A Practical Wave Optics Reflection Model for Hair and Fur Supplemental Material

Mengqi (Mandy) Xia[1,2] Bruce Walter[1] Christophe Hery[3]
Olivier Maury[3] Eric Michielssen[4] Steve Marschner[1]

[1]Cornell University
[2]EPFL
[3]Meta Reality Labs
[4]University of Michigan

May 14, 2023

In this paper, we study wave effects from rough fibers with arbitrary 3D microgeometry. We developed a 3D wave optics simulator based on a physical optics approximation (PO), using a GPU-based hierarchical algorithm to significantly accelerate the calculation. To practically handle geometry variations in the scene, we propose a model based on wavelet noise, capturing the important statistical features in the simulation results that are relevant for rendering. We show in the main paper that our compact noise model can be easily combined with existing scattering models to render hair and fur of various colors, introducing visually important colorful glints that were missing from all previous models.

In this supplemental material, we will include more implementation details regarding our simulator, additional validation results on the simulator, and additional validations of the noise-based representation.

Code, data, and the update-to-date version of this document can be downloaded from the project page: https://mandyxmq.github.io/research/wavefiber_3d.html.

Contents

1 Wave simulation in 3D

In this section, we will present more details about our octree-based physical optics (PO) simulator, as well as provide additional validations of our simulator.

1.1 Wave Optics

In wave optics simulation, we compute electric and magnetic fields and analyze how the scattering object affects them. Fields with sinusoidal time variation are called time-harmonic fields, and their mathematical analysis can be simplified by using complex quantities. Phasors of the electric field $\mathbf{E}$ and the magnetic field $\mathbf{H}$ are defined as:

$$\mathbf{E}_{\text{inst}} = \text{Re}(\mathbf{E}e^{j\omega t}), \quad \mathbf{H}_{\text{inst}} = \text{Re}(\mathbf{H}e^{j\omega t}). \tag{1}$$

$\mathbf{E}_{\text{inst}}$ and $\mathbf{H}_{\text{inst}}$ are the instantaneous electric and magnetic fields, and ω is the angular frequency. In what follows, we assume time-harmonic fields and suppress the time dependence $e^{j\omega t}$ unless specified. The wave fields obey the time-harmonic Maxwell's equations, which relate corresponding currents and fields:

$$\begin{aligned} \nabla \times \mathbf{E} &= -\mathbf{M} - j\omega\mu\mathbf{H} \\ \nabla \times \mathbf{H} &= \mathbf{J} + j\omega\epsilon\mathbf{E}. \end{aligned} \tag{2}$$

ϵ and μ are the permittivity and permeability. Here $\mathbf{J}$ and $\mathbf{M}$ are time-harmonic electric and magnetic current densities. In our problems, they are fictitious currents that make the problem easier to solve mathematically.

The object is illuminated by an incident wave, and the incident electric and magnetic fields are denoted as $\mathbf{E}_i$ and $\mathbf{H}_i$. The presence of the scatterer alters the fields, and we call the resulting fields the *total fields*. We denote the total fields outside the object as $\mathbf{E}_1$ and $\mathbf{H}_1$. We can further write $\mathbf{E}_1$ and $\mathbf{H}_1$ as the sums of incident fields and the *scattered fields* $\mathbf{E}_s$ and $\mathbf{H}_s$.

$$\mathbf{E}_1 = \mathbf{E}_i + \mathbf{E}_s, \quad \mathbf{H}_1 = \mathbf{H}_i + \mathbf{H}_s. \tag{3}$$

The scattered fields propagate outward from the scatterer, and we can compute the energy flow from them. The scattered fields will be the key to computing scattering functions.

1.2 Octree-based physical optics

We propose using a physical optics approximation (PO) to efficiently compute electromagnetic wave scattering. PO assumes single scattering, so that the surface currents can be locally computed based on the incident fields and the material properties. PO also assumes that the local geometry is flat enough to be approximated locally as a plane. Thus, we can relate the reflected field and the incident field via surface reflection coefficients in the flat surface reflection computation. Although PO ignores multiple scattering within a single fiber geometry, it allows us to efficiently analyze surface reflection and diffraction from fibers. Our PO simulator is general and works for arbitrary 3D objects. PO makes the surface current computation so efficient that computing the far-field radiation becomes the new bottleneck of our problem. We accelerate the brute force far-field calculation using an octree-based algorithm. We borrowed the idea from the multilevel fast multipole algorithm (MLFMA) [CMSJ01], which is originally used to accelerate the first step in the full wave simulation that computes the surface currents. The algorithm works by first constructing an octree that includes all the sampled points on the surface of the scatterer. Then, it computes the surface currents associated with the surface points in the same way as it does in the non-tree-based PO algorithm. Next, from the bottom (leaf) level of the tree to the top level, we accumulate the far-field contribution along the way, computing all the scattering directions at once. This accelerated algorithm is general and can be applied to accelerate the far-field calculation in exact wave simulations as well.

We implemented the PO algorithm on the GPU by writing CUDA kernels and utilizing the Thrust library. Thrust provides data-parallel primitives such as scan and reduce, and it is also easy to customize operations on arrays using Thrust.

1.2.1 Prior to the far-field compuatation

The first step is to acquire a densely sampled point surface for the scatterer we are interested in. Given a mesh surface, we compute the center $\mathbf{r}'$, normal $\mathbf{n}(\mathbf{r}')$, and area a for each mesh element.

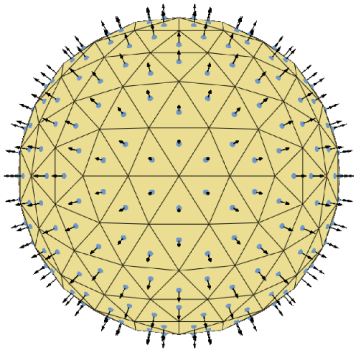


Figure 1: This figure illustrates the surface points and the normals attached to them on a spherical mesh.

The second step is to construct an octree on the GPU using the surface point data. Before constructing the tree, we define the bounding box of the object, which defines the dimensions of the grid on the top level. We also define the maximal depth D of the tree, and the tree will have $D + 1$ levels of grids. Following the quadtree example provided by the CUDA SDK, we make use of CUDA's dynamic parallelism, which enables a kernel to create and synchronize new nested work. For each dimension of the surface point and the normal, as well as the area, we create two buffers that are used in the octree construction algorithm to properly sort the original data. The data that belongs to the same grid of the tree is always in consecutive chunks in the sorted array, following the same ordering of the grid. The CPU first launches a block of threads, and this block will perform the following tasks:

- If the maximum depth of the tree is exceeded, the threads in the block will exit. Before exiting, we ensure that the sorted points are stored in the first buffer and perform a swap when necessary.

- If the tree needs to be further subdivided, we count the number of points in each child (octant). The points contained in the current node are divided into sections and handed to warps of threads, where they use the `__ballot` and `__popc` intrinsics to count the points for each quadrant based on their position relative to the center of the bounding box of the current grid.

- To know the numbers for the block, we run a scan/reduce at the block level.

- The block move points accordingly based on the counted result.

- The block launches eight new blocks, one block per child. Each block will use the same algorithm described here.

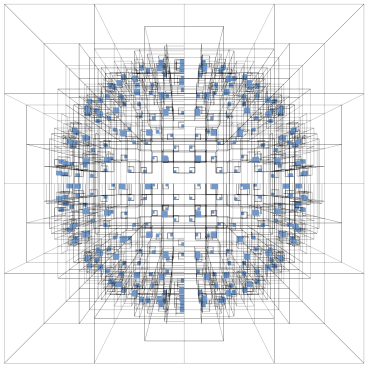


Figure 2: This figure illustrates an octree structure storing the surface points for the sphere.

After constructing the tree and sorting the data, the third step is to compute the surface currents. At each sampled point $\mathbf{r}'$, we consider a local tangent plane that is perpendicular to $\mathbf{n}(\mathbf{r}')$ attached to that point. Then, we compute the surface currents $\mathbf{J}(\mathbf{r}')$ and $\mathbf{M}(\mathbf{r}')$ using the analytic solution for reflection from a flat surface. Specifically, we first build a local polarization frame using the incoming wave direction $\mathbf{e}_i$ and the normal $\mathbf{n}(\mathbf{r}')$, and decompose the incident fields into the sum of parallel and perpendicular polarized fields $\mathbf{E}_i = \mathbf{E}_i^p + \mathbf{E}_i^s$. Then the reflected field and the total field at the boundary can be computed via

$$\begin{aligned} \mathbf{E}_r &= \mathbf{E}_r^p + \mathbf{E}_r^s = F^p \mathbf{E}_i^p + F^s \mathbf{E}_i^s, \\ \mathbf{E}_1 &= \mathbf{E}_i + \mathbf{E}_r. \end{aligned} \tag{4}$$

In the above equations, F^p and F^s are the reflection coefficients in Fresnel's equations for parallel and perpendicular polarization. The reflected field $\mathbf{E}_r$ here is an approximation of the actual scattered field at the surface based on the assumption that the current depends only on the incident field but not on scattering from other parts of the object. Similarly, we can compute the scattered and total magnetic fields. By applying the known relationship between the currents and the total fields, we can compute the surface currents.

$$\mathbf{M} = -\mathbf{n} \times \mathbf{E}_1, \quad \mathbf{J} = \mathbf{n} \times \mathbf{H}_1. \tag{5}$$

Each surface point is associated with six values J_x, J_y, J_z, M_x, M_y, and M_z. This step is computed on the GPU.

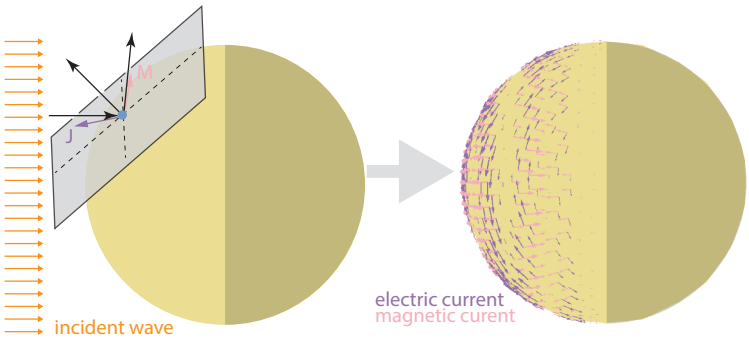


Figure 3: This figure illustrates computing the surface currents using a local tangent plane.

Before performing the actual far-field computation, there are a few precomputation steps we conduct to assist the calculation using the tree structure. Since the octree we construct is complete, we always have $(8^{D+1}-1)/7$ nodes. However, as we only care about points on the surface of an object, most of the nodes in the tree are empty. Therefore, after constructing the tree, we compute a list of active nodes that contain at least one surface point in preparation for efficient traversal and far-field calculation. Specifically, we compute two arrays: one for storing all the indices of active nodes, and another for storing the number of active nodes for each level of the tree.

Then we follow [SC01] and determine the number of directions for each level of the tree. Specifically, we use the formula

$$M = kd + 6(kd)^{1/3}, \tag{6}$$

where M is the number of directions, k is the wave number, and d is the size of the corresponding dimension. For fiber scattering problems, we determine M_θ and M_ϕ for the θ and ϕ angles, and d is the enlarged length and diameter, respectively. We set an enlargement ratio, and in practice, a ratio of 1.2 works well for the fiber problems we tested. Next, we create plans for the Fast Fourier Transform (FFT) calculation based on the number of directions for each level. More information can be found in the cuFFT documentation. This is for interpolating the fields as we go up the tree.

The last precomputation step is to compute the translation from the center of the child node to the center of the current node. The translation kernel is $e^{jk_0(\mathbf{c}_{\text{child}}-\mathbf{c})\cdot\hat{\mathbf{r}}}$, where $\mathbf{c}_{\text{child}}-\mathbf{c}$ is the vector from child to self, and $\hat{\mathbf{r}}$ is each direction in the current direction set. Since the grids on the same level are of the same size, only eight complex numbers need to be computed and stored for each level.

Regarding what we have discussed so far, the construction of the tree, as well as the

active node precomputation is only needed per geometry. In other words, we do not need to repeat these calculations as we change incident directions or wavelengths or the material property of the object. The FFT plans only depend on the number of directions we want to compute. The surface currents depend on the incident direction, wavelength, as well as material property, and need to be recomputed whenever these parameters change. The value of the translation kernel (from the child to the grid itself) only needs to be recomputed when the wavelength changes. One trick to accelerate the PO calculation is that we can discard the surface points that we are sure of in geometric shadow, as they will have zero surface currents and will not contribute to the scattered fields. In that case, constructing the tree and processing it to obtain a list of active nodes are incident-direction dependent.

1.2.2 Far-field scattering based on the octree structure

After computing the surface currents, we effectively transform the original scattering problem into a radiation problem using Huygens's principle: The surface currents serve as the equivalent sources to the original scattering problem and radiate the same scattered fields that we aim to compute. We can mathematically write down the radiation from the surface currents $\mathbf{J}$ and $\mathbf{M}$. In the far field, the scattered electric field at point $\mathbf{r}$ is [Gib21]:

$$\mathbf{E}_s(\mathbf{r}) = j\omega\mu_0 \frac{e^{-jk_0R}}{4\pi R}\hat{\mathbf{r}} \times \int_S \left[\hat{\mathbf{r}} \times \mathbf{J}(\mathbf{r}') + \frac{1}{Z_0}\mathbf{M}(\mathbf{r}')\right] e^{jk_0\mathbf{r}'\cdot\hat{\mathbf{r}}} d\mathbf{r}', \tag{7}$$

where k_0 is wave number, $\hat{\mathbf{r}}$ is the scattering direction, $R = |\mathbf{r}|$, S is the surface of the scatterer and $\mathbf{r}'$ is a point on the surface and $Z_0 = \sqrt{\frac{\mu_0}{\epsilon_0}}$ is free space impedance. $\mathbf{H}_s$ can be easily computed once we know $\mathbf{E}_s$, since waves propagate radially and locally follow a planar wavefront in the far field [Jac21]. We define $\mathbf{E}_s^{\text{far}}(\hat{\mathbf{r}})$ and $\mathbf{H}_s^{\text{far}}(\hat{\mathbf{r}})$ by

$$\mathbf{E}_s(\mathbf{r}) = \frac{e^{-jk_0R}}{R}\mathbf{E}_s^{\text{far}}(\hat{\mathbf{r}}), \quad \mathbf{H}_s(\mathbf{r}) = \frac{e^{-jk_0R}}{R}\mathbf{H}_s^{\text{far}}(\hat{\mathbf{r}}), \tag{8}$$

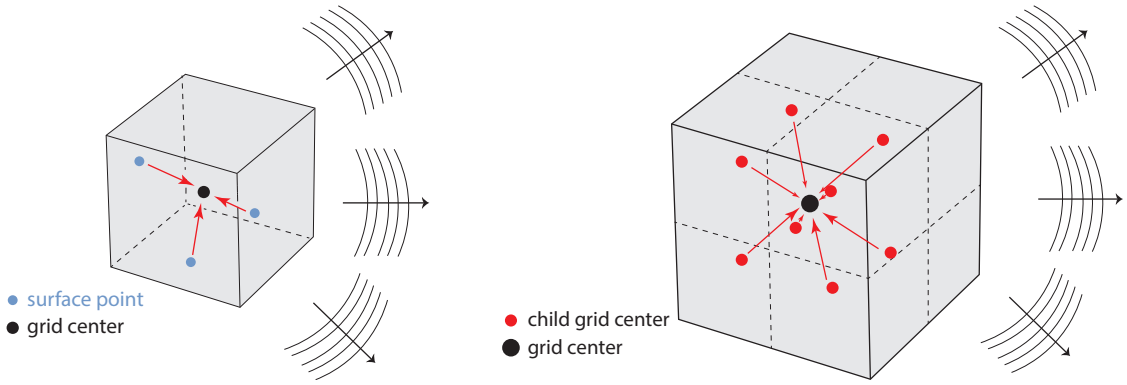
where $\mathbf{E}_s^{\text{far}}$ and $\mathbf{H}_s^{\text{far}}(\hat{\mathbf{r}})$ only depend on the scattering direction $\hat{\mathbf{r}}$ but not R.

To compute the far-field scattering using the tree, we first multiply the surface currents with the corresponding surface area a at the same point to numerically integrate the far-field contribution. We then loop through the tree levels on the CPU, from the bottom (leaf) level to the top level, and compute far-field scattering on the GPU.

At the leaf level (Figure 4 a), we have the coarsest set of directions. For each direction and each surface point $\mathbf{r}'$, we compute $e^{jk_0(\mathbf{r}'-\mathbf{c})\cdot\hat{\mathbf{r}}}$, where $\mathbf{c}$ is the center of the grid containing the surface points. We multiply the above value with the surface currents and denote the resultant arrays corresponding to each dimension of the electric and magnetic currents as

A_x, A_y, A_z, F_x, F_y, and F_z. This step is performed on the GPU. Then, we use the reduce operation in Thrust and sum the far-field contributions for each leaf node.

From the second lowest level to the top level (Figure 4 b), we have the same set of operations: we loop through the active nodes on the current level using the non-empty node list we precomputed. For each active node, we loop through its active children nodes. Looping through active children is straightforward because we constructed a complete tree in the first place. Given a current node with index m on level q, the index of the ith child is $N_{end} + 8(m - N_{start}) + i$, where q takes values from 0 to D, and i takes values from 0 to 7. $N_{start} = (8^q - 1)/7$ and $N_{end} = (8^{q+1} - 1)/7$ are the start index and the end index of this level, respectively. We check for each of the 8 children, whether it is in the active node list. For each active child node, we first upsample its far-field contribution from the previous set of directions to the current set. The upsampling operation is performed on A_x, A_y, A_z, F_x, F_y, and F_z arrays independently. For each array, we need to upsample in both θ and ϕ dimensions. We use Fast Fourier Transform (FFT) to perform interpolation and upsampling [Sar03]. Specifically, for each of the field arrays, we first perform a forward FFT on the ϕ dimension, zero-padding the transformed array, and then perform an inverse FFT. Then, we repeat the same process for the θ dimension. For upsampling each child's scattering contribution, we need in total 12 forward FFT operations and 12 inverse FFT operations. After we upsampling the fields for the new set of directions, we translate the contribution from the child's grid center to the current grid center by multiplying the precomputed translation kernel. This is followed by adding the child's contribution to the current grid. We loop through the nodes and children nodes on the CPU and perform upsampling and summing on the GPU. We perform the same set of operations as we ascend the tree until we reach the top level. At the top level, we translate and accumulate the contribution to the origin to obtain the desired total scattering fields.



(a) Leaf node computation.

(b) Aggregating the far-field contribution to the top level as we go up through the tree.

Figure 4: Traverse the tree and compute far-field scattering.

After we obtain $\mathbf{E}_s^{\text{far}}(\hat{\mathbf{r}})$ and $\mathbf{H}_s^{\text{far}}(\hat{\mathbf{r}})$, which define the scattered fields $\mathbf{E}_s$ and $\mathbf{H}_s$ via (8), we can use them to compute scattered intensity. In electromagnetics, the time-averaged Poynting vector [Jac21]

$$\langle \mathbf{S} \rangle = \tfrac{1}{2}\text{Re}(\mathbf{E} \times \mathbf{H}^*) \tag{9}$$

plays a role analogous to vector irradiance in radiometry: given a differential area dA at location $\mathbf{r}$ with a unit normal vector $\hat{\mathbf{n}}$, then the net radiant flux through dA is $\langle \mathbf{S}(\mathbf{r}) \rangle \cdot \hat{\mathbf{n}}\, dA$. To compute the far-field intensity, consider irradiance on the inside of a sphere of radius R where $kR \gg 1$. Since waves propagate radially [Jac21] in the far field, $\langle \mathbf{S} \rangle$ and $\hat{\mathbf{n}}$ are both parallel to $\hat{\mathbf{r}}$ and the scattered intensity $I_s(\hat{\mathbf{r}})$ can be computed from $\mathbf{E}_s^{\text{far}}(\hat{\mathbf{r}})$ and $\mathbf{H}_s^{\text{far}}(\hat{\mathbf{r}})$ as:

$$\begin{aligned} I_s(\hat{\mathbf{r}}) &= (\langle \mathbf{S}(\mathbf{r}) \rangle \cdot \hat{\mathbf{n}}) R^2 = |\langle \mathbf{S}(\mathbf{r}) \rangle| R^2 \\ &= \left| \frac{1}{2} \text{Re}(\mathbf{E}_s^{\text{far}}(\hat{\mathbf{r}}) \times \mathbf{H}_s^{\text{far}}(\hat{\mathbf{r}})^*) \right| \\ &= \frac{1}{2} \sqrt{\frac{\epsilon_0}{\mu_0}} \left| \mathbf{E}_s^{\text{far}}(\hat{\mathbf{r}}) \right|^2 . \end{aligned} \tag{10}$$

1.3 Simulation validation

In Figure 5, we compare our simulator with Mie scattering on spheres of radii $1, 2, 5, 10, 20$ and 50μm. The incident plane wave travels in $+x$ direction, and we report the normalized scattered intensity in the $x-y$ plane, for Transverse Magnetic (TM) and Transverse Electric (TE) polarizations, as well as unpolarized results. For TM (TE) polarization, the incident electric (magnetic) field aligns with the $z-$axis. We observe that as the size of the sphere

increases, PO becomes more accurate. This is because the local tangent plane becomes a better approximation as the curvature decreases. Human hair fibers usually have diameters of 50-100 μm, where PO serves as a very good approximation.

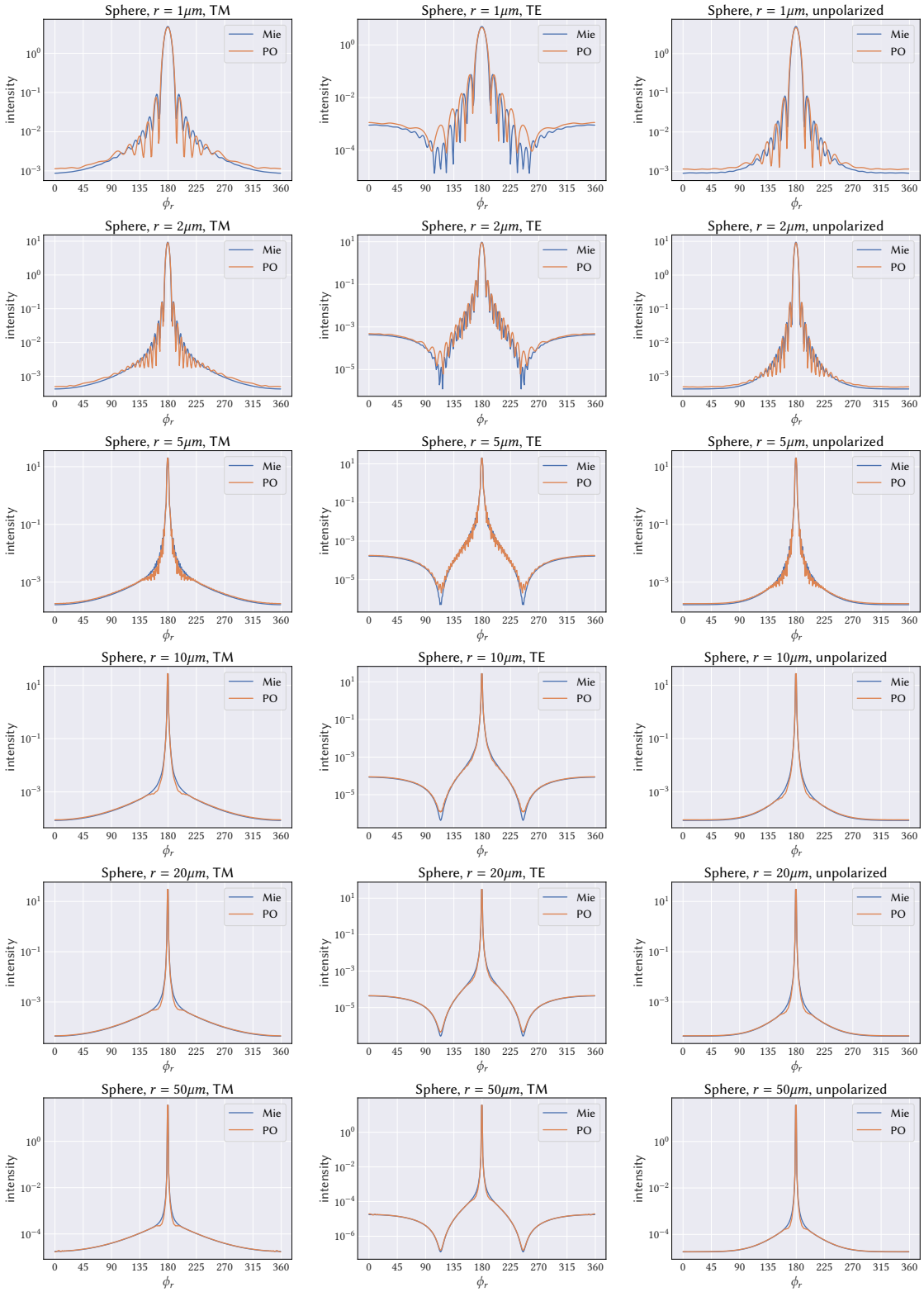


Figure 5: We compare our physical optics based simulator with Mie scattering on varying-sized spheres. For TM and TE polarized, as well as unpolarized results, we observed that PO provides very good accuracy for spheres whose radius is large compared to the wavelength (400 nm).

2 Noise-based representation

In this section, we provide additional results that validate our noise-based representation. In Figure 6, we demonstrate that we can modulate the frequency bands and generate noise patterns that have the desired Gaussian autocorrelation functions (ACFs). Specifically, we first generate a series of Gaussian ACFs with varying standard deviation σ. In our test, we vary σ_x and always set $\sigma_y = 0.5\sigma_x$. We define a square patch with length $L = 10$, a resolution $N = 200$ in both dimensions, and we generate 100 noise patterns on the surface patch for each of the four frequency bands. Next, we compute the ACF for each frequency band and evaluate the target Gaussian ACF. We form a non-negative least square problem to solve for non-negative weights of the ACF of the bands so that their weighted sum equals the target Gaussian ACF. We compare the weighted ACFs of the noise, as well as the ACF of the weighted sum of the noise, to the target Gaussian ACF and show good accuracy in the fitting. The fact that the weighted ACFs of the noise bands and the ACF of the weighted sum of the noise agree well with each other validates Lemma 5.1 in the main paper.

In Figures 7, 8, 9, and 10, we compare our synthesized scattering patterns with the simulated ones for two different wavelengths, 400 nm and 688 nm, and six different incident directions ($\theta_i = 0°, 30°,$ and $60°$; $\phi_i = 0°$and $90°$). The synthesized patterns were generated by multiplying the noise patterns with the corresponding averaged components. The average components used here for the synthesized results are the ensemble averages of the simulation results for each fiber distribution, incident direction, and wavelength equal to 400 nm. We demonstrate that the synthesized results capture the speckle size and shape very well. In Figures 11, 12, 12, and 14, we convert the spectral results into RGB and demonstrate that our synthesized results have similar hues to those in the simulated results. In these tests, we observe structured patterns in the simulation results, which cannot be treated as fully developed speckles. Therefore, our synthesized noise cannot capture them very well. We also observe that fitting is more accurate for small incident theta angles.

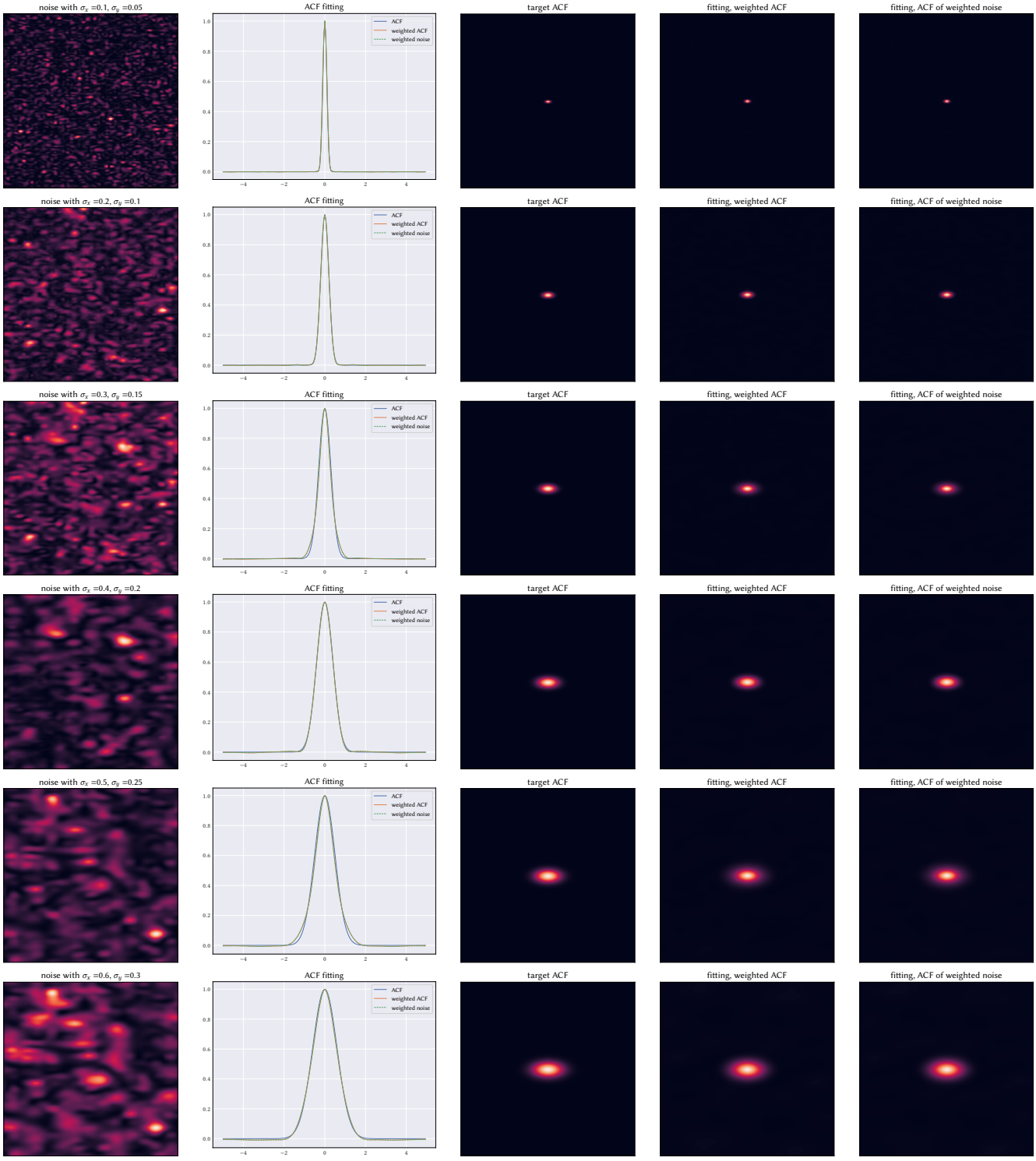
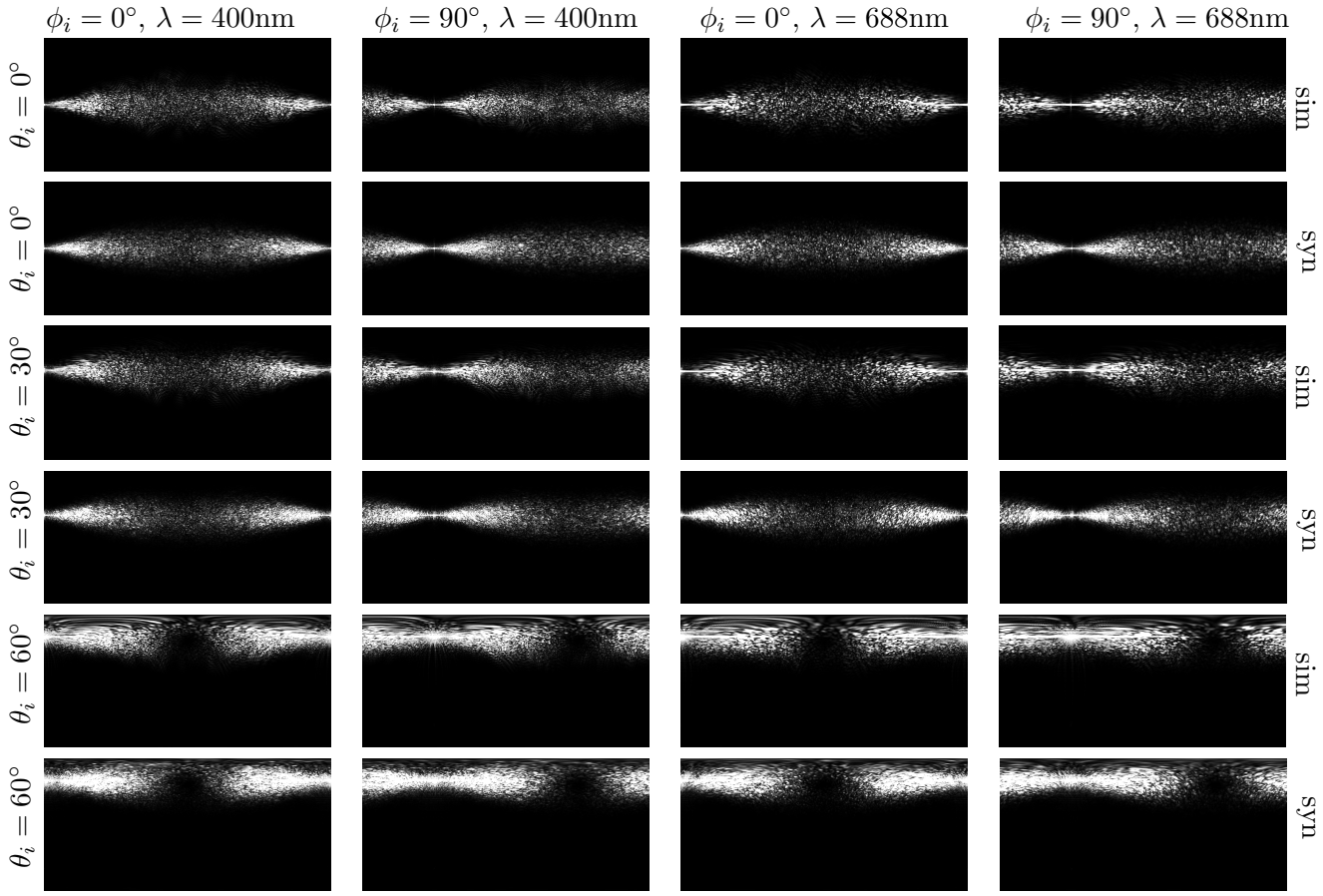


Figure 6: We demonstrate that we can modulate frequency bands and achieve the desired Gaussian autocorrelation functions (ACFs) with varying widths. We compare the weighted sum of the ACFs and the ACF of the weighted noise to the target ACFs. In practice, the weighted sum of the ACFs from individual bands and the ACF of the weighted noise agree with each other quite well, and our fitting achieves high accuracy.



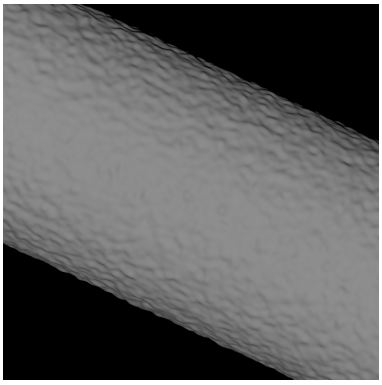


Figure 7: The rough fiber segment we used was generated by displacing a circular cylinder segment with a radius of 40µm and a length of 50µm. We wrapped a rough surface patch with dimensions corresponding to the circumference of the circular cross-section and the length of the cylinder, respectively. The surface roughness is 0.1 and the refractive index is 1.55. We compare the synthesized noise patterns to the simulation results for different incident directions and wavelengths. The synthesized results are computed by taking the product of the averaged simulation results of 50 fiber instances and the fitted wavelet noise pattern. This also applies to the other cases on the following pages. The synthesized results produce similar granular patterns as the simulated results.

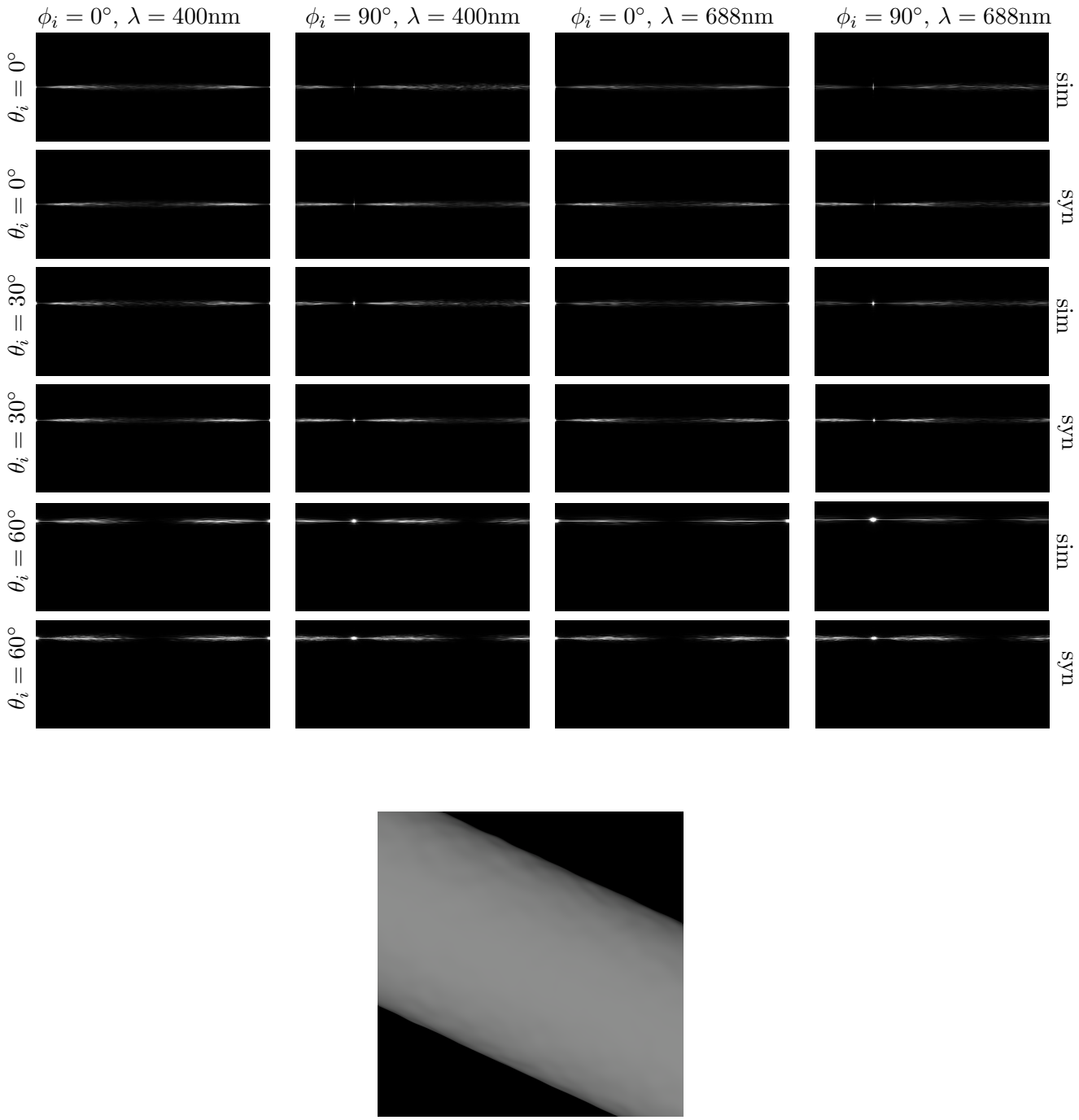
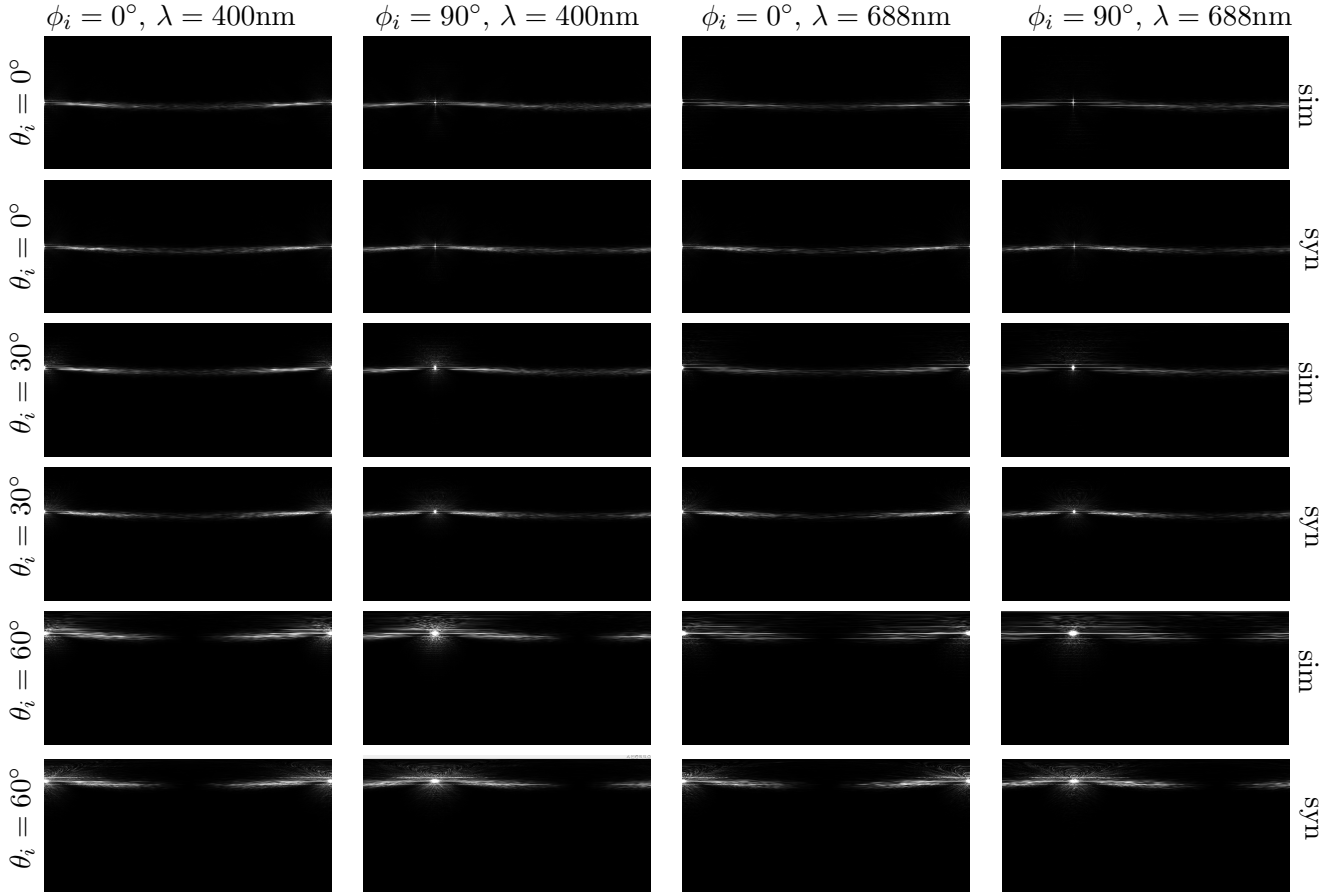


Figure 8: The rough fiber segment we used was generated by displacing an elliptical cylinder segment with major and minor radii 40µm and 35µm. The length of the segment is 400µm. We wrapped a rough surface patch with dimensions corresponding to the circumference of the elliptical cross-section and the length of the cylinder, respectively. The surface roughness is 0.02 and the refractive index is 1.55. The synthesized result captures the anisotropic shape of the speckles well.



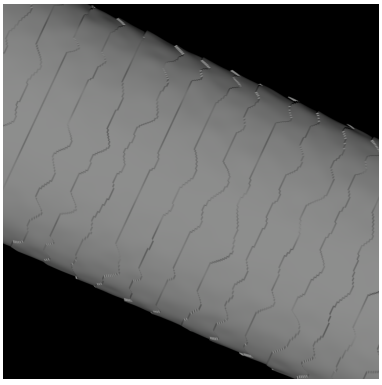
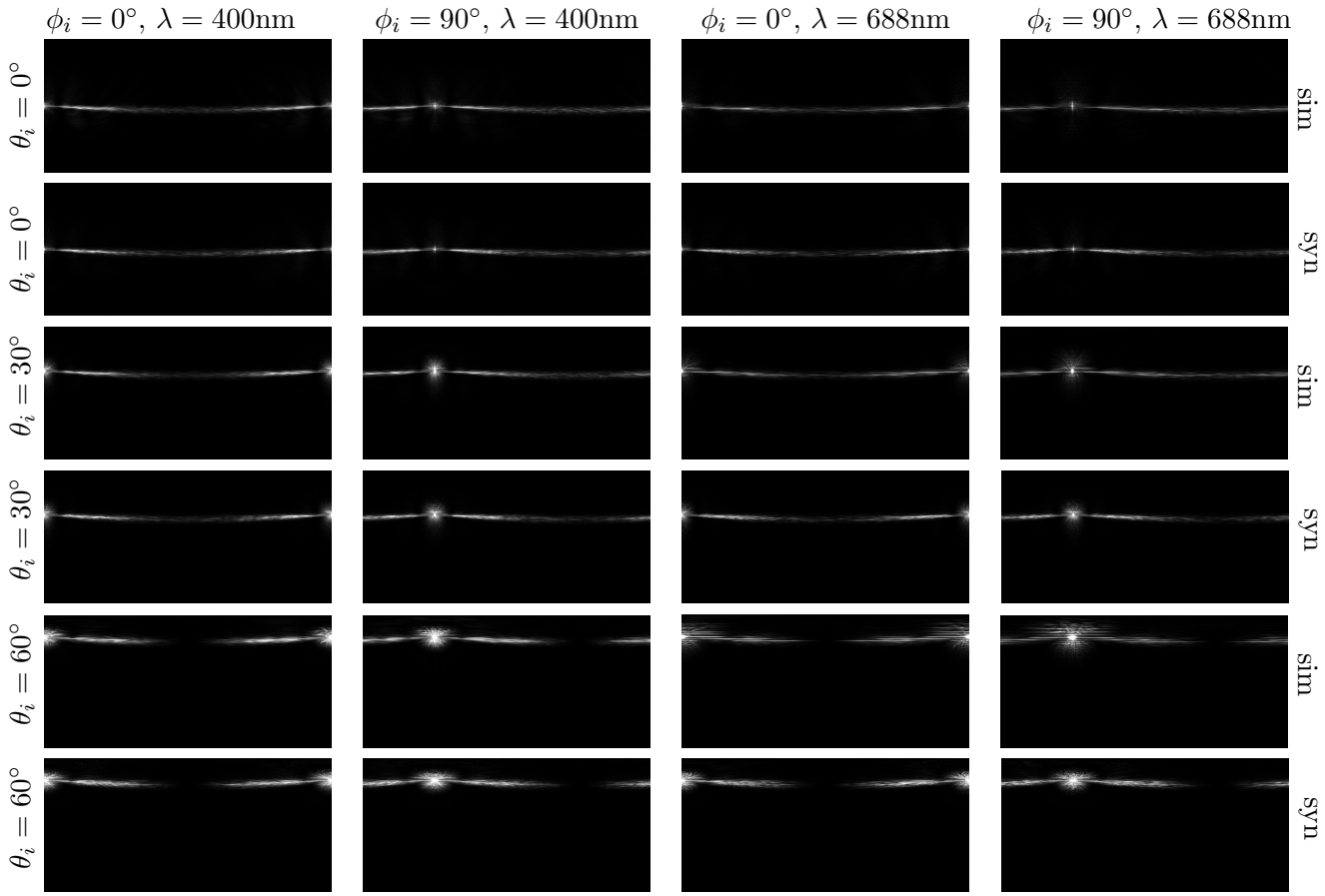


Figure 9: This rough fiber segment was generated by displacing an elliptical cylinder segment with major and minor radii 40µm and 35µm. The length of the segment is 400µm. In addition to wrapping a rough surface patch around the cylinder, we simulate the cuticle scales by introducing a 1D Gaussian random structure that describes the irregular edge of the cuticle. The surface roughness is 0.02 and the 1D Gaussian random structure has a roughness 0.5. The cuticle angle is -3 degrees and the refractive index is 1.55. There are some structural mismatches around the forward scattering highlight. Our synthesized patterns in general are similar to the simulation results.



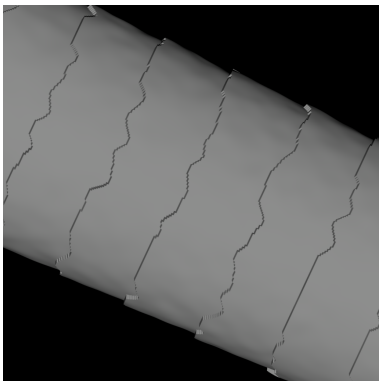


Figure 10: This rough fiber segment has the same surface roughness parameters as the one in Figure 9, except that it has a larger gap between cuticle layers.

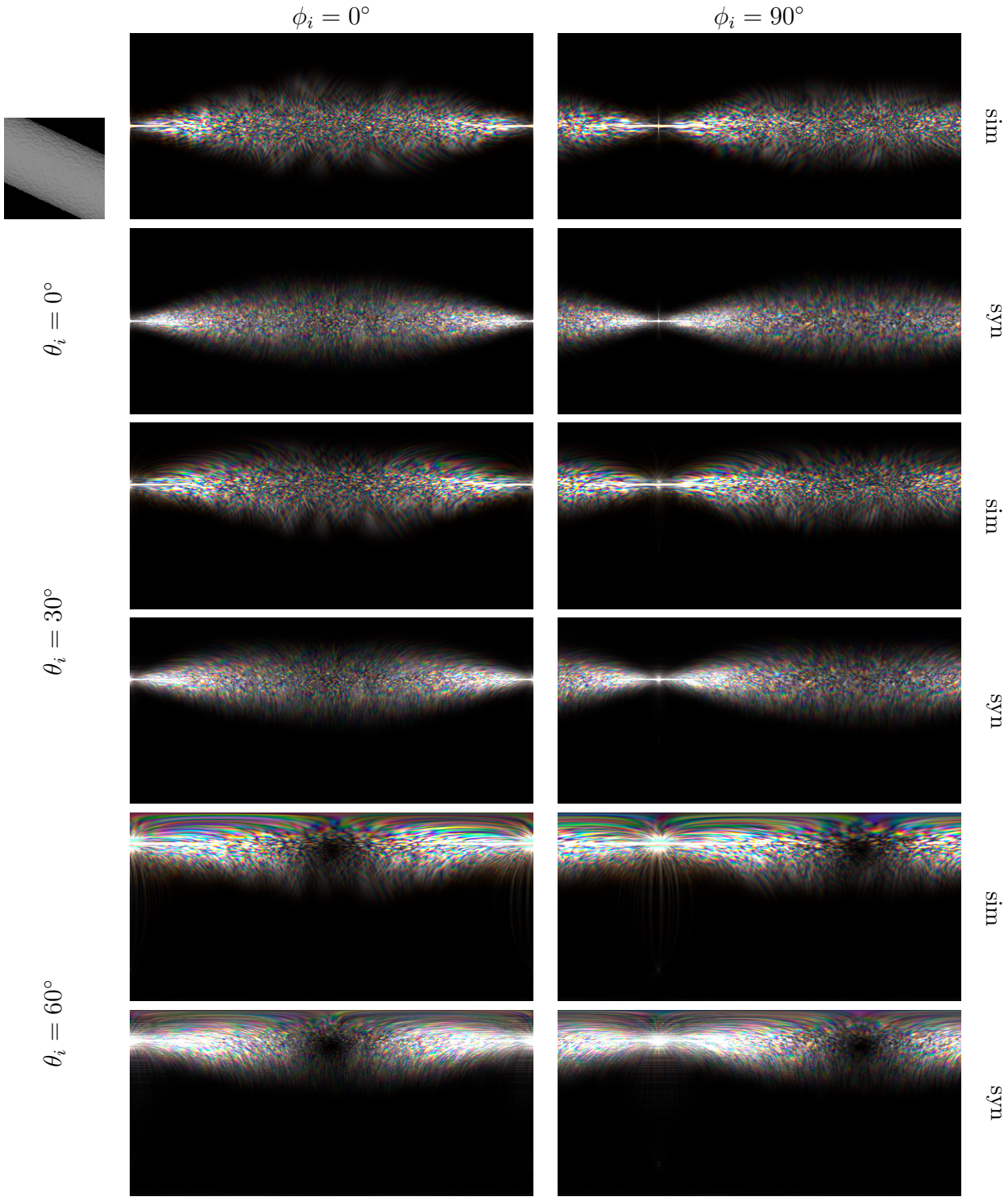


Figure 11: We compare the simulated and synthesized patterns for the first type of rough fibers, whose single wavelength results are shown in Figure 7. We compute the spectral scattering patterns and convert them to RGB.

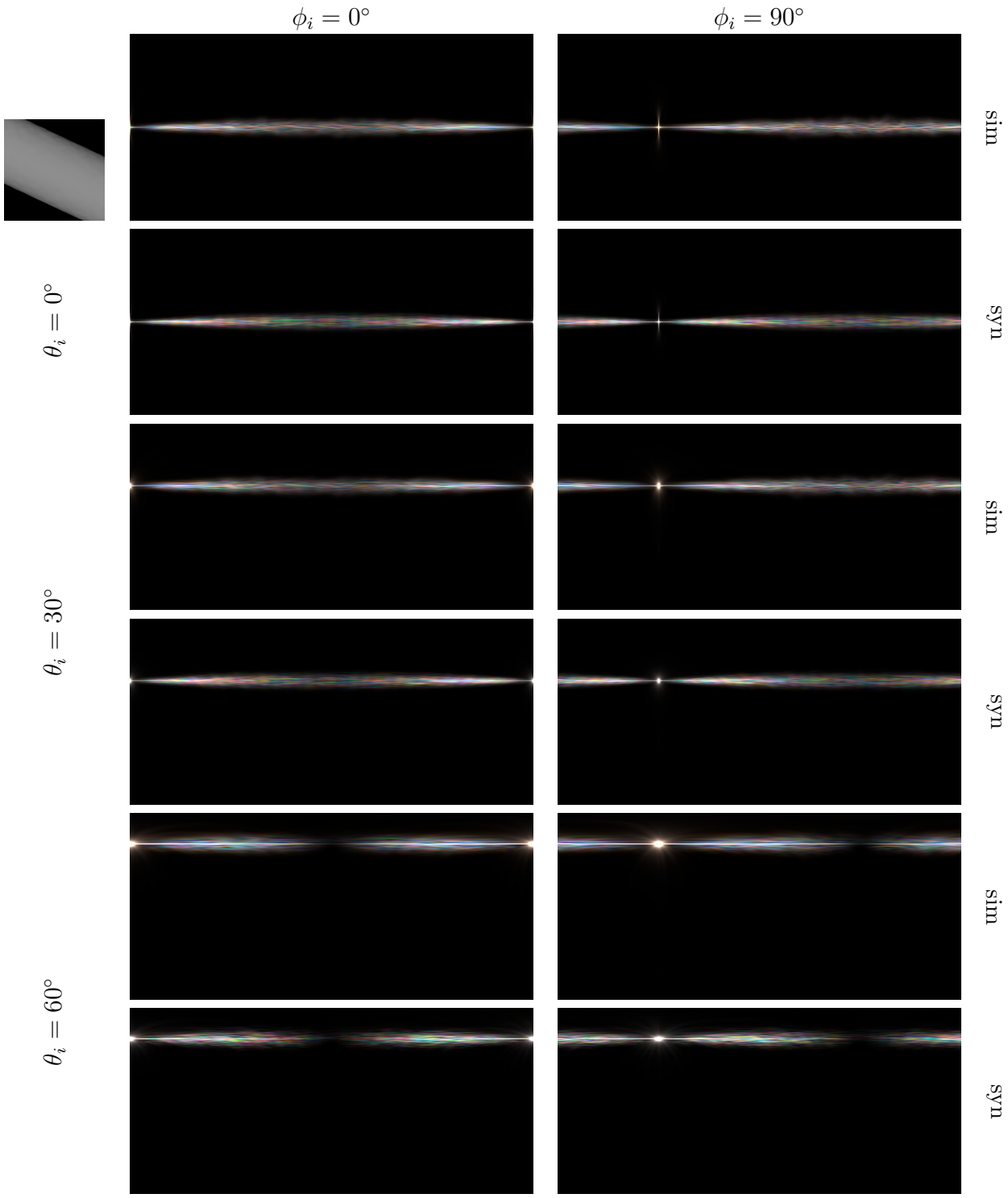


Figure 12: We compare the simulated and synthesized patterns for the second type of rough fibers, whose single wavelength results are shown in Figure 8. We compute the spectral scattering patterns and convert them to RGB.

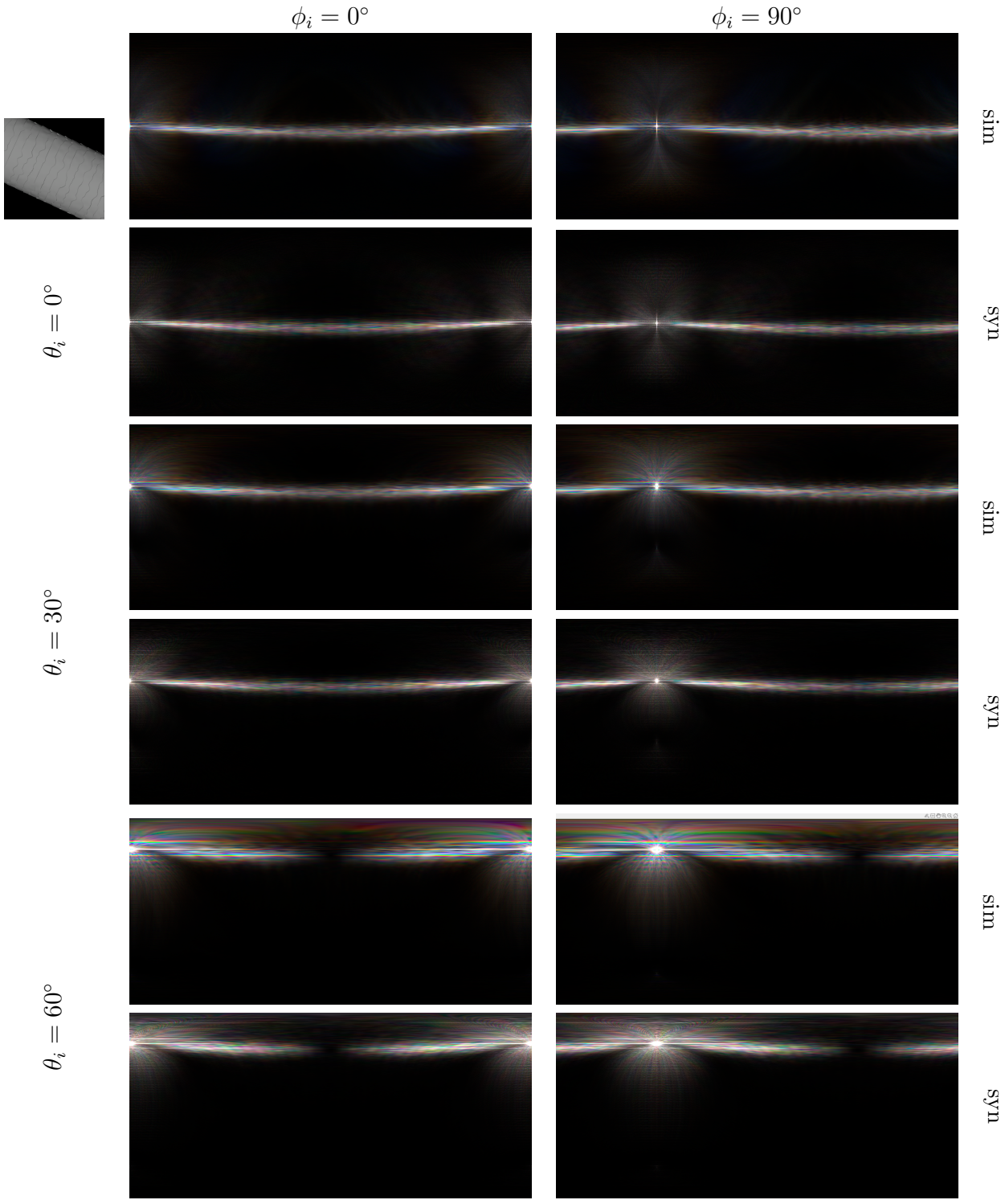


Figure 13: We compare the simulated and synthesized patterns for the third type of rough fibers, whose single wavelength results are shown in Figure 9. We compute the spectral scattering patterns and convert them to RGB.

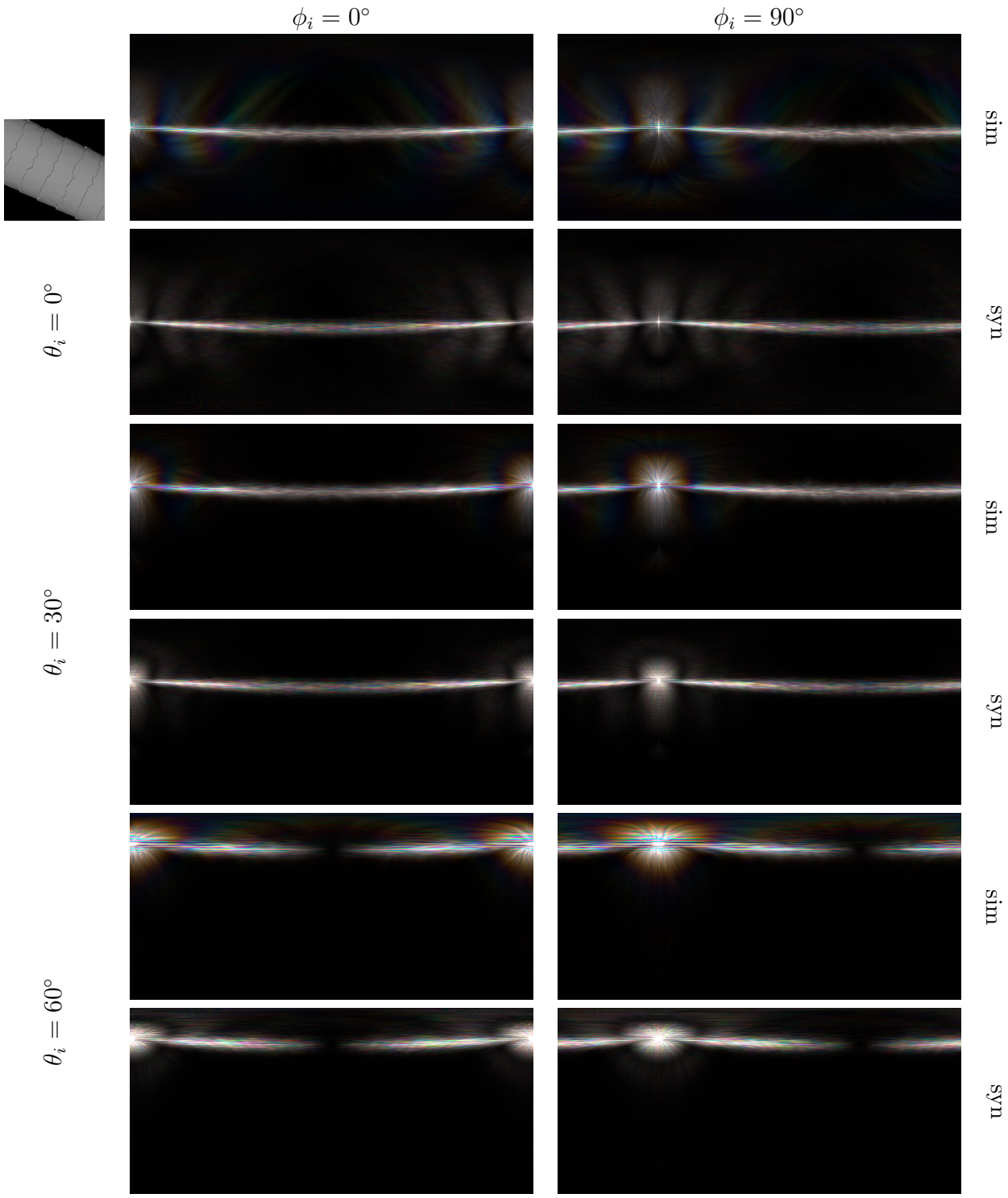


Figure 14: We compare the simulated and synthesized patterns for the last type of rough fibers, whose single wavelength results are shown in Figure 10. We compute the spectral scattering patterns and convert them to RGB.

References

[CMSJ01] Weng Cho Chew, Eric Michielssen, JM Song, and Jian-Ming Jin. *Fast and efficient algorithms in computational electromagnetics*. Artech House, Inc., USA, 2001. 1.2

[Gib21] Walton C Gibson. *The method of moments in electromagnetics*. Chapman and Hall/CRC, USA, 2021. 1.2.2

[Jac21] John David Jackson. *Classical electrodynamics*. John Wiley & Sons, 2021. 1.2.2, 1.2.2, 1.2.2

[Sar03] Jukka Sarvas. Performing interpolation and anterpolation entirely by fast fourier transform in the 3-d multilevel fast multipole algorithm. *SIAM Journal on Numerical Analysis*, 41(6):2180–2196, 2003. 1.2.2

[SC01] Jiming Song and Weng Cho Chew. Error analysis for the truncation of multipole expansion of vector green's functions [em scattering]. *IEEE microwave and wireless components letters*, 11(7):311–313, 2001. 1.2.1