1 Algorithms

Algorithms 1 and 2 describe the training and compression tasks, respectively. Algorithm 1 outputs an ensemble of multi-dimensional dictionaries, while Algorithm 2 depicts the process of computing the sparse coefficients together with the membership index for one data point.

**Algorithm 1**

**Training an ensemble of 4-dimensional dictionaries.**

**Require:** The training set \( \{ \mathbf{X}^{(i)} \}_{i=1}^{N_l} \), sparsity \( \tau_l \), error threshold \( \epsilon \), and the number of dictionaries \( K \)

**Ensure:** A 4-dimensional dictionary ensemble \( \{ \mathbf{U}^{(1,k)}, \mathbf{U}^{(2,k)}, \mathbf{U}^{(3,k)}, \mathbf{U}^{(4,k)} \}_{k=1}^{K} \)

\begin{enumerate}
    \item Set \( \mathbf{U}^{(1,k)}, \mathbf{U}^{(2,k)}, \mathbf{U}^{(3,k)}, \mathbf{U}^{(4,k)} \) to random orthonormal matrices and initialize \( M_{i,j} \leftarrow K^{-1}, \forall i, \forall j \)
    \item \( \beta = 0.01 \) \quad \triangleright \text{Initialization of the inverse temperature}
    \item repeat \quad \triangleright \text{Increase the inverse temperature}
        \begin{enumerate}
            \item \( \beta = \beta \times 2 \)
            \item for \( k = 1, \ldots, K \) do
                \begin{enumerate}
                    \item \( Z^{(j,k)} = 0, \forall k, \forall j \)
                    \item for \( i = 1, \ldots, N_l \) do
                        \begin{enumerate}
                            \item \( S^{(i,k)} = \mathbf{X}^{(i)} \times_1 \mathbf{U}^{(1,k)} \times_2 \mathbf{U}^{(2,k)} \times_3 \mathbf{U}^{(3,k)} \times_4 \mathbf{U}^{(4,k)} \)
                            \item Nullify \( (\prod_{j=1}^{4} m_{j}) - \tau_l \) smallest elements in absolute value from \( S^{(i,k)} \)
                            \item \( e_i^j = \| \mathbf{X}^{(i)} - S^{(i,k)} \times_1 \mathbf{U}^{(1,k)} \times_2 \mathbf{U}^{(2,k)} \times_3 \mathbf{U}^{(3,k)} \times_4 \mathbf{U}^{(4,k)} \|_F^2 \) \quad \triangleright \text{Compute the error of representation}
                        \end{enumerate}
                    \end{enumerate}
                \end{enumerate}
            \item for \( k = 1, \ldots, K \) do
                \begin{enumerate}
                    \item \( M_{i,k} = \left( \sum_{b=0}^{K} e^b (e_i^b - e_i^k) \right)^{-1} \)
                    \item \( Z^{(j,k)} = \sum_{i=1}^{N_l} M_{i,k} \mathbf{X}^{(i)} \times_1 \mathbf{U}^{(j,k)} \times_2 \ldots \times_4 \mathbf{U}^{(j+1,k)} \times_4 \mathbf{U}^{(j-1,k)} \times_4 \ldots \times_4 \mathbf{U}^{(4,k)} \) \quad \triangleright \mathbf{X}^{(i)} \text{ is the unfolding of } \mathbf{X}^{(i)} \text{ along the } j\text{th mode}
                \end{enumerate}
            \item end for
        \end{enumerate}
    \item end for
    \item until Convergence of \( \{ \mathbf{U}^{(1,k)}, \mathbf{U}^{(2,k)}, \mathbf{U}^{(3,k)}, \mathbf{U}^{(4,k)} \}_{k=1}^{K} \)
    \item until \( \| M - \lfloor M \rfloor \|_F < \epsilon \) \quad \triangleright \text{i.e. until } M \text{ is binary or near binary}
\end{enumerate}
the interpolation between the four nearest neighbors as follows:

As discussed in the paper, we perform bilinear interpolation to find the reflectance value at new

2 Angular interpolation

As discussed in the paper, we perform bilinear interpolation to find the reflectance value at new

3 Model parameters and storage calculations

To enable model comparison with Rainer et al. [RGJW20], we adjusted the storage cost of our method
to correspond to theirs, where 16, 32, 64, 128, and 256 latent variables were used. In the following,
we explain how the storage requirements are set for our method to fulfill this comparison. Once we
project the data points of each material onto our learned dictionary, we obtain a sparse representa-

Using the three nearest neighbors, we can utilize the fast local access to each texel in the sparse coefficients space and apply

1: e ∈ R^K ← ∞ and z ∈ R^K ← 1
2: for k = 1, . . . , K do
3:    X^{(k)} ← Y^{(i)} × (U^{(1,k)})^T × (U^{(2,k)})^T × (U^{(3,k)})^T × (U^{(4,k)})^T
4:    while z_k ≤ τ_t and e_k > ϵ do
5:        Nullify (Π_j=1 mj) − z_k smallest elements in absolute value from X^{(k)}
6:        e_k ← ∥Y^{(i)} − X^{(k)} × U^{(1,k)} × U^{(2,k)} × U^{(3,k)} × U^{(4,k)}∥_F^2
7:        z_k = z_k + 1
8:    end while
9: end for
10: a ← index of min(z)
11: if z_a = τ_t then
12:    a ← index of min(e)
13: end if
14: Y^{(i)} ← X^{(a)}
[RG JW20] also uses \((n \times 400 \times 400 \times 2 + 38269 \times 2) = 320000 \times n + 76538\) coefficients because of the storage of \(n\) latent vectors together with the network weights as 16-bit float EXR images. For the fair comparison, we do not include the storage cost of encoder as it is not used during the rendering.

We set testing sparsities of \(\tau_{ty} = 308, \tau_{tu} = 32,\) and \(\tau_{tv} = 32\) for SparseBTF (16), \(\tau_{ty} = 885, \tau_{tu} = 64,\) and \(\tau_{tv} = 64\) for SparseBTF (32), \(\tau_{ty} = 2037, \tau_{tu} = 128,\) and \(\tau_{tv} = 128\) for SparseBTF (64), \(\tau_{ty} = 4617, \tau_{tu} = 256,\) and \(\tau_{tv} = 256\) for SparseBTF (128), and \(\tau_{ty} = 9226, \tau_{tu} = 512,\) and \(\tau_{tv} = 512\) for SparseBTF (256) for Y, U, and V channels, respectively.

4 Results

4.1 Angular interpolation

In the paper, we demonstrate that interpolation can be performed in the coefficient space, independently of the interpolation algorithm used. We test a variety of interpolation techniques, as illustrated in Figure 1, including basic approaches such as nearest neighbor and bilinear, as well as kernel density estimation [Sil86] using various kernel functions, including triangular, Epanechnikov, biweight, cubic, and Gaussian. The images are rendered using the NVIDIA OptiX framework [PBD+10] under the same light and view directions. Our experiments reveal that all interpolation strategies can reconstruct plausible appearances for diffuse materials like Carpet11. For shiny materials, however, nearest neighbor results in highlight aliasing due to the limited resolution of the angular bins. The Gaussian kernel function [Pav90] with 3-by-3 nearest neighbors was found to most effectively eliminate the highlight aliasing, even in an interactive session where the user could look for challenging grazing angle configurations. The Gaussian kernel also seemed a good trade-off between avoiding highlight aliasing while also limiting the amount of blur in the Lambertian components. To generate the ground truth, we used the \(k\)-Nearest Neighbor algorithm for interpolation, with the weights of the points adjusted by a power parameter to enable smooth interpolation at the boundaries.

![Figure 1: Comparison of renderings using different interpolation algorithms.](image-url)
4.2 Reconstruction

We used following 18 BTFs for the training set: Carpet01, Carpet02, Fabric01, Fabric02, Felt01, Felt02, Leather02, Leather03, Leather08, Leather10, Stone04, Stone06, Stone11, Wallpaper01, Wallpaper02, Wallpaper11, Wood01, and Wood02. To further compare the representation error using SparseBTF and [RGJW20], we compute the mean square reconstruction error on the cropped dataset with a resolution of $100 \times 100$. Figure 2 shows the log($MSE$) of both methods with different number of coefficients, ranging from 32 to 128 latent maps for each spatial dimension. The reconstruction error for SparseBTF decreases consistently by reducing the number of coefficients, while the improvement achieved by [RGJW20] is marginal. To further analyze the reconstruction quality, in Figure 3, we evaluate SparseBTF (32) and [RGJW20] (32) in terms of PU2-PSNR [AMS08], an image quality metric that computes Peak Signal-to-Noise Ratio (PSNR) in the perceptually uniform space. After reconstruction, we measure PU2-PSNR over all angular images and report the maximum, minimum, median, 25th, and 75th percentile values. Our method acquires a higher average PU2-PSNR in 6 materials showing the superior visual quality of angular images reconstructed by SparseBTF.

To further evaluate the performance of SparseBTF, we conduct additional experiments on the UBO2003 dataset [SSK03]. Table 1 compares SparseBTF and Sparse Tensor Decomposition [RK09] in terms of the ratio of their mean squared reconstruction errors to that of PCA for the Pulli dataset. We ensure that all methods have the same storage cost. The error ratio ($\sigma$) is defined as

$$\sigma = \frac{e}{e_{PCA}} \tag{2}$$

where $e_{PCA}$ represents the reconstruction MSE of PCA, while $e$ represents the reconstruction MSE of the method in question (SparseBTF and that of [RK09]). A smaller error ratio indicates better performance as it signifies the lower error achieved by the method being evaluated compared to PCA. The results suggest that SparseBTF provides superior reconstruction quality, making it a favorable choice for compression tasks in BTF representations.

![Figure 2: Logarithm of Mean Square Error comparison of reconstructed BTFs. SparseBTF (32), SparseBTF (64), and SparseBTF (128) imply that our compression ratio is equal to [RGJW20] with 32, 64, and 128 latent coefficients, respectively.](image)

Figures 4–7 visualize the reconstructed angular images with two different combinations of light and view directions. The error images are normalized by dividing by the maximum value and then, multiplied by 2 for the ease of visualization. In addition, we apply gamma correction to the
Figure 3: PU2-PSNR comparison of reconstructed images using our method and [RGJW20] with similar storage complexity. Each box contains 25 percentile (bottom of the box), 75 percentile (top of the box), maximum value (top of the dashed line), minimum value (bottom of the dashed line) and median (line inside the box).

<table>
<thead>
<tr>
<th>Reconstruction error ratio</th>
<th>SparseBTF</th>
<th>[RK09]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.27</td>
<td>0.65</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Comparison of reconstruction error ratio of SparseBTF and the method of [RK09] for the Pulli dataset.

Our reconstruction quality surpasses that of Rainer et al. [RGJW20] in most cases. We investigate the impact of sparsity, i.e. the number of non-zero coefficients, on reconstructed images produced by SparseBTF in Figure 8.
Figure 4: Comparison of reconstructed BTFs using our method and [RGJW20]. Incident azimuth and elevation angles: 0°, 90°. Observation azimuth and elevation angles: 0°, 45°.
Figure 5: Comparison of reconstructed BTFs using our method and [RGJW20]. Incident azimuth and elevation angles: 270°, 66.5°. Observation azimuth and elevation angles: 15°, 79°.
Figure 6: Comparison of reconstructed BTFs using our method and [RGJW20]. Incident azimuth and elevation angles: 127.5°, 22.5°. Observation azimuth and elevation angles: 120°, 52.5°.
Figure 7: Comparison of reconstructed BTFs using our method and [RGJW20]. Incident azimuth and elevation angles: 195°, 60°. Observation azimuth and elevation angles: 105°, 30°.
Figure 8: Comparison of reconstructed BTFs using our method with different testing sparsities, i.e., the number of non-zero coefficients. Incident azimuth and elevation angles: 270°, 66.5°. Observation azimuth and elevation angles: 15°, 79°.
References


