as high-level shading language (HLSL), and *C for graphics* (Cg) [65]. Due to the parallelizability of our relighting problem, we develop a set of shader programs to perform the relighting. To relight in real-time, the k relighting maps are initially loaded as

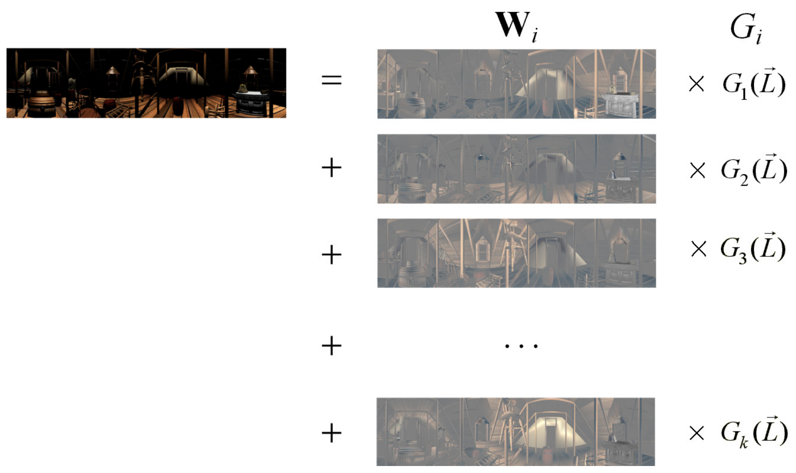


Fig. 15. Directional-source relighting as a linear combination of relighting maps.

textures on GPU. Whenever the user specifies a novel light vector, k basis values $\{G_1(\vec{L}), \cdots, G_k(\vec{L})\}$ are passed into a shader and the relighting maps $\mathbf{W}_i$ are blended (linearly combined) by hardware to obtain the relighting result.

7.2 Point Light Source

If the physical coordinate of surface point associated with every pixel in the scene is available, we can relight the scene by a point light source whose property is similar to a light bulb. The light vector of a pixel at (x, y) in the view–plane is computed by the following equation:

$$\vec{L}_{x,y} = \frac{\vec{S} - \vec{V}_{x,y}}{\|\vec{S} - \vec{V}_{x,y}\|_2} \tag{25}$$

where $\vec{S}$ is the position of the point source, and $\vec{V}_{x,y}$ is the physical coordinate of surface point associated with pixel (x, y). To model the distance fall-off effect of a point source, we can multiply an attenuation factor that is related to the distance between $\vec{S}$ and $\vec{V}_{x,y}$, given by

$$A_{x,y} = \frac{A_o}{\|\vec{S} - \vec{V}_{x,y}\|_2} \tag{26}$$

where $A_o > 0$ is an user defined parameter. The overall relighting equation for a pixel (x, y) with a point source at $\vec{S}$ is given by

$$\mathrm{Im}(x, y, \vec{S}) = \sum_{i=1}^{k} A_{x,y} G_i(\vec{L}_{x,y}) w_i(x, y) \quad , \tag{27}$$

where $w_i(x, y)$'s are the SRBF weights at pixel position (x, y). Let us denote the basis values as

$$B_i(x, y) = A_{x,y} G_i(\vec{L}_{x,y}) \, . \tag{28}$$

We arrange the basis values $B_i(x,y) = A_{x,y}G_i(\vec{L})$ in matrix form, denoted as $\mathbf{B}_i$. The relighting process shown in Figure 16 can be written in a more compact form

$$\mathrm{Im}(\vec{S}) = \sum_{i=1}^{k} \mathbf{B}_i \otimes \mathbf{W}_i \,. \tag{29}$$

where $\otimes$ is an element-by-element operator. Let Ξ and Ψ be two matrices. $D = \Xi \otimes \Psi$ means that $D_{ij} = \Xi_{ij}\Psi_{ij}$.

To relight in real-time, those k relighting maps and all physical coordinates are loaded into the texture buffers of GPU. Whenever the user specifies a new position of the point source, the position $\vec{S}$ is passed into the shader program. The shader program first computes those k basis values $B_i(x,y)$'s and then computes (29), in a *per-pixel* manner.

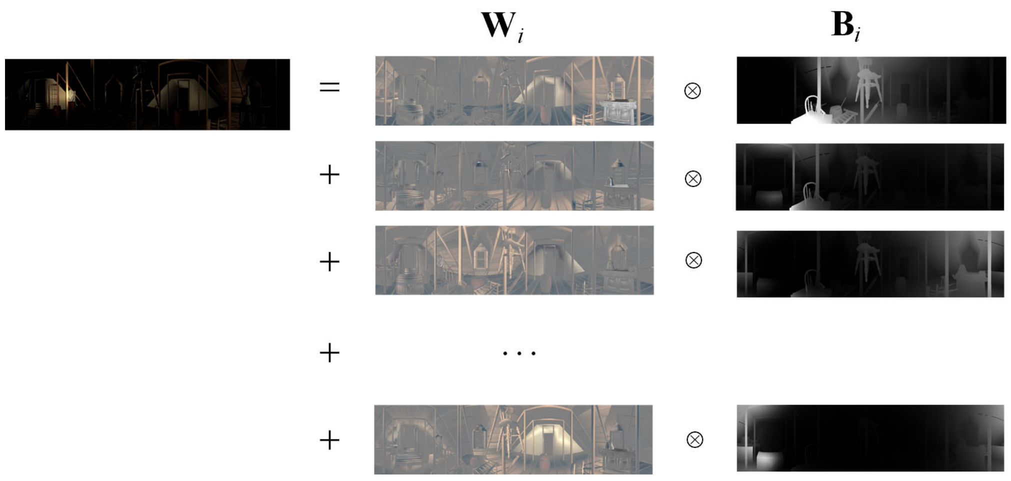
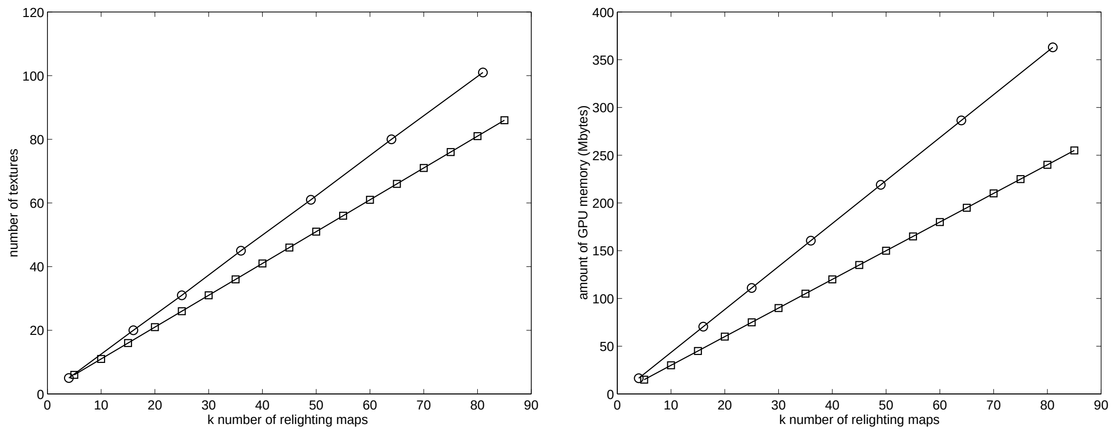


Fig. 16. Point-source relighting as an element-wise linear combination.

7.3 *Rendering Simplicity*

Compared with the SH approach, the SRBF rendering process is simpler. In the SH case, evaluating SH basis functions involves the evaluation of associated Legendre polynomials which are complex and defined recursively. Hence the relighting of non-directional light sources is very slow as we have to evaluate the SH basis functions for each pixel. One can use look-up tables to speed up. In the practical GPU implementation, we can store the look-up tables in the form of *cube-maps*. Cube-map is a kind of hardware-supported textures designed for looking up with a vector as the look-up parameter. A cube-map contains six faces. Each face is simply a square image. If the face resolution of the cube-map is 256×256 and k=25, the consumption of GPU on-board memory is already 36 MB. We ignore the



(a) count of active textures (b) Amount of memory consumed

Fig. 17. Comparison of GPU texture consumption in terms of (a) active texture count, and (b) amount of memory.

memory consumption of the first basis function since it is a constant function. Note that the size of GPU memory is usually much smaller than that of main memory. As an example, nVidia GeforceFX 5950 has 256MB memory.

Moreover, all GPUs have a limit on the number of textures being simultaneously active. For example, the maximum number of active textures on the current nVidia GeforceFX family is equal to 16 only. When all four channels (RGBA) of a cube-map are used, we can store 4 look-up tables in one cube-map texture. For $k = 25$ (excluding the first constant SH basis), the number of active textures consumed by SH basis function evaluation is already equal to 6. Since those cube-maps should be active during relighting, this reduces the number of relighting maps that can be loaded within a single *rendering pass* (one execution pass of a shader program). Hence, multiple rendering passes are needed. The rendering speed will be slowed down due to the overhead of each rendering pass.

On the other hand, the SRBF approach is much simpler because we only need to evaluate a function in the form of "$\exp\{-a\cos^{-1}(\cdot)\}$" for each SRBF. Again, we can precompute a *single* 1D look-up table common for all basis functions. Hence, we save a lot of GPU memory and the count of active textures. As an example, if the resolution of the 1D table is 2048, the table occupies only 8 KB storage, which is substantially smaller than a normal cube-map. Moreover, the 1D table occupies only one active texture count. Such compactness is important for game development as a large number of textures and texture memory are utilized in a computer game. Figure 17 summarizes the total amount of GPU memory used by SH and SRBF approaches, including cube-maps and 2D textures. It can be seen that the SH approach consumes more GPU memory than the SRBF approach.

8 Simulation Results

8.1 Number of SRBF nodes

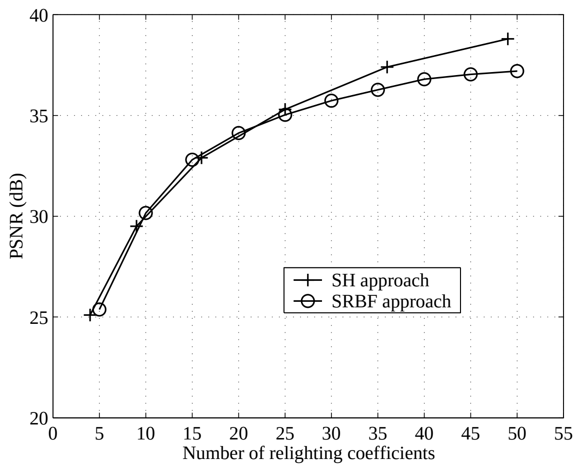
To evaluate the performance of the first-level compression, we measure the peak signal-to-noise ratio (PSNR) of the reconstructed reference images against the number k of relighting maps used. As a comparison, we also show the result of the SH approach. The SH coefficients can also be organized in the form of relighting maps [10], just like SRBF. The result is summarized in Figure 18. The PSNR of the SRBF approach consistently improves with the increase of the number of SRBF nodes. However, the improvement in PSNR slows down after around 25 coefficients. For data set *forbid*, the performance of our approach is better than that of the SH approach. For data sets *attic* and *ding*, our performance is slightly poorer than that of the SH approach. In general, the overall performance of the SRBF approach is comparable to that of the SH one.

Since many SH-based algorithms for IBR [9], [10], [40] suggested to use 25 SH coefficients for approximating the lighting property, we use the PSNR values at 25 SH coefficients as reference values. For data sets *attic* and *forbid*, we achieve comparable performance using around 25 SRBF nodes. For data set *ding*, the SRBF approach requires more nodes to achieve the same PSNR. However, the difference in PSNR for 25 SRBF and 25 SH coefficients is only 1.4 dB. To sum up, 25 to 40 SRBF nodes are sufficient to approximate the lighting property of a pixel. Although the first-level compression performances of SRBF and SH approaches are similar, the rendering time of the SRBF approach is much smaller than that of the SH approach.

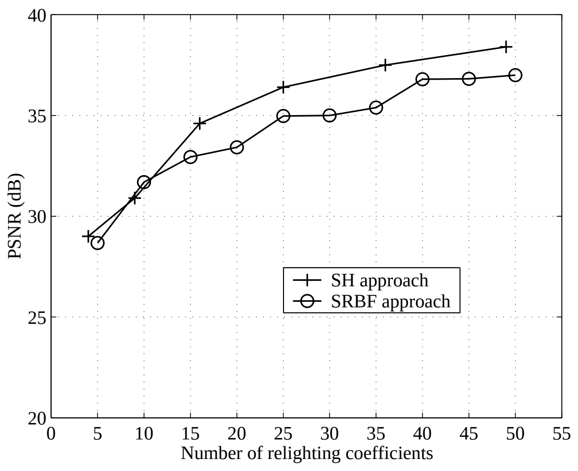
8.2 Unconstrained and Constrained Solutions

After the second-level compression, there are substantial difference between the constrained and unconstrained solutions of SRBF weights. Let us consider data set *attic*. As suggested before, we estimate 25 SRBF relighting maps in the first-level compression. During the second-level compression, we compress those SRBF relighting maps with EPWIC algorithm. We try different code rates for the EPWIC codec.

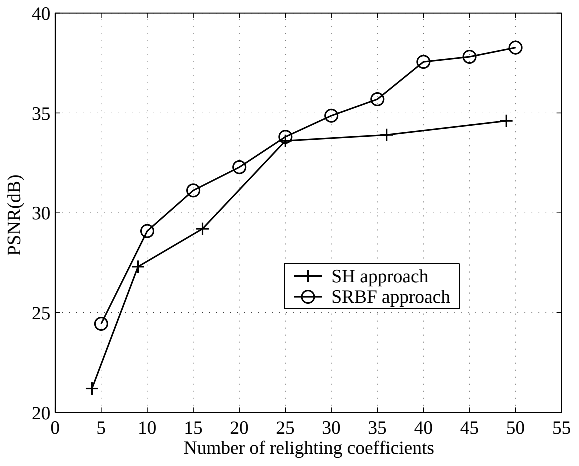
Two different bit allocation methods are experimented on these two sets of 25 relighting maps. In the first method, we assign the same target bit rate to all relighting maps. This scheme is the same as the method used in [10]. The reason behind this scheme is that each relighting map accounts for representing a region of the illumination sphere. Unless we know the trajectory of the illuminating light source, we can only assume all directions (all relighting maps) are equally important.



(a) *attic*



(b) *ding*



(c) *forbid*

Fig. 18. The performance of the first-level compression.

In the second method, we assign different bit rates for these relighting maps according to their variances. Figure 19 plots the variances of SRBF weights of all relighting maps. Note that the plots are very similar to the mean of squared weights in Figure 13. The optimal bit allocation scheme [57] is used. That is, we first define a target average bit rate. Based on the optimal bit allocation scheme, every relighting map has its own target bit rate and is then compressed based on this rate.

We measure the PSNR values of the reconstructed reference images under various overall bit rates. The result is summarized in Figure 20. The performance of the constrained least square is much better than that of the unconstrained one. The constrained least square method produces much smaller magnitude SRBF weights than the unconstrained one (Figure 13), hence it is less sensitive to noise.

As shown in Figure 13, the variation in variances is not too large, therefore bit rates allocated for

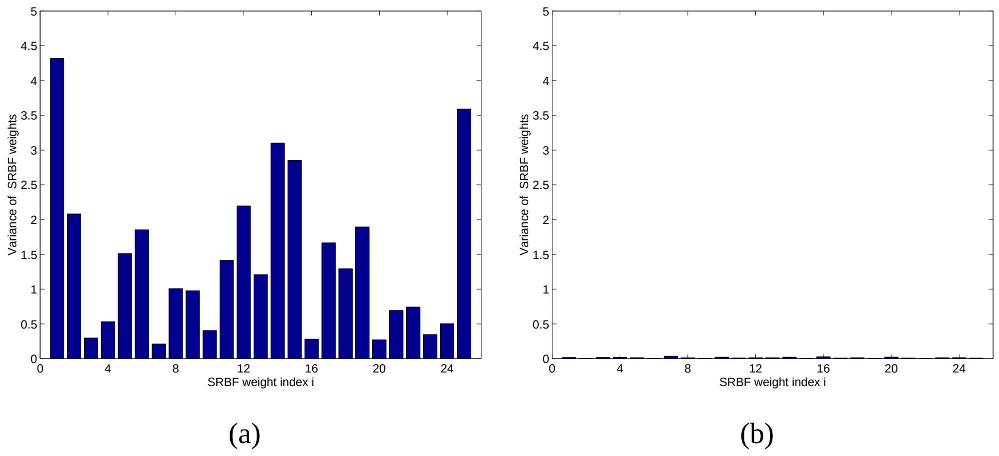


Fig. 19. Variances of SRBF weights (relighting maps) estimated by (a) unconstrained least square method, and (b) constrained least square method.

different relighting maps are quite similar if the optimal bit allocation is used[1]. Therefore, assigning different bit rates for different relighting maps cannot effectively solve the noise-sensitivity problem of the unconstrained least square solution (as evidenced in Figure 20). Another reason is that SRBFs are not an orthogonal set. Hence, the optimal bit allocation for different relighting maps may not improve the PSNR value.

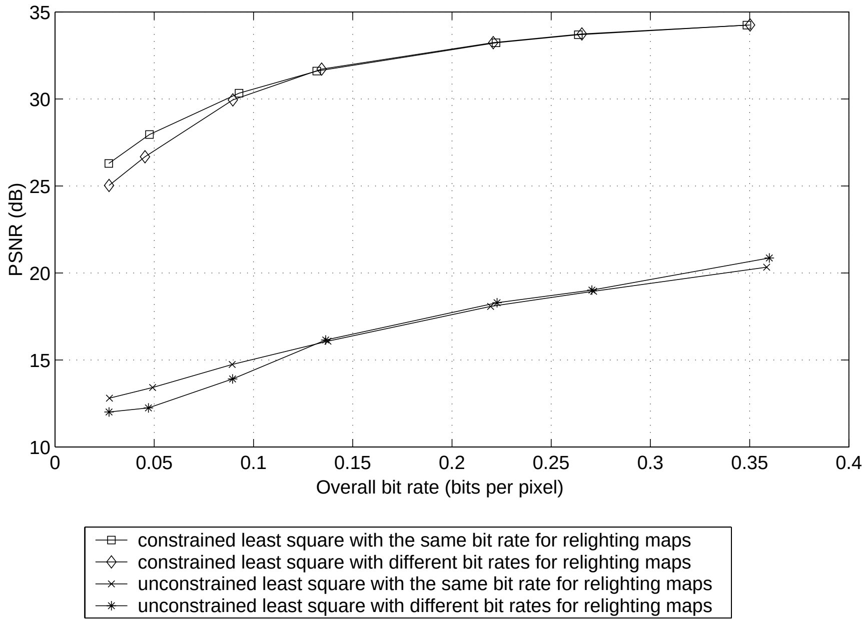


Fig. 20. The performance comparison after the second-level compression.

[1]In the case of subband code, the variation in variances for different subbands are very large.

8.3 Overall Performance

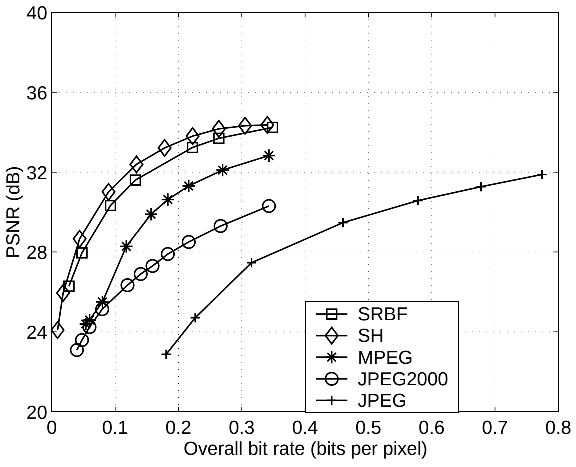
In this section, we compare the proposed method with standard image compression methods and the SH method. To have a fair comparison, we ensure the first-level compression in both SRBF and SH approaches achieve the same image quality. The image quality achieved by the first 25 SH functions are used as reference as it is commonly adopted in previous works [9], [10], [40]. Based on Figure 18, we use 25 SRBF relighting maps for data sets *attic* and *forbid* and 40 SRBF maps for data set *ding*. Hence, the distortion due to the first-level compression is quite similar in both SH and SRBF approaches. Figure 21 shows the PSNR of the reconstructed data sets.

The standard image compression methods, JPEG (Matlab JPEG codec) [11], JPEG2000 (a popular JPEG2000 implementation, JasPer) [62], and MPEG (LSX-MPEG Encoder 3.5) [13], are considered. In JPEG and JPEG2000, we compress each reference image individually. To compress the data set with MPEG, we need to order all reference images to form an 1D sequence, such that two consecutive frames are coherent. We use the following ordering method. Reference images with the same elevation θ form a group of frames. Within a group, reference images are ordered according to their values of ϕ. Finally, groups of frames are ordered according to their values of θ.

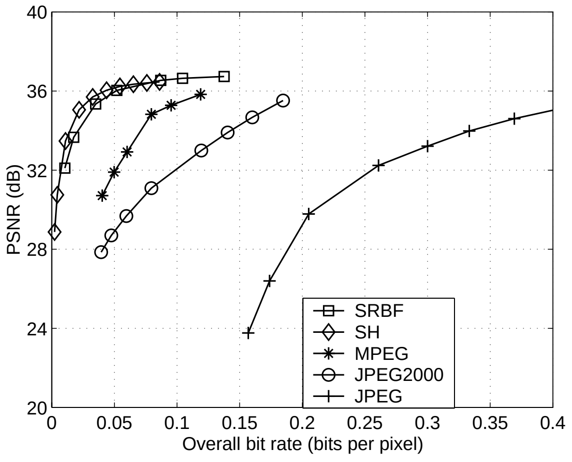
In Figure 21, the PSNR of all methods consistently improves as the bit rate increases. Among the methods, the SH and SRBF approaches out-perform other standard image and video coding methods. Also, the SH and SRBF approaches have a very similar overall performance. For both SH and SRBF approaches, there seems to have a point of saturation. When the bit rate reaches certain point (e.g. 0.05 bit for data set *ding*), the PSNR only improves slightly. This is because the overall PSNR values are bounded by the distortion introduced by the first-level compression. For example, in *ding*, the PSNR after the first-level compression is around 36.8 dB. Hence, the overall PSNR cannot be better than this value.

Our method is much better than the standard image compression methods. For example, in *ding*, when the bit rate is around 0.05, the PSNR of our method is around 36 dB while the PSNR of JPEG, JPEG2000 and MPEG, PSNR values are much smaller than 33 dB. The performance of the SRBF approach is only a bit lower than that of the SH approach. But the advantage of the SRBF approach is that its rendering is more efficient (discussed in Section 8.5).

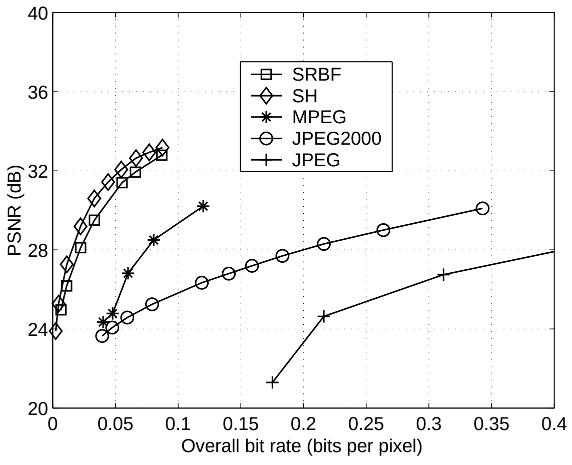
Figure 22 visually compares the reconstructed images. The visual quality of reconstructed images from JPEG (bit rate = 0.2051, Figure 22(b)) and JPEG2000 (bit rate=0.0602, Figure 22(c)) encoded data are very poor. The reconstructed image from MPEG (bit rate = 0.0602, Figure 22(d)) contains blocking and



(a) *attic*



(b) *ding*



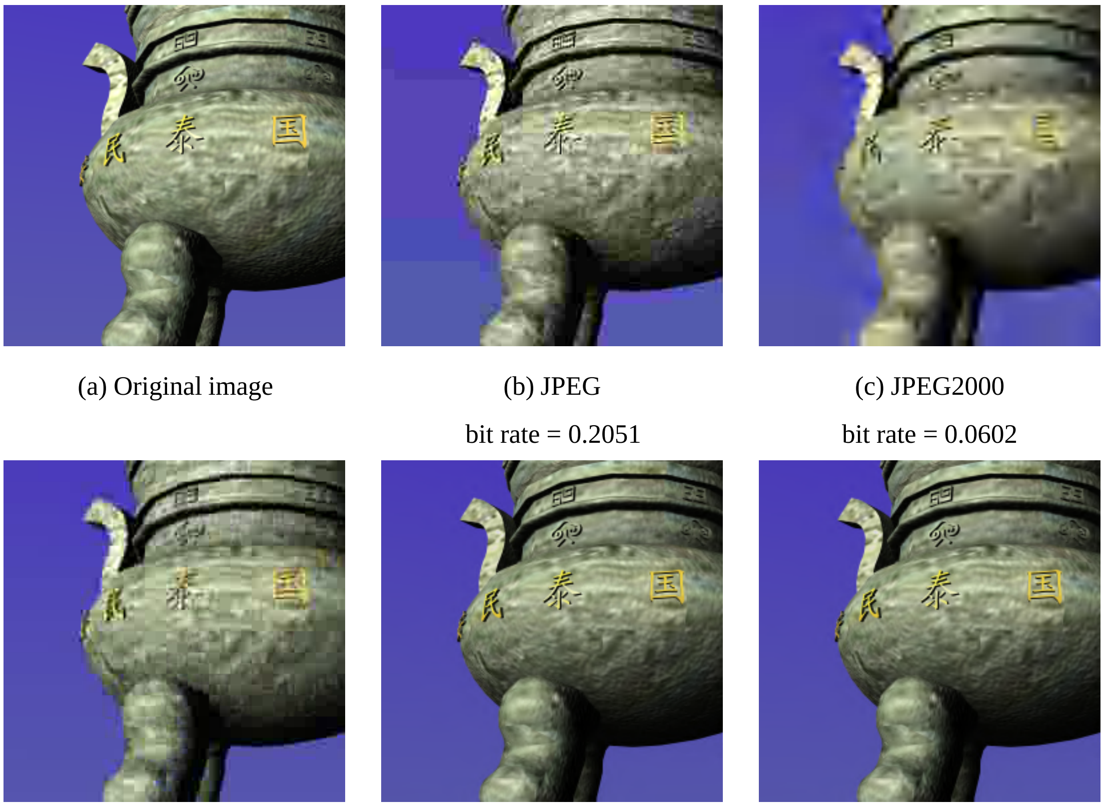
(c) *forbid*

Fig. 21. The overall PSNR in the reconstructed reference images.

blurring artifact. On the other hand, the reconstructed image from data encoded by our SRBF method (bit rate = 0.0519, Figure 22(e)) contains no significant visual artifact. Similar result is obtained from the SH approach (bit rate = 0.0544, Figure 22(f)).

8.4 Glossy Surface

To investigate the capability of our method in handling glossy surface, we modify data set *ding* to increase its specularity. A new set of reference images, *glossy ding*, is synthesized (Figure 23). We encode this new data set with both SRBF and SH approaches using 49 coefficients. For comparison, we use a point light source to relight the scene in order to emphasize the specular highlight. The control images are obtained by software interpolating the reference images as described in Section 3.2.



(a) Original image

(b) JPEG
bit rate = 0.2051

(c) JPEG2000
bit rate = 0.0602

(d) MPEG
bit rate =0.0602

(e) SRBF
bit rate = 0.0519

(f) SH
bit rate = 0.0544

Fig. 22. Visual comparison of the reconstructed images.

For this data set, measuring PSNR of relit images may not be very meaningful as the high-frequency specular highlight contributes only a small portion of the total energy. Therefore, we visually evaluate the ability in preserving specular highlights. Figure 23 compares the relit images. For each method, two relit images are obtained by positioning the point light source at two positions. Compared with the control images (Figure 23(a)), the specular highlights in images relit from both SH and SRBF encoded data are blurred and broaden. The result from SH and SRBF approaches are very similar.

8.5 Speed

For comparison of speed, we have implemented two relighting engines, one for SRBF and the other for SH. Both of them are GPU-accelerated and tailored for point-source relighting (which needs more computation). Standard image and video coding methods including JPEG, JPEG2000 and MPEG are not compared because they are not rendering-friendly and cannot achieve any performance close to interactive.

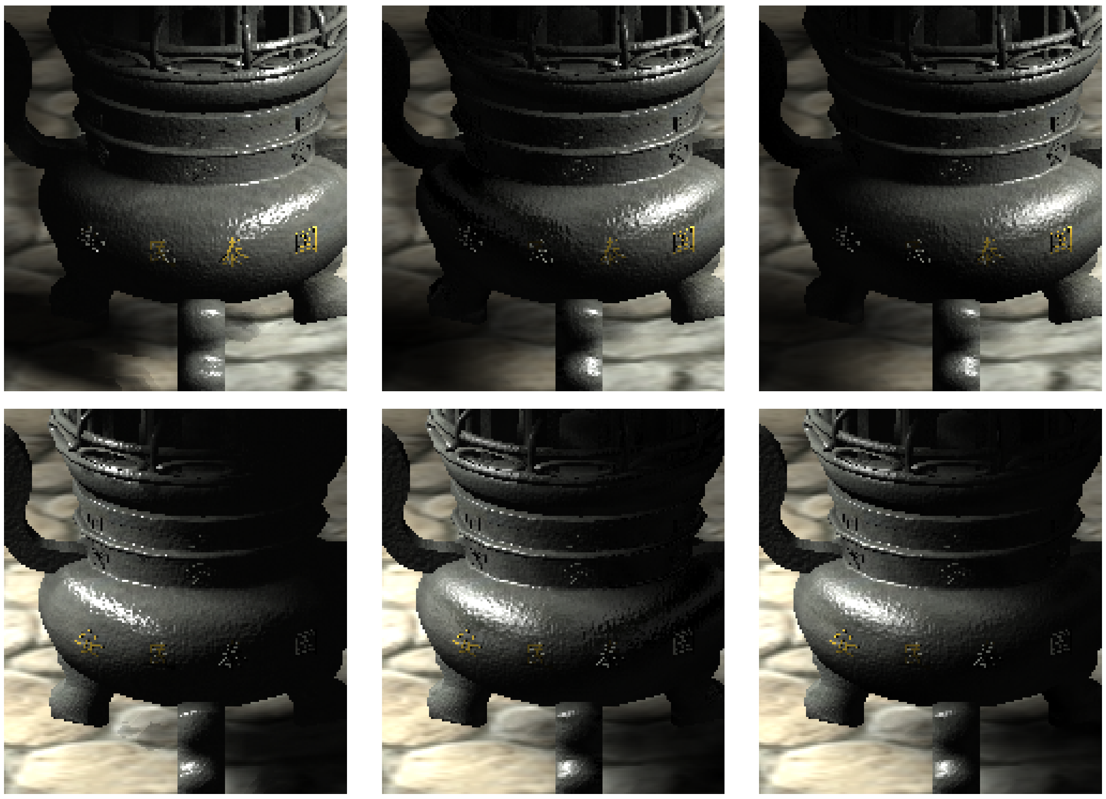

(a) from reference images (b) SH approach (c) SRBF approach

Fig. 23. Visual comparison of relit images with specular highlight.

We measure the rendering speed in terms of average frame rate (frames per second, fps) against the number of relighting maps used. Note that the more relighting maps are used, the more realistic the visual effect is. Figure 24 summarizes the statistics. All timing statistics are taken on a PC (Pentium IV 2.4GHz CPU) installed with a consumer-level GPU (nVidia Geforce Fx 5950 Ultra with 256MB memory). The resolution of tested image-based scene is 1024×256. We can see that the rendering speed of the SRBF approach is faster than that of the SH approach. When the number of relighting maps is 25, the SRBF approach achieves 15 fps while the SH approach only achieves 6.3 fps. The statistics evidence the fast evaluation of SRBF basis functions.

9 Discussions and Conclusions

This paper described a SRBF-based method for compressing IBR data sets. Firstly, all reference images are represented by a few SRBF relighting maps. Then, a wavelet-based coding is applied to pursue the spatial correlation within each SRBF relighting map. In terms of PSNR, our approach is much better than standard image and video coding methods, including JPEG, JPEG2000 and MPEG. It is also comparable

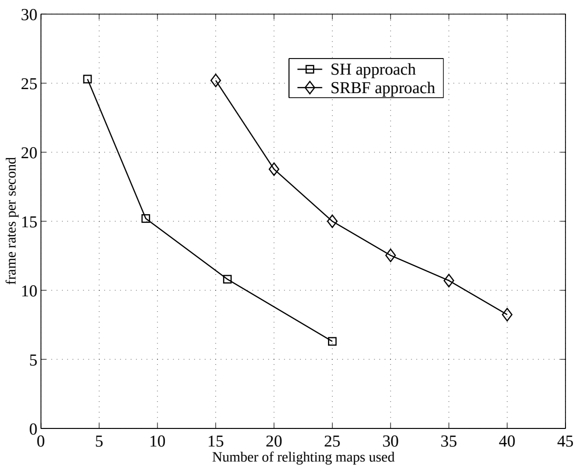


Fig. 24. Rendering speed for the point light source.

to the previous SH approach. The major advantage of our approach over the SH approach is the speed performance, due to the simplicity and efficiency in evaluating the basis functions.

We discussed the weight estimation process which is based on a constrained least square method. The constrain is crucial to reduce the noise sensitivity of rendering to the quantization noise introduced by further compression (the second-level compression in our case). We also discussed the relighting process and ways to render on modern consumer-level graphics accelerators. During rendering, the compressed relighting maps are decompressed from the secondary storage and loaded into texture buffer of GPU. With these relighting maps, users can configure arbitrary illumination condition interactively.

Although we mainly discussed how to relight image-based scene, the proposed SRBF approach can be extended to 3D object rendering with the viewpoint being changeable. The proposed SRBF method can be used for encoding viewpoint-independent reflectance property on any 3D surface. Just like PRT [40], it can represent reflection and self-shadowing as demonstrated in Figure 25. Each vertex of the torus holds a spherical function which represents the reflectance and local visibility. Hence the spherical function can also be encoded by our SRBF method. The example in Figure 25 demonstrates that the proposed method can be extended to allow the change of viewpoint as well as the change of lighting.

For glossy surface (viewpoint-dependent reflection), every vertex holds a 4D bidirectional function which input domain is the cross product of two unit spheres. Similar to the SH-based method, the 4D function can be either doubly SRBF-projected in both lighting and viewing dimensions or SRBF-projected in the lighting dimensions but sampled in the viewing dimensions [39][40].

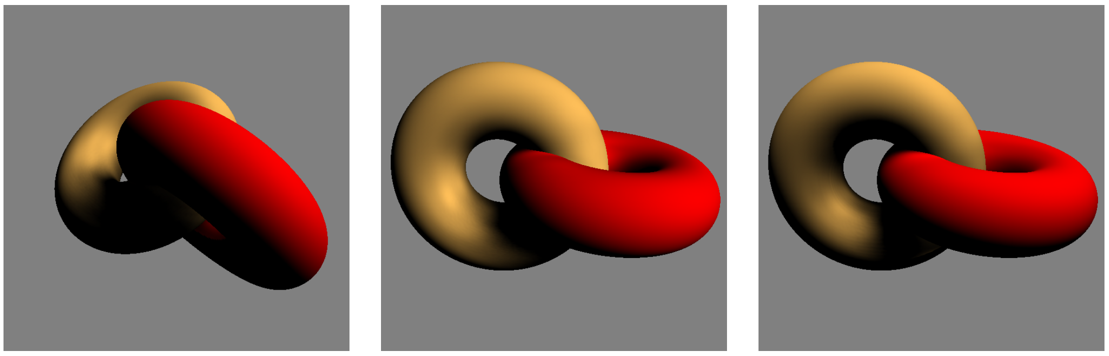


Fig. 25. Extension to encode viewpoint-independent reflectance property on 3D object. The surface is Lambertian and self-shadowing is considered.

ACKNOWLEDGMENT

The work was supported by a research grant from City University of Hong Kong (Project No. 7001549) and the Hong Kong Special Administrative Region RGC Earmarked Grant (Project No. CUHK 4189/03E).

REFERENCES

[1] P. Debevec, “Image-based lighting,” *IEEE Computer Graphics and Applications*, pp. 26–34, 2003.

[2] T. T. Wong, P. A. Heng, S. H. Or, and W. Y. Ng, “Image-based rendering with controllable illumination,” in *Proceedings of Eighth Eurographics Workshop on Rendering*, 1997, pp. 13–22.

[3] Y. Yu and J. Malik, “Recovering photometric properties of architectural scenes from photographs,” in *SIGGRAPH ’98 Conference Proceedings*, July 1998.

[4] T. T. Wong, C. W. Fu, and P. A. Heng, “Interactive relighting of panoramas,” *IEEE Computer Graphics and Applications*, pp. 32–41, 2001.

[5] M. Levoy and P. Hanrahan, “Light field rendering,” in *SIGGRAPH ’96 Conference Proceedings*, August 1996.

[6] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, “The lumigraph,” in *SIGGRAPH ’96 Conference Proceedings*, August 1996, pp. 43–54.

[7] H. Y. Shum and L. W. He, “Rendering with cconcentric mosaics,” in *SIGGRAPH ’99 Conference Proceedings*, August 1999, pp. 299–306.

[8] S. C. Chen, “QuickTime VR - an image-based approach to virtual environment navigation,” in *SIGGRAPH ’95 Conference Proceedings*, August 1995, pp. 29–38.

[9] T. T. Wong, C. W. Fu, P. A. Heng, and C. S. Leung, “The plenoptic illumination function,” *IEEE Trans. Multimedia*, vol. 3, pp. 361–371, 2003.

[10] T. T. Wong and C. S. Leung, “Compression of illumination-adjustable images,” *IEEE Trans. CSVT*, vol. 13, pp. 1107–1118, 2003.

[11] G. K. Wallace, “The JPEG still picture compression standard,” *IEEE Trans. Consumer Electronics*, pp. 18–34, 1992.

[12] M. D. Adams and R. Ward, “Wavelet transforms in the jpeg-2000 standard,” in *Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, 2001, vol. 1, pp. 160–163.

[13] G. L. Chen, J. S. Pan, and J. L. Wang, “Video encoder architecture for MPEG2 real time encoding,” *IEEE Trans. Consumer Electronics*, pp. 290–299, 1996.

[14] R. Ramamoorthi and P. Hanrahan, “Frequency space environment map rendering,” *ACM Trans. Graphics*, vol. 21, pp. 517–526, 2002.

[15] P. P. Sloan, X. Liu, H. Y. Shum, and J. Snyder, “Bi-scale radiance transfer,” *ACM Trans. Graphics*, vol. 22, pp. 370–375, 2003.

[16] J. Kautz, “Hardware lighting and shading: a survey,” *Computer Graphics Forum*, vol. 23, pp. 85–112, 2004.

[17] T. T. Wong, S. H. Or, and C. W. Fu, *Real-Time Relighting of Compressed Panoramas*, pp. 375–388, Charles River Media, 2003.

[18] D. S. Broomhead and D. Lowe, “Multivariate functional interpolation and adaptive networks,” *Complex Systems*, pp. 321–355, 1988.

[19] R. L. Jenison and K. Fissell, “A spherical basis function neural network for modeling auditory space,” *Neural Computation*, vol. 8, pp. 115–128, 1996.

[20] G. Wahba and J. Wendelbergar, “Some new mathematical methods for variational objective analysis using splines and cross-validation,” *Monthly Weather Rev*, pp. 1122–1145, 1980.

[21] T. Poggio and F. Girosi, “Networks for approximation and learning,” *Proceedings of IEEE*, pp. 1481–1497, 1990.

[22] T. Poggio and S. Edelman, “A network that learns to recognize three-dimensional objects,” *Nature*, 1990.

[23] G. Golub and C. Van Loan, *Matrix Computations*, Johns Hopkins Univ. Press, 1990.

[24] M. Magnor and B. Girod, “Data compression for light-field rendering,” *IEEE Trans. CSVT*, vol. 10, pp. 338–342, April 2000.

[25] C. Bajaj, I. Ihm, and S. Park, “3D RGB image compression for interactive applications,” *ACM Trans. Graphics*, vol. 20, pp. 10–38, 2001.

[26] C. Zhang and J. Li, “Compression of lumigraph with multiple reference frame (MRF) prediction and just-in-time rendering,” in *Proc. IEEE Conference on Data Compression*, 2003, pp. 253–262.

[27] Ingmar Peter and Wolfgang Straber, “The wavelet stream - progressive transmission of compressed light field data,” in *Proc. IEEE Visualization*, 1999.

[28] W.H. Leung and T. Chen, “Compression with mosaic prediction for image-based rendering applications,” in *Proc. International Conference on Multimedia and Expo*, 2000, pp. 1649–1653.

[29] P. L. Dragotti, G. Poggi, and A. R. P. Ragozini, “Compression of multispectral images by three-dimensional SPIHT algorithm,” *IEEE Trans. Geoscience and Remote Sensing*, vol. 38, pp. 416–428, 2000.

[30] B. J. Kim, Z. X. Xiong, and W. A. Pearlman, “Low bit-rate scalable video coding with 3-D set partitioning in hierarchical trees (3-D SPIHT),” *IEEE Trans. CSVT*, vol. 10, pp. 1374–1387, 2000.

[31] R. Epstein, P. Hallinan, and A.L. Yuille, “5+/-2 eigenimages suffice: An empirical investigation of low-dimensional lighting models,” in *Proc. IEEE Workshop on Physics-Based Modeling in Computer Vision*, 1995, pp. 108–116.

[32] P. N. Belhumeur and D. J. Kriegman, “What is the set of images of an object under all possible lighting conditions?,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1996, pp. 270–277.

[33] Z. Zhang, “Modeling geometric structure and illumination variation of a scene from real images,” in *Proc. International Conference on Computer Vision*, 1998.

[34] S. K. Nayar and H. Murase, “Dimensionlity of illumination in appearance matching,” in *Proc. IEEE International Conference on Robotics and Automation*, 1996, pp. 1326–1332.

[35] G. Hager and P. Belhumeur, “Real-time tracking of image regions with changes in geometry and illumination,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, June 1996.

[36] K. Nishino, Y. Sato, and Katsushi Ikeuchi, “Real-time tracking of image regions with changes in geometry and illumination,” in *Proc. IEEE Computer Vision and Pattern Recognition Conference*, June 1999, pp. 618–624.

[37] N. Max B. Cabral and R. Springmeyer, “Bidirectional reflection functions from surface bump maps,” in *SIGGRAPH ’87 Conference Proceedings*, June 1987, pp. 273–281.

[38] F. X. Sillion, J. R. Arvo, S. H. Westin, and D. P. Greenberg, “A global illumination solution for general reflectance distributions,” in *SIGGRAPH ’91 Conference Proceedings*, 1991, pp. 187–196.

[39] J. Kautz, P. Sloan, and J. Snyder, “Fast, arbitrary BRDF shading for low-frequency lighting using spherical harmonics,” in *Proc. of the Thirteenth Eurographics Workshop on Rendering*, 2002, pp. 291–297.

[40] P. Sloan, J. Hall, J. Hart, and J. Snyder, “Clustered principal components for precomputed radiance transfer,” *ACM Trans. on Graphics*, vol. 22, pp. 381–391, 2003.

[41] E. H. Adelson and J. R. Bergen, “The plenoptic function and the elements of early vision,” *Computational Models of Visual Processing*, pp. 3–20, 1991.

[42] T. T. Wong, W. S. Luk, and P. A. Heng, “Sampling with hammersley and halton points,” *Journal of Graphic Tools*, vol. 2, pp. 9–24, 1997.

[43] Zhouchen Lin, Tien-Tsin Wong, and Heung-Yeung Shum, “Relighting with the reflected irradiance field: Representation, sampling and reconstruction,” *International Journal of Computer Vision*, vol. 49, no. 2-3, pp. 229–246, September-October 2002.

[44] G. Ward, *Real Pixels*, pp. 80–83, Addison Wesley, 1991.

[45] Paul Debevec and Tim Hawkins and Chris Tchou and Haarm-Pieter Duiker and Westley Sarokin and Mark Sagar, “Acquiring the reflectance field of a human face,” in *SIGGRAPH ’2000 Conference Proceedings*, 2000, pp. 145–156.

[46] S. Chen, B. Mulgrew, and P. M. Grant, “A clustering technique for digital communications channel equalization using radial basis function networks,” *IEEE Trans. Neural Networks*, vol. 4, pp. 570–579, 1993.

[47] A. Roy, S. Govil, and R. Miranda, “A neural-network learning theory and a polynomial time rbf algorithm,” *IEEE Trans. Neural Networks*, vol. 8, pp. 1301–1313, 1997.

[48] V. Kardirkamanathan, M. Niranjan, and F. Fallside, “Sequential adaptation of radial basis function neural networks and its application to time-series prediction,” in *Advances in Neural Information Processing System*, 1991, pp. 721–727.

[49] A. C. Tsoi and S. Tan, “Recurrent neural networks: A constructive algorithm and its properties,” *Neurocomputing*, vol. 15, pp. 309–326, 1997.

[50] A. C. Tsoi, “Multilayer perceptron trained using radial basis functions,” *Electronics Letters*, vol. 25, pp. 1296–1297, 1989.

[51] K. Kiernan, J. D. Mason, and K. Warwick, “Robust initialisation of gaussian radial basis function networks using partitioned k-mean clustering,” *Electronics Letters*, vol. 32, pp. 671–673, 1996.

[52] S. Chen, C. F. N. Cowan, and P. M. Grant, “Orthogonal least squares learning for radial basis function networks,” *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, 1991.

[53] J. Platt, “A resource-allocating network for function interpolation,” *Neural Computation*, vol. 3, pp. 213–225, 1991.

[54] Jian Jun Cui and Willi Freeden, "Equidistribution on the sphere," *SIAM Journal on Scientific Computing*, vol. 18, no. 2, pp. 595–609, Mar. 1997.

[55] S. Haykin, *Adaptive Filter Theory*, Englewood Cliffs, NJ: Prentice Hall, 1991.

[56] C. S. Leung, A. C. Tsoi, and L. W. Chan, "Two regularizers for recursive least squared algorithms in feedforward multilayered neural networks," *IEEE Trans. Neural Networks*, vol. 12, pp. 1314–1332, 2001.

[57] N. S. Jayant and P. Noll, *Digital Coding of Waveforms*, Englewood CliRs, NJ:Prentice-Hall, 1984.

[58] D. Mackay, "Bayesian interpolation," *Neural Computation*, vol. 4, pp. 415–447, 1992.

[59] J. L. Orr Mark, "Regularization in the selection of radial basis function centres," *Neural Computation*, vol. 7, pp. 606–623, 1995.

[60] P. C. Cosman, R. M. Gray, and M. Vetterli, "Vector quantization of image subbands: a survey," *IEEE Trans. Signal Processing*, vol. 5, pp. 202–225, 1996.

[61] C. S. Leung and L. W. Chan, "Transmission of vector quantization over a noisy channel," *IEEE Trans. Neural Networks*, vol. 8, pp. 582–589, 1997.

[62] B. Usevitch, "A tutorial on modern lossy wavelet image compression: Foundations of jpeg 2000," *IEEE Signal Processing Magazine*, pp. 22–34, 2001.

[63] R. W. Buccigrossi and E. P. Simoncelli, "Progressive wavelet image coding based on a conditional probability model," in *Proc. International Conference Acoustics Speech and Signal Processing*, 1997, vol. 1, pp. 21–24.

[64] R. Hall, *Illumination and Color in Computer Generated Imagery*, Springer-Verlag, 1989.

[65] R. Fernando and M. J. Kilgard, *The Cg Tutorial: The Definitive Guide to Programmable Real-Time Graphics*, Addison-Wesley Pub Co., 2003.