SparseBTF: Sparse Representation Learning for Bidirectional Texture Functions

Supplementary Material

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 $\{\boldsymbol{\mathcal{X}}^{(i)}\}_{i=1}^{N_l}$, sparsity $\tau_l$, error threshold $\epsilon$, and the number of dictionaries $K$
Ensure: A 4-dimensional dictionary ensemble $\left\{\mathbf{U}^{(1,k)}, \mathbf{U}^{(2,k)}, \mathbf{U}^{(3,k)}, \mathbf{U}^{(4,k)}\right\}_{k=1}^{K}$
 1: Set $\left\{\mathbf{U}^{(1,k)}, \mathbf{U}^{(2,k)}, \mathbf{U}^{(3,k)}, \mathbf{U}^{(4,k)}\right\}_{k=1}^{K}$ to random orthonormal matrices and initialize $\mathbf{M}_{i,j} \leftarrow K^{-1}, \forall i, \forall j$
 2: $\beta = 0.01$                                  ▷ Initialization of the inverse temperature
 3: repeat
 4:     $\beta = \beta \times 2$                    ▷ Increase the inverse temperature
 5:     repeat
 6:         for $i = 1, \dots, N_l$ do
 7:             $\mathbf{Z}^{(j,k)} = 0, \forall k, \forall j$
 8:             for $k = 1, \dots, K$ do
 9:                 $\boldsymbol{\mathcal{S}}^{(i,k)} = \boldsymbol{\mathcal{X}}^{(i)} \times_1 \left(\mathbf{U}^{(1,k)}\right)^T \times_2 \left(\mathbf{U}^{(2,k)}\right)^T \times_3 \left(\mathbf{U}^{(3,k)}\right)^T \times_4 \left(\mathbf{U}^{(4,k)}\right)^T$
10:                 Nullify $(\Pi_{j=1}^{4} m_j) - \tau_l$ smallest elements in absolute value from $\boldsymbol{\mathcal{S}}^{(i,k)}$
11:                 $\mathbf{e}_k^i = \| \boldsymbol{\mathcal{X}}^{(i)} - \boldsymbol{\mathcal{S}}^{(i,k)} \times_1 \left(\mathbf{U}^{(1,k)}\right)^T \times_2 \left(\mathbf{U}^{(2,k)}\right)^T \times_3 \left(\mathbf{U}^{(3,k)}\right)^T \times_4 \left(\mathbf{U}^{(4,k)}\right)^T \|_F^2$
12:                                             ▷ Compute the error of representation
13:             end for
14:             for $k = 1, \dots, K$ do
15:                 $\mathbf{M}_{i,k} = \left(\Sigma_{b=1}^{K} e^{\beta(\mathbf{e}_k^i - \mathbf{e}_b^i)}\right)^{-1}$
16:                 $\mathbf{Z}^{(j,k)} = \Sigma_{i=1}^{N_l} \mathbf{M}_{i,k} \boldsymbol{\mathcal{X}}_{[j]}^{(i)} \left(\mathbf{U}^{(4,k)} \otimes \cdots \otimes \mathbf{U}^{(j+1,k)} \otimes \mathbf{U}^{(j-1,k)} \otimes \cdots \otimes \mathbf{U}^{(1,k)}\right) \left(\boldsymbol{\mathcal{S}}_{[j]}^{(i,k)}\right)^T, \forall j \in \{1, 2, 3, 4\}$    ▷ $\boldsymbol{\mathcal{X}}_{[j]}^{(i)}$ is the unfolding of $\boldsymbol{\mathcal{X}}^{(i)}$ along the $j$th mode
17:             end for
18:         end for
19:         for $k = 1, \dots, K$ do
20:             $\mathbf{U}^{(j,k)} = \mathbf{Z}^{(j,k)} \left(\left(\mathbf{Z}^{(j,k)}\right)^T \mathbf{Z}^{(j,k)}\right)^{\frac{-1}{2}}, \forall j \in \{1, 2, 3, 4\}$
21:         end for
22:     until Convergence of $\left\{\mathbf{U}^{(1,k)}, \mathbf{U}^{(2,k)}, \mathbf{U}^{(3,k)}, \mathbf{U}^{(4,k)}\right\}_{k=1}^{K}$
23: until $\|\mathbf{M} - \lfloor \mathbf{M} \rceil\|_F^2 < \epsilon$            ▷ i.e. until $\mathbf{M}$ is binary or near binary
```

Algorithm 2 Computing non-zero coefficients and the membership index for a data point.

```
Require: A data point $\boldsymbol{\mathcal{Y}}^{(i)}$ in the testing set, sparsity $\tau_t$, error threshold $\epsilon$, and the dictionary
    ensemble
Ensure: The coefficient tensor $\boldsymbol{\mathcal{S}}$, the dictionary membership index $a$
 1: $\mathbf{e} \in \mathbb{R}^K \leftarrow \infty$ and $\mathbf{z} \in \mathbb{R}^K \leftarrow 1$
 2: for $k = 1, \ldots, K$ do
 3:     $\boldsymbol{\mathcal{X}}^{(k)} \leftarrow \boldsymbol{\mathcal{Y}}^{(i)} \times_1 \left(\mathbf{U}^{(1,k)}\right)^T \times_2 \left(\mathbf{U}^{(2,k)}\right)^T \times_3 \left(\mathbf{U}^{(3,k)}\right)^T \times_4 \left(\mathbf{U}^{(4,k)}\right)^T$
 4:     while $\mathbf{z}_k \leq \tau_t$ and $\mathbf{e}_k > \epsilon$ do
 5:         Nullify $(\Pi_{j=1}^4 m_j) - \mathbf{z}_k$ smallest elements in absolute value from $\boldsymbol{\mathcal{X}}^{(k)}$
 6:         $\mathbf{e}_k \leftarrow \|\boldsymbol{\mathcal{Y}}^{(i)} - \boldsymbol{\mathcal{X}}^{(k)} \times_1 \mathbf{U}^{(1,k)} \times_2 \mathbf{U}^{(2,k)} \times_3 \mathbf{U}^{(3,k)} \times_4 \mathbf{U}^{(4,k)}\|_F^2$
 7:         $\mathbf{z}_k = \mathbf{z}_k + 1$
 8:     end while
 9: end for
10: $a \leftarrow$ index of $\min(\mathbf{z})$
11: if $z_a = \tau_t$ then
12:     $a \leftarrow$ index of $\min(\mathbf{e})$
13: end if
14: $\boldsymbol{\mathcal{Y}}^{(i)} \leftarrow \boldsymbol{\mathcal{X}}^{(a)}$
```

2 Angular interpolation

As discussed in the paper, we perform bilinear interpolation to find the reflectance value at new angles. We can utilize the fast local access to each texel in the sparse coefficients space and apply the interpolation between the four nearest neighbors as follows:

$$\hat{\boldsymbol{\mathcal{Y}}}^{(i)}_{x_1,x_2,x_3,x_4} = \sum_{z=1}^{\tau_t} \hat{\boldsymbol{\mathcal{S}}}^{(i)}_{l_1^z,l_2^z,l_3^z,l_4^z} \mathbf{U}^{(1,\mathbf{m}_i)}_{x_1,l_1^z} \mathbf{U}^{(2,\mathbf{m}_i)}_{x_2,l_2^z} \Big(\beta((1-\alpha)\mathbf{U}^{(3,m_i)}_{x_{3_s},l_3^z}\mathbf{U}^{(4,m_i)}_{x_{4'_s},l_4^z} + \alpha\mathbf{U}^{(3,m_i)}_{x_{3_s},l_3^z}\mathbf{U}^{(4,m_i)}_{x_{4'_t},l_4^z}) + (1-\beta)((1-\alpha)\mathbf{U}^{(3,m_i)}_{x_{3_t},l_3^z}\mathbf{U}^{(4,m_i)}_{x_{4'_s},l_4^z} + \alpha\mathbf{U}^{(3,m_i)}_{x_{3_t},l_3^z}\mathbf{U}^{(4,m_i)}_{x_{4'_t},l_4^z})\Big), \quad (1)$$

where, s, t, and s', t', denote the index to the two nearest samples to the current incident light and view directions, respectively. α is the distance between current viewing and sampled outgoing angles, and β is the distance between current light direction and sampled incident light.

3 Model parameters and storage calculations

To enable 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 representation of that material, noted as sparse coefficients. We require 1 byte for storing the membership matrix $\mathbf{M}$, 2 bytes for the number of non-zero values, 3 bytes for the non-zero locations, and 2 bytes for the sparse coefficients per channel resulting in $(2\times\tau_{t_Y}\times1600+2\times\tau_{t_U}\times1600+2\times\tau_{t_V}\times1600)+(3\times\tau_{t_Y}\times1600+3\times\tau_{t_U}\times1600+3\times\tau_{t_V}\times1600)+(1600\times3+1600\times3+1600\times3) = 8000\times(\tau_{t_Y}+\tau_{t_U}+\tau_{t_V})+14400$ bytes for all channels. We also store our 4D dictionaries (8) as 16-bit floating point values for each color channel with the storage cost of $(10\times10\times8+10\times10\times8+151\times151\times8+151\times151\times8)\times3\times2 = 2198496$ elements making the total cost equal to $8000\times(\tau_{t_Y}+\tau_{t_U}+\tau_{t_V})+2212896$. Rainer et al.

[RGJW20] 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 $\tau_{t_Y} = 308$, $\tau_{t_U} = 32$, and $\tau_{t_V} = 32$ for SparseBTF (16), $\tau_{t_Y} = 885$, $\tau_{t_U} = 64$, and $\tau_{t_V} = 64$ for SparseBTF (32), $\tau_{t_Y} = 2037$, $\tau_{t_U} = 128$, and $\tau_{t_V} = 128$ for SparseBTF (64), $\tau_{t_Y} = 4617$, $\tau_{t_U} = 256$, and $\tau_{t_V} = 256$ for SparseBTF (128), and $\tau_{t_Y} = 9226$, $\tau_{t_U} = 512$, and $\tau_{t_V} = 512$ 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.

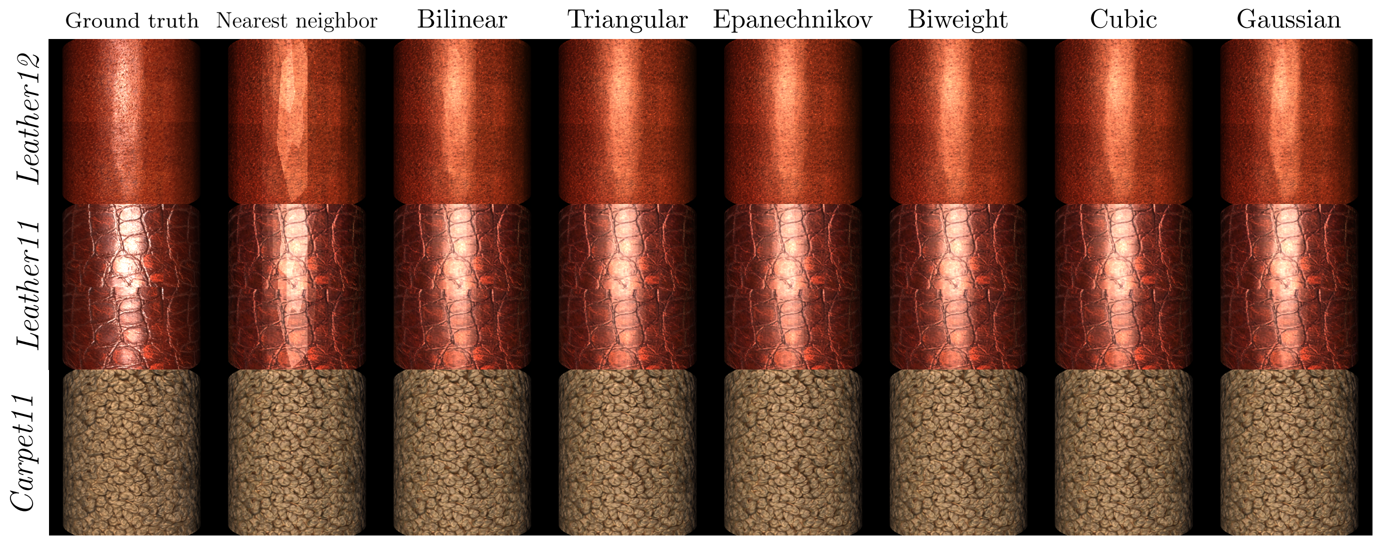

Figure 1: Comparison of renderings using different interpolation algorithms.

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 100×100. 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 (σ) 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.

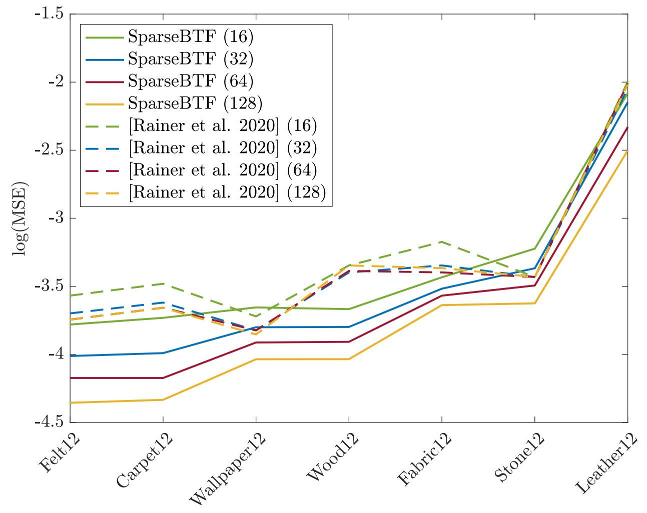

Figure 2: Logarithm of Mean Square Error comparison of reconstructed BTFs. SparseBTF (32), SparseBTF (64), and SparseBTF (128) imply that our compression ratio is equal to [RGJW20] with 32, 64, and 128 latent coefficients, respectively.

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

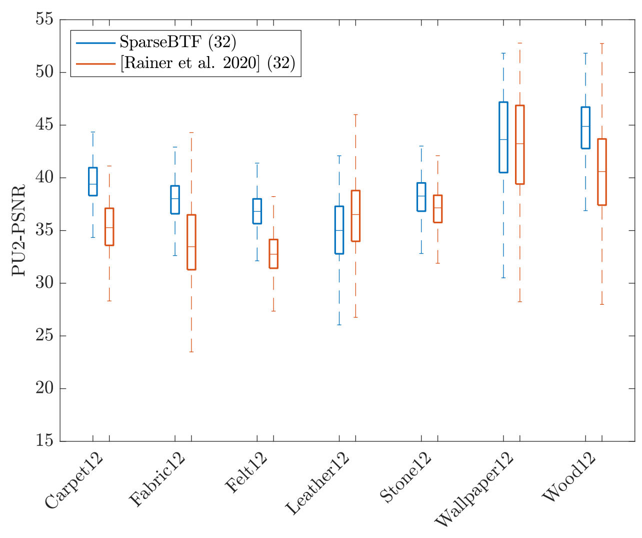

Figure 3: PU2-PSNR comparison of reconstructed images using our method and [RGJW20] with similar storage complexity. Each box contains 25 percentile (bottom of the box), 75 percentile (top of the box), maximum value (top of the dashed line), minimum value (bottom of the dashed line) and median (line inside the box).

	SparseBTF	[RK09]
Reconstruction error ratio	**0.27**	0.65

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

reconstructed images, $I_{out} = aI_{in}^{-\gamma}$, where $a = 1.3$ and $\gamma = 1.8$. 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.

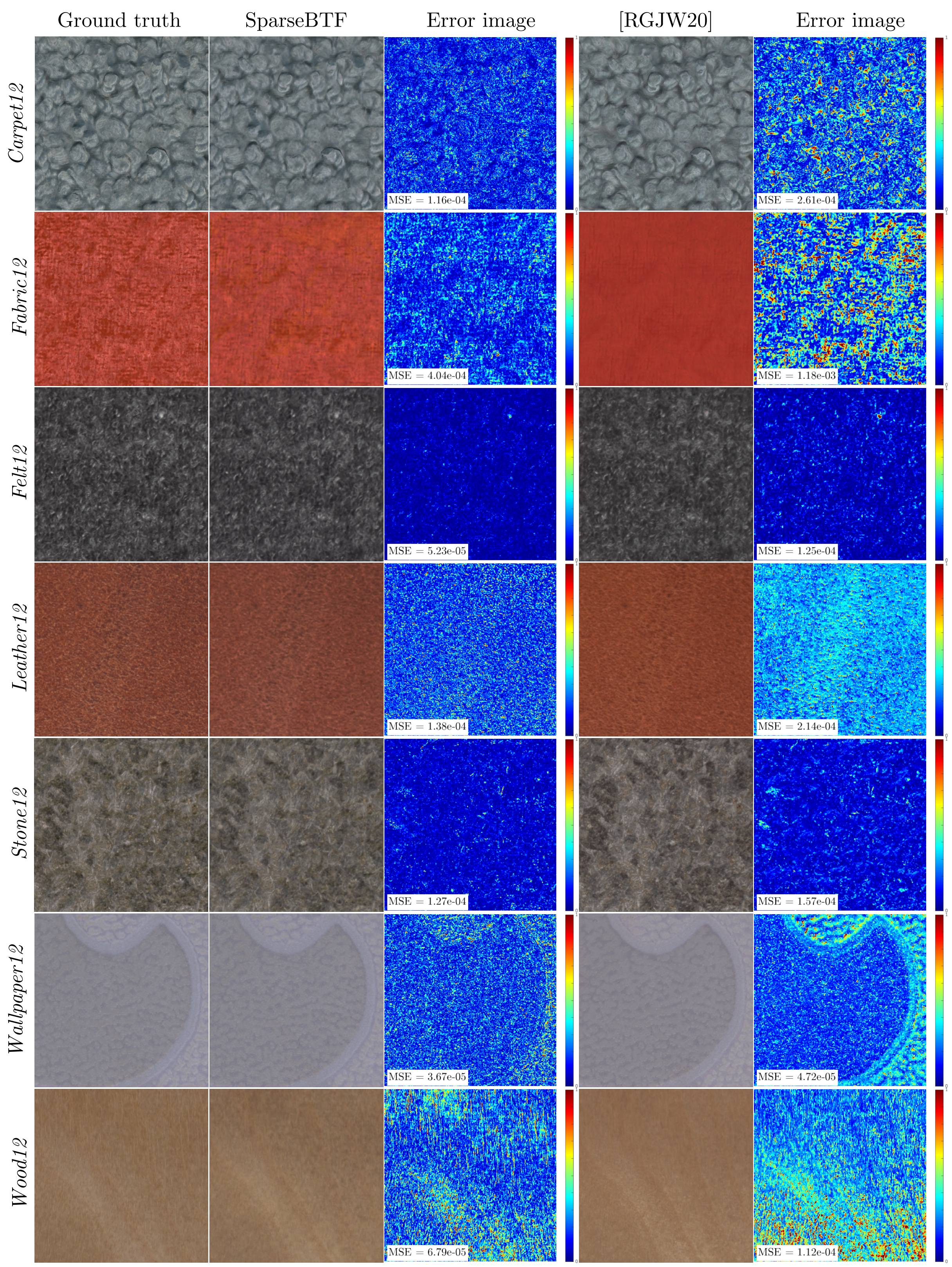

Figure 4: Comparison of reconstructed BTFs using our method and [RGJW20]. Incident azimuth and elevation angles: 0°, 90°. Observation azimuth and elevation angles: 0°, 45°.

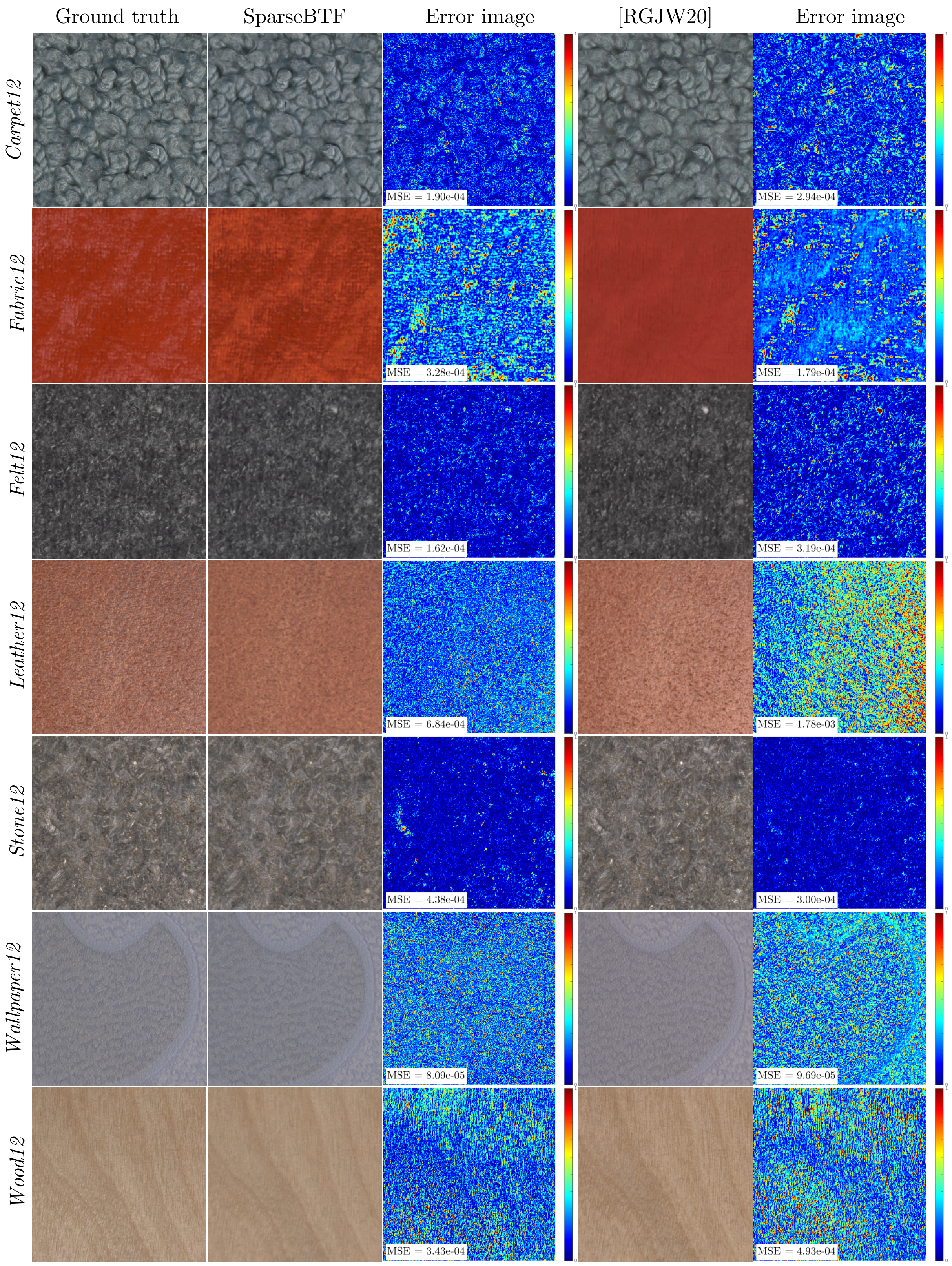

Figure 5: Comparison of reconstructed BTFs using our method and [RGJW20]. Incident azimuth and elevation angles: 270°, 66.5°. Observation azimuth and elevation angles: 15°, 79°.

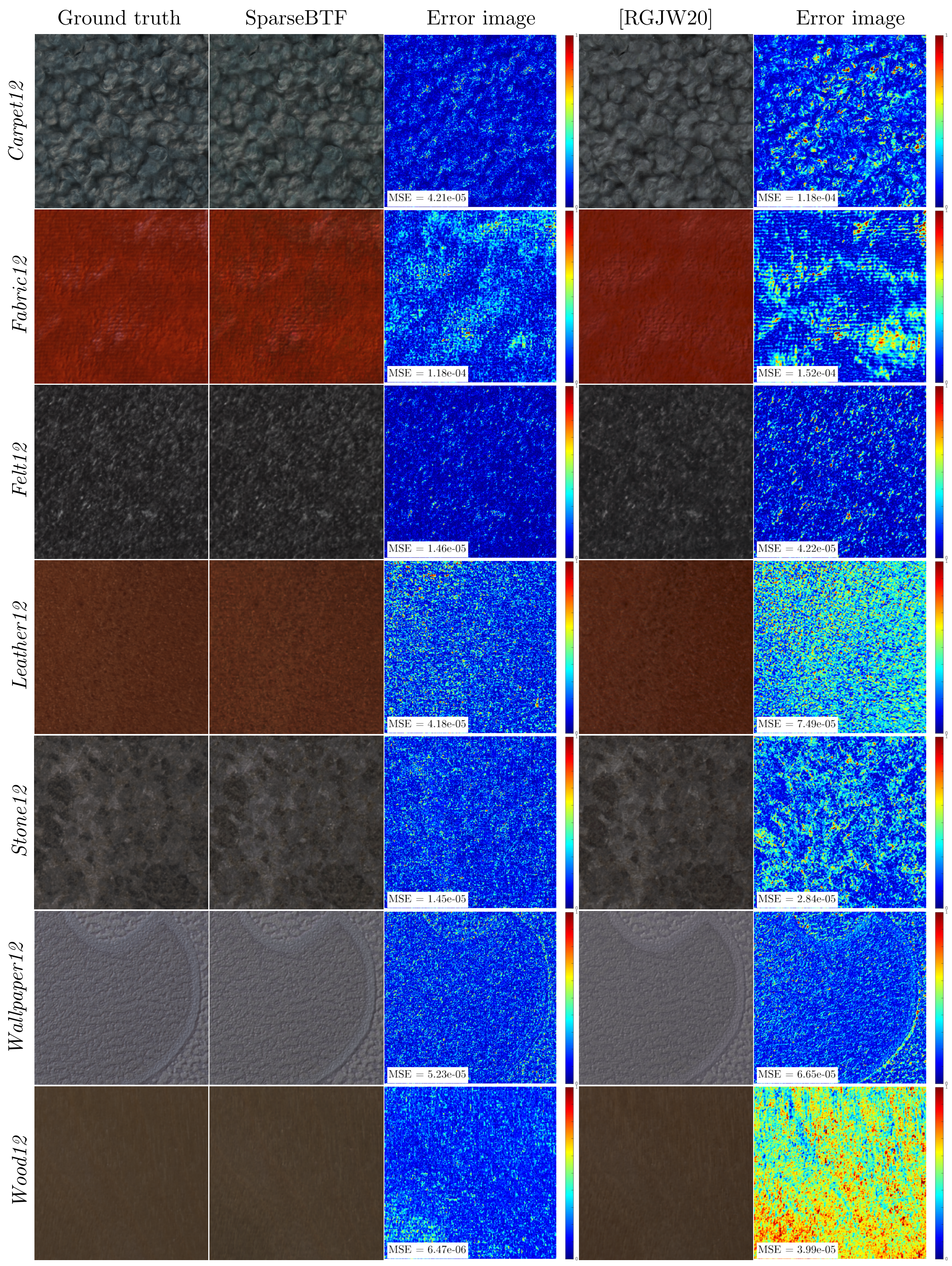

Figure 6: Comparison of reconstructed BTFs using our method and [RGJW20]. Incident azimuth and elevation angles: 127.5°, 22.5°. Observation azimuth and elevation angles: 120°, 52.5°.

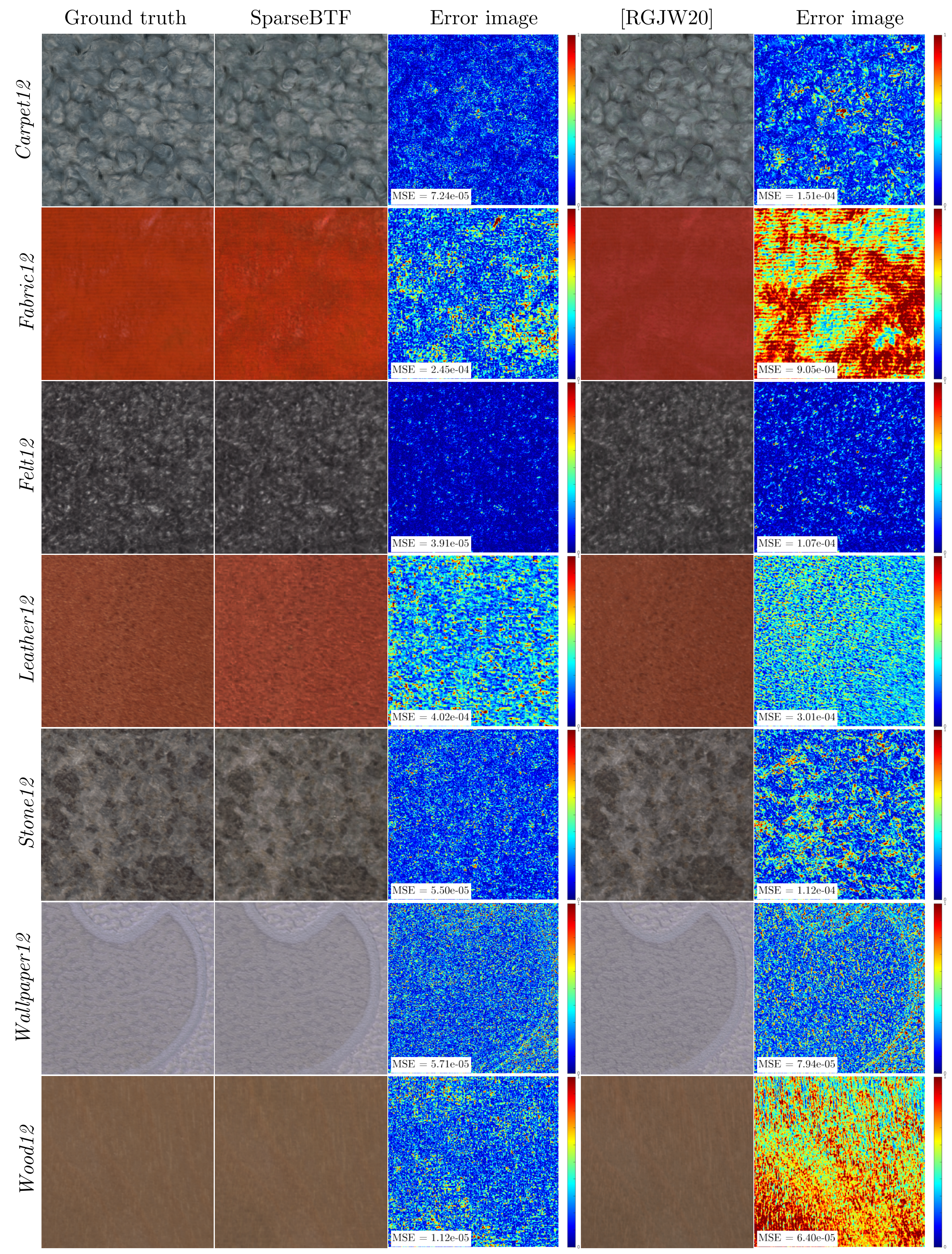

Figure 7: Comparison of reconstructed BTFs using our method and [RGJW20]. Incident azimuth and elevation angles: 195°, 60°. Observation azimuth and elevation angles: 105°, 30°.

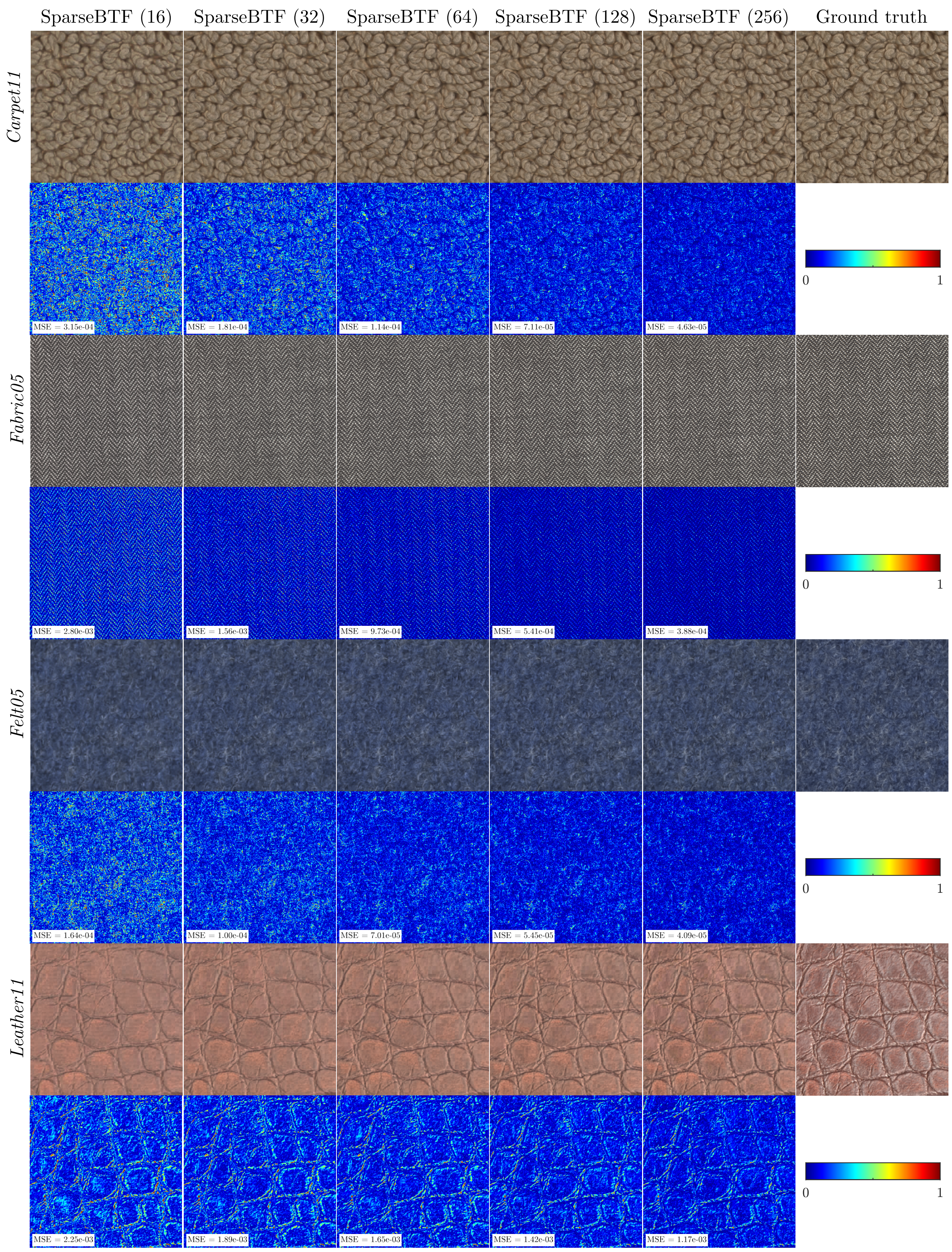
SparseBTF (16)
SparseBTF (32)
SparseBTF (64)
SparseBTF (128)
SparseBTF (256)
Ground truth
Carpet11
MSE = 3.15e-04
MSE = 1.81e-04
MSE = 1.14e-04
MSE = 7.11e-05
MSE = 4.63e-05
0
1
Fabric05
MSE = 2.80e-03
MSE = 1.56e-03
MSE = 9.73e-04
MSE = 5.41e-04
MSE = 3.88e-04
0
1
Felt05
MSE = 1.64e-04
MSE = 1.00e-04
MSE = 7.01e-05
MSE = 5.45e-05
MSE = 4.09e-05
0
1
Leather11
MSE = 2.25e-03
MSE = 1.89e-03
MSE = 1.65e-03
MSE = 1.42e-03
MSE = 1.17e-03
0
1

Figure 8

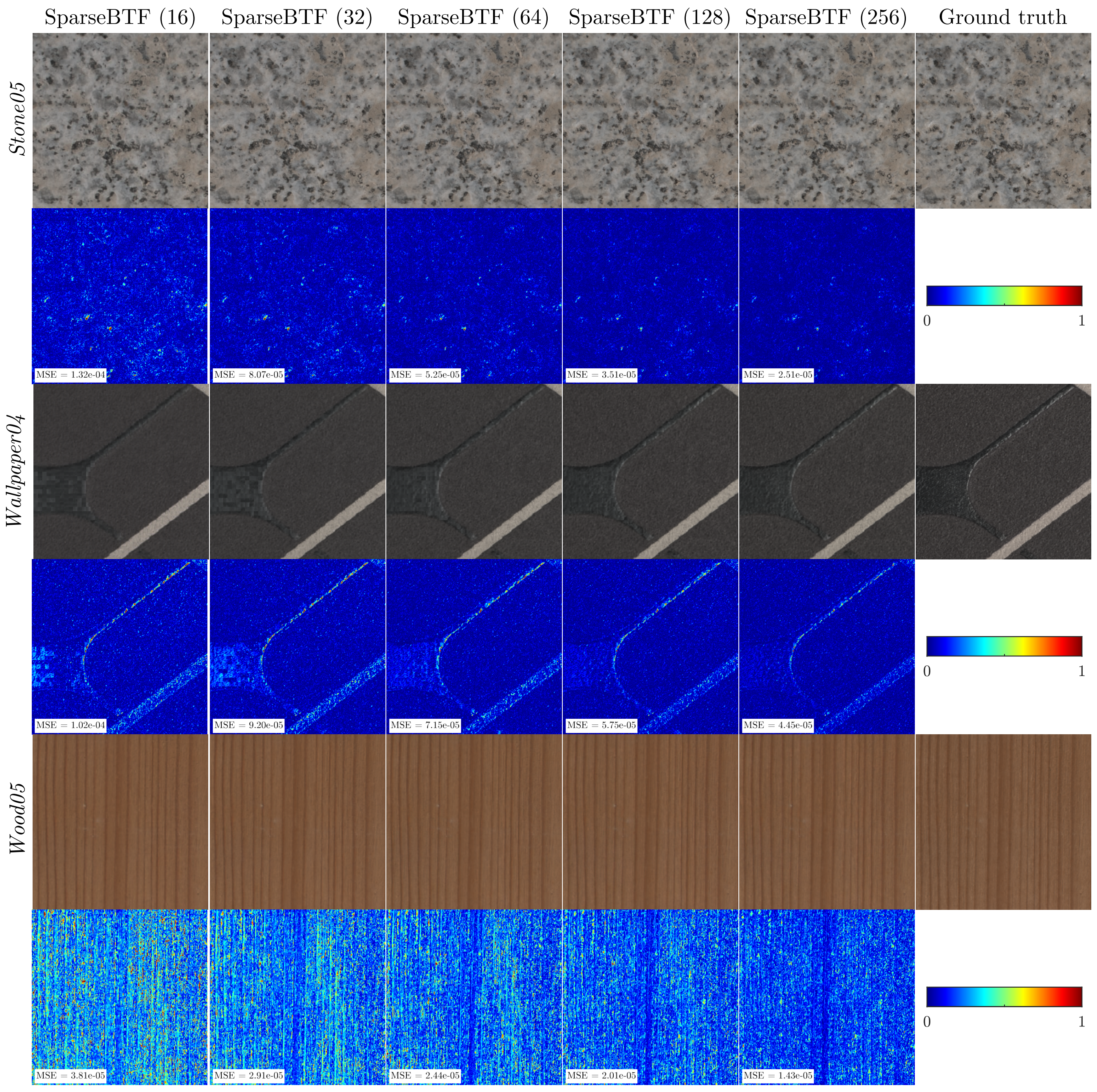

Figure 8: Comparison of reconstructed BTFs using our method with different testing sparsities, i.e., the number of non-zero coefficients. Incident azimuth and elevation angles: 270°, 66.5°. Observation azimuth and elevation angles: 15°, 79°.

References

[AMS08] Aydin T. O., Mantiuk R., Seidel H.: Extending quality metrics to full luminance range images. In *Human Vision and Electronic Imaging XIII* (2008), vol. 6806 of *Proceedings of SPIE*, SPIE. doi:https://doi.org/10.1117/12.765095.

[Pav90] Pavicic M. J.: Convenient anti-aliasing filters that minimize "bumpy" sampling. In *Graphics Gems*, Glassner A. S., (Ed.). Academic Press, 1990, ch. III.1, pp. 144–146.

[PBD*10] Parker S. G., Bigler J., Dietrich A., Friedrich H., Hoberock J., Luebke D., McAllister D., McGuire M., Morley K., Robison A., Stich M.: OptiX: A general purpose ray tracing engine. *ACM Trans. Graph. 29*, 4 (jul 2010). doi:https://doi.org/10.1145/1833349.1778803.

[RGJW20] Rainer G., Ghosh A., Jakob W., Weyrich T.: Unified neural encoding of BTFs. *Computer Graphics Forum 39*, 2 (jun 2020), 167–178.

[RK09] Ruiters R., Klein R.: BTF compression via sparse tensor decomposition. EGSR '09, Eurographics Association, pp. 1181—-1188. doi:10.1111/j.1467-8659.2009.01495.x.

[Sil86] Silverman B. W.: *Density Estimation for Statistics and Data Analysis.* Chapman & Hall, 1986. doi:https://doi.org/10.1201/9781315140919.

[SSK03] Sattler M., Sarlette R., Klein R.: Efficient and realistic visualization of cloth. In *Eurographics Workshop on Rendering (EGWR '03)* (2003), Eurographics Association, pp. 167–177. doi:10.2312/EGWR/EGWR03/167-177.