A Procedural, Multiscale and Real-time Feather Appearance Model

XIANG CHEN*, School of Software, Shandong University, China
BIN CHEN*, University of Melbourne, Australia
SHOUYI WANG, School of Software, Shandong University, China
ZAHRA MONTAZERI, University of Manchester, United Kingdom
LINGQI YAN, MBZUAI, United Arab Emirates
LU WANG, School of Software, Shandong University, China
JUNQIU ZHU†, School of Software, Shandong University, China

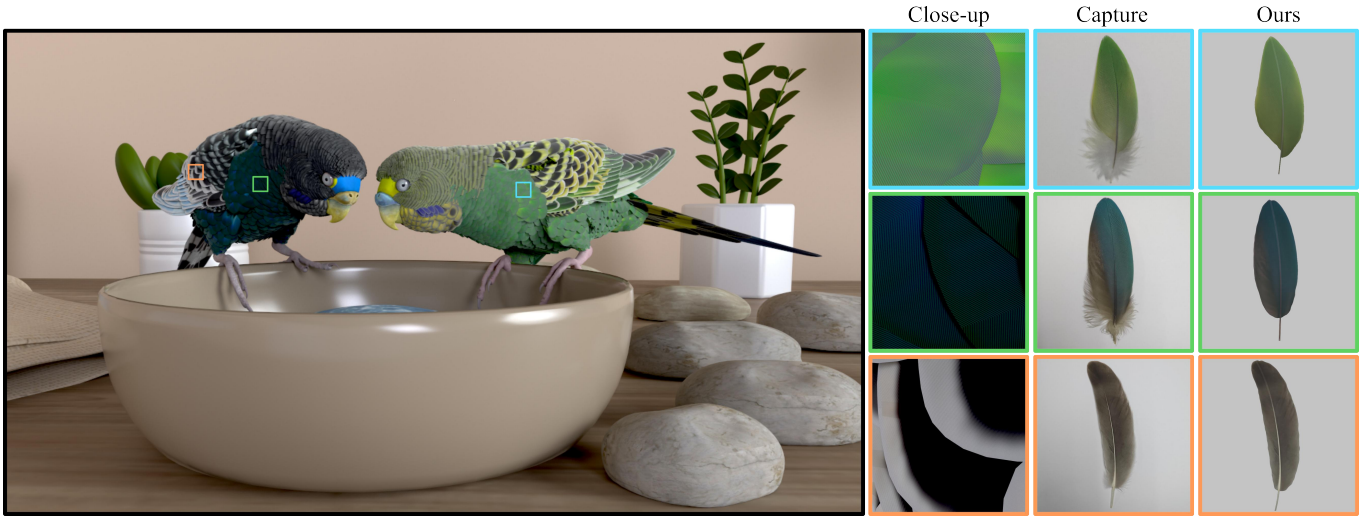

Fig. 1. Our method enables real-time rendering of complex, realistic feathers. We demonstrate a full scene containing two parrots with approximately 6,250 feathers. An explicitly modeled fiber-geometry representation would require 1.91 TB of memory, whereas our method uses only 200 MB, which is only about 0.02% of the memory footprint. We also present close-up views of various feather types, comparing captured reference images with our rendered results, which demonstrate accurate reproduction of real feather appearance.

We propose a complete pipeline for modeling and rendering realistic bird feathers from a single photograph, achieving both high visual fidelity and practical efficiency. Given a single input image of a feather, our approach extracts the feather's shaft curve, outline, and albedo, then reconstructs a compact hierarchical representation in a planar/curve (UV) domain. This representation encodes fine barb and barbule details procedurally, enabling continuous multiscale rendering with correct self-shadowing and masking. We analyze the appearance phenomena of different feathers and propose a new feather scattering model for non-iridescent feathers (e.g., parrot feathers), while introducing an additional sheen lobe to capture the distinctive fluffy rim-lighting effect. Our pipeline produces consistent, realistic results under arbitrary lighting and viewing conditions, and achieves real-time performance with a minimal memory footprint (0.02% of explicit-fiber geometry models), making it a practical solution for digital feather rendering without compromising realism.

CCS Concepts: • **Computing methodologies** → **Rendering**.

ACM Reference Format:
Xiang Chen, Bin Chen, Shouyi Wang, Zahra Montazeri, Lingqi Yan, Lu Wang, and Junqiu Zhu. 2026. A Procedural, Multiscale and Real-time Feather Appearance Model . *ACM Trans. Graph.* 45, 4, Article 93 (July 2026), 17 pages. https://doi.org/10.1145/3811328

1 Introduction

Bird feathers exhibit rich visual appearances due to their hierarchical structure and complex light-scattering behavior. Each feather consists of a central shaft (rachis) with numerous barbs growing from either side, and each barb further branches into dense barbules, forming a multi-scale hierarchical geometric structure. This hierarchy, together with the feather's intrinsic material properties, gives

*Both authors contributed equally to this research.
†Corresponding author.

Authors' Contact Information: Xiang Chen, xiang_chen@mail.sdu.edu.cn, School of Software, Shandong University, China; Bin Chen, bin.chen@unimelb.edu.au, University of Melbourne, Australia; Shouyi Wang, shouyiw439@gmail.com, School of Software, Shandong University, China; Zahra Montazeri, zahra.montazeri@manchester.ac.uk, University of Manchester, United Kingdom; Lingqi Yan, lingqi.yan@mbzuai.ac.ae, MBZUAI, United Arab Emirates; Lu Wang, luwang_hcivr@sdu.edu.cn, School of Software, Shandong University, China; Junqiu Zhu, zhujunqiu@mail.sdu.edu.cn, School of Software, Shandong University, China.

ACM 1557-7368/2026/7-ART93
https://doi.org/10.1145/3811328

rise to complex scattering phenomena, producing a soft appearance and strong view-dependent effects. In some species, feathers even display vivid structural coloration. Accurately reproducing these phenomena is crucial for realistic digital birds in film, games, and virtual reality; however, doing so remains challenging because of the tightly coupled geometric and optical complexity.

Existing feather modeling and rendering methods often struggle to balance realism, efficiency, and practicality. Some approaches rely on overly simplified geometric representations—approximating a feather as coarse curves or planar surfaces—and therefore fail to capture key appearance cues introduced by barbs and barbules [Baron et al. 2022; Franco and Walter 2002; Padrón-Griffe et al. 2024]. Other methods explicitly model individual fibers or adopt wave-optics interference models to reproduce structural colors observed in specific species [Huang et al. 2022; Yu et al. 2024]. Although these methods can achieve high physical accuracy, they are computationally expensive and typically require complex grooming procedures or specialized simulations, which limits scalability and practical use. Texture- or BTF-based techniques [Chen et al. 2002; Franco and Walter 2007] can produce convincing results under fixed lighting, but they often entangle illumination with material appearance, reducing generalization to novel viewpoints and lighting conditions.

In this paper, we present a practical and efficient framework for feather modeling and rendering. Instead of explicitly reconstructing every barb and barbule, we leverage the feather's inherent hierarchical structure to build a compact and scalable appearance representation. Given *a single image*, we automatically extract intrinsic properties, including the albedo, the central rachis curve, and the overall silhouette. We then parameterize the feather into a 2D UV domain and procedurally encode fine-scale details. This representation supports efficient point and area queries, enabling seamless multi-scale rendering: it preserves high-frequency details at close viewing distances while avoiding aliasing at far distances. We also design an analytical method to account for self-occlusion (masking) between barbs and barbules, ensuring stable energy behavior and consistent appearance under varying viewing and lighting directions. For appearance modeling, we develop a novel feather bidirectional curve scattering distribution function (BCSDF) for non-iridescent structurally colored feathers, such as parrot feathers. We introduce an additional sheen lobe to capture the characteristic edge glints produced by dense barbules at grazing angles, which is essential for reproducing the soft, fluffy appearance of real feathers. Combined with geometric-level masking and our area-query integration, the proposed BCSDF yields consistent multi-view appearance and faithfully captures multi-level visual effects. We further derive an analytic BCSDF formulation for the area-query case to ensure correct aggregated shading across scales. Our method enables real-time, high-fidelity feather rendering, and we demonstrate results on both entire birds and individual feathers. The rendered appearance closely matches real feather photographs, reproducing subtle details such as sheen and the smooth attenuation of fluffiness along feather edges. Our approach is also highly efficient: we achieve 52.8 FPS when rendering full parrots with high-fidelity appearance, while using only about 0.1% of the memory required by explicitly modeled feather geometry. In summary, our main contributions are:

- A hierarchical, procedural feather representation that supports efficient range queries and continuous level-of-detail rendering, while accurately preserving masking effects.
- A novel feather BCSDF that incorporates a sheen lobe and structural barb attributes, enabling faithful reproduction of fluffy appearance and highly saturated colors, validated against real photographs and spectral measurements.
- A single-image-based pipeline for feather modeling and rendering that automatically extracts structural and intrinsic appearance information and achieves real-time, realistic rendering.

2 Related Work

2.1 Feather Geometry and Appearance Modeling

Early feather rendering work focuses on explicit geometric hierarchies and image-based appearance models. Chen et al. [2002] introduced an L-system based feather model and an efficient BTF formulation to capture visible micro-structure. Streit and Heidrich [2002] proposed a biologically-parameterized model that supports intuitive control and multi-level detail.

Several methods use parametric curves to represent feather geometry. Franco and Walter [2002] modeled individual feathers using parameterized Bézier curves. To synthesize pigmentation patterns, Franco and Walter [2007] combined a parametric feather layout with procedural pattern generation. Haapaoja and Genzwürker [2019] presented a pipeline for generating and animating groomed feathers. Baron et al. [2021] proposed procedural shading to approximate feather micro-structure effects without explicitly modeling all details, and later extended this direction with microstructure-based appearance rendering [Baron et al. 2022]. Several works model iridescent feather appearance. Huang et al. [2022] proposed a feather-specific BSDF that statistically captures barbule-scale structures and supports importance sampling at render time. Yu et al. [2024] modeled iridescent feathers with diverse nanostructures for more natural variation. Padrón-Griffe et al. [2024] proposed a surface-based appearance model for pennaceous feathers that improves efficiency in the far field. In contrast, our work targets a practical single-image pipeline and a compact hierarchical representation that remains stable under varying viewpoints and illumination, while also addressing saturated and near-fluorescent feather colors.

2.2 Hair and Fur Rendering

Hair and fur rendering has been extensively studied, with a focus on modeling light scattering from individual fibers. Marschner et al. [2003] introduced a classical BCSDF that models multiple reflection and transmission paths within a cylindrical fiber, and has been widely adopted for human hair rendering. However, the model assumes a solid circular cross-section and primarily accounts for single-fiber scattering, which limits its applicability to light-colored and optically sparse animal fur. Yan et al. [2015] observed that many animal fur fibers contain an inner medulla that significantly affects internal scattering. They introduced a dual-cylinder fiber model that produces broader and softer reflectance lobes, improving the appearance of light-colored fur. Chiang et al. [2016] further compressed high-order scattering effects into an energy compensation term,

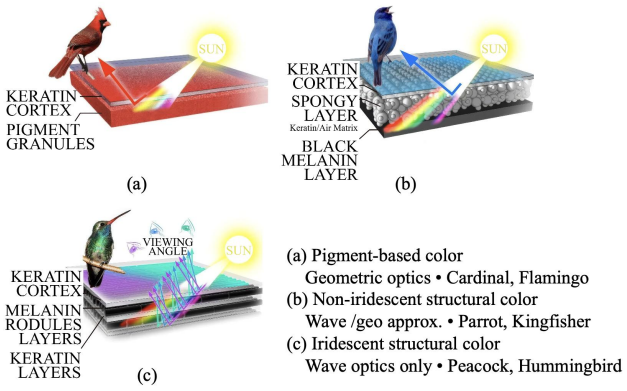

Fig. 2. Three types of feather coloration: pigment-based color, non-iridescent structural color and iridescent structural color. The illustration is adapted from Busch [2024].

achieving a practical balance between visual fidelity and performance. While these models capture many aspects of fiber scattering, they are designed for hair and fur and do not directly generalize to feather structures, which exhibit flattened barbs, dense barbules, and additional optical effects such as rim sheen and fluorescence. Our work builds upon insights from hair and fur scattering models, but introduces a feather-specific BCSDF that explicitly accounts for barbule structure, internal scattering, and saturated appearance.

2.3 Procedural Texture

Procedural texturing represents surface detail using analytic functions instead of explicit images, enabling compact storage, infinite detail, and constant-time random access at arbitrary spatial coordinates [Lagae et al. 2010; Perlin 1985]. Classical approaches such as noise-based and solid textures have been widely used to generate patterns like wood, marble, and terrain while supporting efficient point evaluation and analytic filtering [Lagae et al. 2009; Liu et al. 2016; Peachey 1985; Perlin 1985].

Several works extend procedural textures to support region-based queries and appearance aggregation, allowing filtered or MIP-mapped results to be computed in constant time and making them suitable for level-of-detail rendering [Wang et al. 2020]. More recently, neural procedural textures learn continuous mappings from spatial coordinates to appearance attributes, combining the flexibility of procedural models with data-driven expressiveness while retaining random-access evaluation [Henzler et al. 2020]. However, most procedural texture methods focus on spatial color or normal variation and treat reflectance separately from pattern generation [Henzler et al. 2020; Lagae et al. 2010]. In contrast, our approach encodes position-dependent reflectance directly in a procedural representation, enabling efficient point and region queries while supporting view-dependent appearance within a unified framework.

3 Background and Motivation

3.1 Feather Hierarchy Structure

Bird feathers have a clear multi-level structure: a central shaft (rachis) with rows of barbs on each side, and finer barbules branching from each barb [Lucas 1972] (Fig. 3). In flight feathers, adjacent barbules can interlock, keeping the vane surface continuous and smooth, providing the structural support needed for flight [Ennos et al. 1995; Lucas 1972]. By contrast, downy feathers typically do not form a tightly interlocked vane, which are usually hidden and not visible for birds. [Hendrickx-Rodriguez and Lentink 2025]. Their structure remains loose and fluffy, making them good for keeping warm.

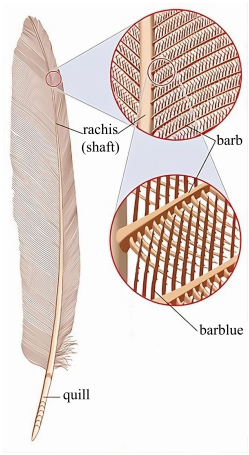

Fig. 3. Illustration of the multi-level structure of a typical flight feather. Hundreds of barbs grow on each side of a central shaft (rachis), and barbules branch from each barb. The feather diagram is adapted from Rafferty [2025].

This geometric complexity will strongly influence a feather's appearance. The overall shape (rachis and barbs) defines the silhouette and large-scale reflections, while the tiny interlocking barbules form a fine fringe at the edges, which catches light and gives the feather a subtle sheen. Modeling every barbule explicitly is often impractical (for a single feather, we need around 200MB to model the detailed hierarchical structure), so computer graphics employs a mix of explicit and implicit representations to balance realism and performance. For example, in real-time applications, it is common to use flat polygonal "feather cards" with alpha-mapped textures to suggest barbs and barbules. Some systems extend this with shell texturing, stacking semi-transparent textured layers to mimic feather volume and depth [Lengyel et al. 2001]. These impostor-based methods are fast but tend to look flat, missing depth cues like self-shadowing and parallax.

3.2 Feather Color Mechanisms (Pigmentary vs. Structural)

As shown in Fig. 2, feather coloration can be summarized as three primary sets: pigment-based color, non-iridescent structural color and iridescent structural color [Tinbergen et al. 2013].

Pigment-based color feather. Pigments such as *melanins* and *carotenoids* are natural chemicals inside feathers that absorb certain wavelengths of light. They produce built-in colors (brown, red, yellow, etc.) that are mostly diffuse and angle-independent, mainly varying with pigment concentration (similar to hair or fur). In rendering, pigmented feathers can be approximated with fur shading models introduced by Yan et al. [2017].

In contrast, structural colors are generated by the feather's nanostructure rather than pigments. These occur in two forms: *non-iridescent structural color* and *iridescent structural color*.

Non-iridescent structural color feather. Non-iridescent structural color results from incoherent scattering of light by microscopic air/keratin structures in the feather barb. Certain feather barbs

contain a spongy medulla that selectively reinforces shorter wavelengths by constructive scattering [Yin et al. 2012]. This produces brilliant colors (often blue or blue-green) with high saturation. Notably, many birds that appear blue (e.g., bluebirds, blue jays) have no blue pigment at all; the color comes purely from this nano-structure. The key characteristic of non-iridescent structural color is that it is **angle-independent**: the color looks the same from different viewing directions (hence "non-iridescent"). That is the reason why we can approximate this phenomenon under a geometry optical rendering framework, and in this paper we derive a new BCSDF model to approximate this phenomenon. However, these feathers usually have a deep melanin layer underlying the spongy structure, which absorbs light that penetrates through. As a result, the feather's front side shows a vivid color, while the back side appears dark or dull (Fig. 11).

Iridescent structural color feather. Iridescent structural color arises from highly ordered nanostructures that cause coherent interference effects. Here, the feather's barbules (or sometimes the barb cortex) contain multilayered thin films or photonic crystal-like arrays of melanin and keratin. When light waves reflect within these periodic structures, certain wavelengths constructively interfere while others cancel out, producing intense, narrow-band color that shifts with viewing angle [Hirayama et al. 2001]. In computer graphics, capturing iridescence requires a spectral, angle-dependent reflectance model. Prior work has modeled bird feather iridescence by simulating thin-film interference in the barbule structure [Yu et al. 2024], or by measuring and tabulating the feather's bidirectional reflectance and building a wave optics based shader.

3.3 Hair and Fur Scattering Models as a Foundation

Feathers share similarities with collections of narrow fibers (the barbs and barbules). Marschner et al. [2003] introduced a physically based BCSDF for hair fibers. In this model, each hair strand is treated as a translucent dielectric cylinder, with three dominant scattering modes (lobes): the R lobe, which is light reflecting off the surface of the fiber (producing the primary, typically white shine); the TT lobe, where light transmits through the fiber (entering and exiting, akin to forward scattering, often creating a translucent glow especially under backlight); and the TRT lobe, where light enters, reflects once inside, then exits (producing a secondary, often reddish-tinted highlight due to absorption by pigments in the fiber). This multi-lobe BCSDF captures characteristic hair effects like the color secondary highlight and soft backlit appearance that simpler models could not reproduce.

Building on this, Yan et al. [2017] extended the hair model to account for structural differences in animal fur. Unlike human hair, many animal hairs have a significant medulla (an inner core of air-filled or scattering material) which behaves like the pigment-based color feather. Yan's model adds two additional lobes to represent light that scatters within the medulla: a transmitted-scattering lobe (TT^s) for light that goes through the fiber and scatters internally, and a reflected-scattering lobe (TRT^s) for light that reflects internally and then scatters out through the medulla. These extra paths explain the more diffuse, desaturated look of fur that the original Marschner model could not fully capture.

Table 1. Parameters Definition for feather geometry and UV-to-canonical mapping.

Symbol	Definition
r, b, bl	Notations of rachis, barb and barbule
bd, bp	Distal/proximal barbules (bl if undistinguished)
$\mathcal{P}$	Feather contour polyline
F_P	Feather rachis centerline
N_b	Barb count along the rachis
d_{bl}	Barbule spacing parameter along the barb
α_b	Barb orientation angle
α_{bl}	Barbule angle offset w.r.t. the barb
r_r, r_b, r_{bl}	Radius of rachis, barb and barbule
$\boldsymbol{n}_b$	Barb normal $(\cos\alpha_b, \sin\alpha_b)$
$\boldsymbol{v}_b$	Barb direction $(-\sin(\alpha_b), \cos(\alpha_b))$
$\boldsymbol{n}_{bl}$	Barbule normal (includes $\boldsymbol{n}_{bd}$ and $\boldsymbol{n}_{bp}$)
$\boldsymbol{n}_{bd}$	Distal barbule normal $(\cos(\alpha_b - \alpha_{bl}), \sin(\alpha_b - \alpha_{bl}))$
$\boldsymbol{n}_{bp}$	Proximal barbule normal $(\cos(\alpha_b + \alpha_{bl}), \sin(\alpha_b + \alpha_{bl}))$
p_b	Barb period (gap), $p_b = \sin(\alpha_b)/N_b$
p_{bl}	Barbule period (gap), $p_{bl} = d_{bl}\sin(\alpha_{bl})$
$\mathcal{U}$	UV space on the feather surface
$\mathcal{C}$	Canonical space
M	UV-to-canonical map: $(x, y) = (u - F_P(v) + 0.5,\ v)$

Motivated by this background, we aim to address the geometric and appearance complexity of feathers by enabling fast yet high-fidelity appearance queries without explicitly modeling the full multi-level hierarchy. In particular, we seek a compact, physically motivated representation that preserves the dominant visual cues while remaining efficient for rendering. In this paper we present two sets of results: pigment-color feathers and non-iridescent structurally color feathers. For pigment coloration, we directly use the hair model of Yan et al. [2017], but with modulated refractive index to approximate eccentricity for the flat, high-aspect-ratio fiber geometry of feather barbs and barbules (much flatter than human hair). For non-iridescent structural coloration, we introduce a new BCSDF model in Sec. 6.1 by modifying the medulla representation: the front side (i.e., the feather's outward-facing direction) is modeled as a specular reflective layer, while the medulla itself is treated as a fully black absorbing layer. Fig. 5 shows our complete pipeline.

4 Geometric Representation

As discussed in Sec. 3.1, feather microstructures are complex and difficult to model explicitly, yet their geometric organization is highly regular: barbs are approximately parallel, and barbules are densely packed and consistently aligned within each barb. This regularity enables procedural and analytic approximations, whereas explicit micro-geometry incurs high costs for intersections, visibility, and shading. We therefore propose an implicit procedural feather representation that encodes a complete feather using a single UV-mapped rectangle, while supporting efficient point and range queries, continuous level-of-detail rendering, and accurate shadowing and masking effects. As defined in Table 1 and Fig. 6, we summarize key parameters used throughout this section. Most parameters are adopted with

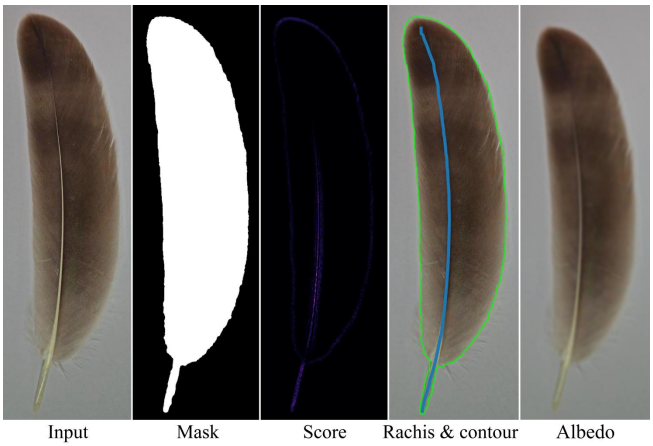

Fig. 4. Illustration of our feather parameters extraction. Given a feather capture as input, we first obtain a mask according to the color and take its boundary as the contour. To extract the rachis, we build a per-pixel score map and find a line path formed by the most possible pixels. The extracted rachis and contour are shown in blue and green. The albedo is obtained through performing a Gaussian blur on the input.

tuning from literature (e.g., Baron and Patterson [2019] and Broggi et al. [2011]), except for the albedo, contour and rachis automatically extracted from pictures. Since a feather is split into left and right sides by its central rachis, which can be defined by the same kind of parameters, we only focus on one side for the feather geometry.

4.1 Image Cue Extraction

If we already have a feather modeled by the artist, we can directly obtain its albedo, rachis and contour. Otherwise, we extract them from a single captured feather image as in Fig 4. The image should contain a single, fully visible feather, not partially occluded feathers in whole-bird images. Specifically, we recover (i) a clean albedo map, (ii) mask and outer contour, and (iii) a smooth rachis centerline, which together provide stable inputs for canonical mapping and downstream queries.

Our extraction follows a three-stage process. First, we preprocess the image and segment the feather region to obtain a binary mask. Second, we recover the outer contour from the mask and resample it uniformly in arc length to obtain an ordered contour representation. Third, we estimate the rachis by constructing a cue-fused centerline score map inside the mask and solving for a mask-constrained minimum-cost path, followed by lightweight refinement and smoothing. This formulation avoids dataset-level landmarking while remaining robust to noise and occlusion.

Albedo. Since our representation does not explicitly reconstruct barb and barbule geometry, fine-scale shadows and streaks induced by the micro-structure would otherwise contaminate color estimation in the input image. To obtain a base appearance that remains stable across levels of detail, we define albedo as a low-frequency color field decoupled from the micro-structure, and extract it using a Gaussian low-pass filter. The resulting albedo preserves the feather's large-scale color variation while substantially suppressing high-frequency fluctuations caused by the micro-structure.

Mask and contour. We segment the input image to obtain a clean binary silhouette mask of the feather. This mask constrains rachis estimation and all subsequent procedural steps to valid feather pixels. We first normalize the input resolution and apply an edge-preserving denoiser to reduce texture noise while preserving boundaries. To improve feather background separation, we enhance local contrast with luminance-adaptive histogram equalization in a perceptual color space. We then segment the feather region in two steps: an HSV color threshold provides a coarse mask, and an Otsu intensity threshold [Otsu et al. 1975] refines the boundary. Then we clean the mask with morphological opening/closing and remove small connected components to obtain a single, connected silhouette. From the silhouette mask, we extract the largest external contour using the border-following method of Suzuki and Abe [1985], yielding an ordered boundary polyline. We resample this contour uniformly in arc length to obtain a fixed-length polyline. The silhouette mask defines the valid domain for rachis centerline search and downstream queries, while the resampled polyline provides a stable boundary representation for normalization and for clamping the procedurally generated structure to the feather silhouette.

Rachis estimation. We estimate the rachis as a smooth centerline inside the feather mask, driven by a dense *score map*. We observed that rachis has the following three features. First, the rachis forms a contrast with the vane. Second, its direction roughly aligns with the feather main direction. Third, it is not located at the boundary of a feather, and is usually the centerline of a feather. Considering these features, we build a score map for all the positions x of the image based on three cues: a vesselness response $v(x)$ by Frangi filtering [Frangi et al. 1998], an alignment term $a(x) = t(x) \cdot t_{main}$ that favors directions consistent with the feather's main direction, and a distance $d(x)$ from the mask to the current point. We multiply these cues and normalize:

$$S(x) = v(x)\, a(x)\, d(x), \tag{1}$$

where $S(x)$ is the per-pixel likelihood of the rachis.

To obtain a globally consistent centerline, we convert the score map to a cost image and solve for a minimum-cost path between the two feather endpoints, constrained to stay inside the mask. This produces a robust initial rachis even under noise and partial occlusion. We then resample the path uniformly in arc length and refine it with a few iterations. At each sample, we search only along the local normal direction to maximize a local objective based on the score, with regularization that limits large normal offsets and large changes between neighboring samples. We estimate sub-pixel updates with a simple quadratic fit (e.g., a three-point parabolic fit). After each iteration, we smooth the path along the curve to remove high-frequency jitter. This alternating local fitting and global smoothing balances image evidence with geometric regularity.

If the refined path does not exactly reach the mask endpoints, we complete it with short mask-constrained connecting segments to ensure connectivity. Finally, we remove duplicates, order the samples consistently, and fit a smooth parametric spline F_p as rachis.

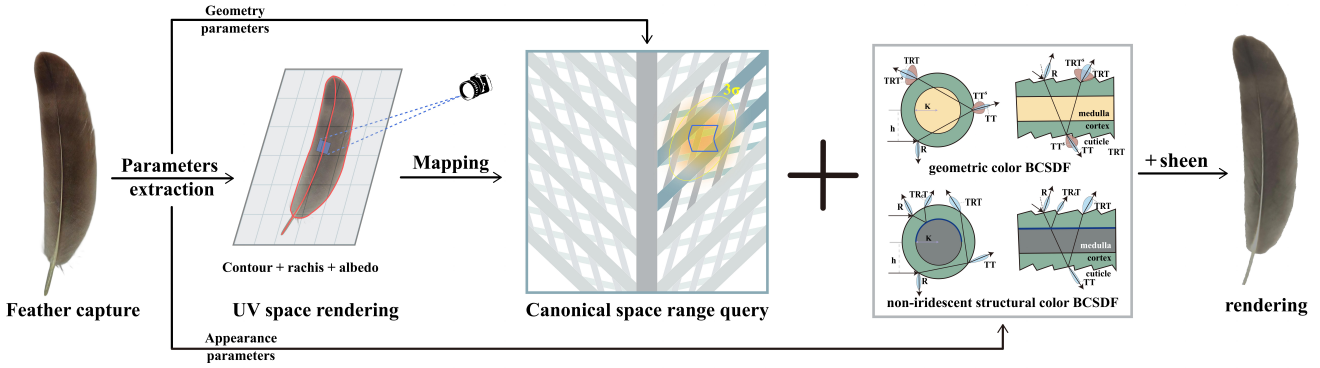

Fig. 5. The complete pipeline of our method. We first extract and match the parameters from an input feather capture, including contour, rachis and albedo in the UV space, geometry parameters in the canonical space and appearance parameters for the BCSDF. With these parameters determined, to render a feather defined on a surface, we take the pixel footprint in the UV space as a Gaussian and map it to the canonical space, where we can perform range query efficiently with simple parameters. The range query returns weight of each feather component, and we evaluate our appearance with the weights using suitable feather BCSDFs (Ours or Yan et al. [Yan et al. 2017]) with sheen term and we finally obtain the rendering result.

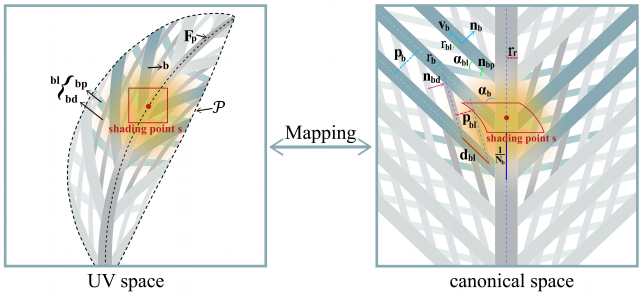

Fig. 6. Our geometry parameters and space mapping for range query. In the UV space, we define a rachis curve F_P and a closed polyline contour $\mathcal{P}$ to determine the shape of a feather. In the canonical space, we define the parameters of rachis r, barb b, and barbule bl (distal barbule bd and proximal barbule bp), which are listed in the Table 1. For range query, given a shading point and its pixel footprint range in the UV space, we map them to the canonical space according to the rachis curve.

Through the extraction method proposed above, we can obtain the albedo, rachis and contour from a captured image within a few seconds in practice.

4.2 Procedural Geometry Representation

Given the extracted contour and rachis from the previous section, we now define our procedural geometry representation. Since the rachis and contour can be arbitrary in the UV space, and thus the barbs and barbules are irregular, making them difficult to query for efficient rendering, the key idea is to avoid working directly in UV space $\mathcal{U}$. Instead, we introduce a canonical space C where all the rachis, barbs and barbules are regular and straight, among which the rachis is the centerline of the space and the barbs and barbules are extended from the rachis. Thus, we can define some parameters in this space to represent feather geometry.

Inputs in UV space. In $\mathcal{U}$, we represent the feather contour as a closed polyline $\mathcal{P} = \{\mathbf{v}_i\}_{i=0}^{N-1}$, and the rachis centerline as a curve fitted as a 1D function $u = F_P(v)$. These compact representations provide the constraints needed for mapping and clamping.

Canonical mapping. We define a mapping $M : \mathcal{U} \to C$ such that the rachis becomes straight in C. For a UV point $\mathbf{P}_\mathcal{U} = (u, v)$, we define its canonical coordinate $\mathbf{P}_C = (x, y)$ as

$$x = u - F_P(v) + 0.5, \qquad y = v. \tag{2}$$

This transform is a translation along the u-axis at each v; it introduces no rotation and no non-linear warping. The range of (u, v) and (x, y) are all from 0 to 1. We use the value of x to determine which side this point belongs to, and 0.5 corresponds to the rachis in the canonical representation.

All subsequent geometry construction and appearance queries are performed in C.

Procedural primitives in canonical space. In C, we approximate the rachis, barbs, and barbules with straight line primitives. Barb attachment points are uniformly spaced along the straightened rachis with distance $1/N_b$, and barb orientation is controlled by α_b. For each barb, we place two sets of straight barbules on the two sides of the barb, using spacing parameter d_{bl} and angle offset α_{bl} (equivalently, period $p_{bl} = d_{bl} \sin(\alpha_{bl})$ in the normal direction). We model distal and proximal barbules separately to capture their layered masking behavior when viewed from the front.

Mapping back and clamping. To place the procedural structure back in $\mathcal{U}$, we invert the translation: for $\mathbf{P}_C = (x, y)$ we set $v = y$ and $u = x + \tilde{\mathbf{r}}(v)$. We then clamp the mapped structure by the silhouette polyline $\mathcal{P}$ to restrict it to the true vane region. This representation is compact and stable, while preserving the dominant structural regularities needed for efficient queries, continuous LOD, and masking effects.

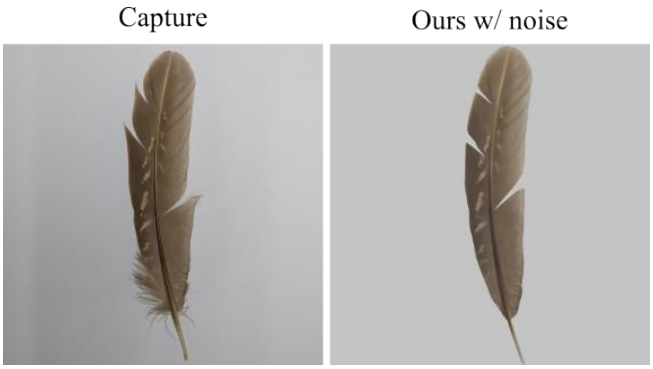

Fig. 7. With our noise-driven contour perturbation, we can match the imperfection on the barb's level of a real feather well.

4.3 Noise-driven Perturbation for Contour

Based on the definition above, we now have a perfect geometry. However, real feather may have some random gaps on its contour. To introduce such imperfection, we make some gaps for the feather contour through a noise control, as in Fig. 7. The gaps are generated by such a process: first we choose some position to make these gaps, and then we determine the gap widths. Finally, we define which direction and how far the gap extends. To achieve this, given a closed contour $\mathcal{P}$, we uniformly sample hundreds of points $\mathbf{v}_i{}_{i=0}^{N_v-1}$ on it, and N_v is the point number. Then, we sample a 1D noise field $\mathcal{F}(\tau)$ to distribute a weight to each point:

$$\lambda_{v_i} = \mathcal{F}(i), \tag{3}$$

in practice we use a Perlin noise. According to the weight λ_{v_i}, we choose vertices with the top-K values whose indices are $I = i_k$ as the centers of gaps. We use a Gaussian with standard deviation σ_k to determine the width of each gap, which perturbs neighbour points around the gap center v_{i_k} by

$$\begin{aligned} d_{i_k} &= \min(|i - i_k|,\, N_v - |i - i_k|), \\ \epsilon_{i_k} &= \exp\Big(-\frac{d_{i_k}^2}{2\sigma_k^2}\Big), \end{aligned} \tag{4}$$

where d_{i_k} is the distance between v_i and v_{i_k}, and ϵ_{i_k} is the corresponding Gaussian weight. We perturb a point when its ϵ_{i_k} is greater than $1e-3$. Therefore we determine the gap width, and the points inside the gap are mapped to the canonical space to get the corresponding barb directions, which are then mapped back to obtain a perturb direction $\hat{\mathbf{v}}$. To compute the distance of the gap points extend, we reuse the Gaussian weight ϵ_{i_k} with an amplitude A_k. Thus, the perturbed new point is

$$v_i' = v_i + A_k \epsilon_{i_k} \hat{\mathbf{v}}. \tag{5}$$

5 Geometric Query

To render a feather, we need to know the exact position of a point in UV space. Therefore, we present a querying approach for capturing feather microstructure across scales. First, we define a point query in a canonical feather coordinate system that precisely classifies any surface point by its distance to the different feather structure, clarifying our UV-to-canonical mapping and establishing the foundation for the range-query formulations that follow. Building on this, we introduce range queries for pixel footprints with adaptive anti-aliasing. For near-field footprints, an analytic *spatial*-domain integration computes the soft coverage of barbs and barbules under a Gaussian footprint, eliminating flicker and aliasing. For far-field footprints covering many barbules at once, a *frequency*-domain formulation efficiently aggregates the periodic contributions of barbule groups. Together, these query strategies ensure accurate and stable feather appearance across all viewing distances. Considering the symmetry of a feather, we only take the left part of a feather as an example.

5.1 Point Query

Given our canonical-space representation, a point query reduces to evaluating distances to the analytic primitives (rachis, barbs, and barbules) in C. For a query point, if its distance to the closest feather component is smaller than the corresponding radius, we think it is on that component and then obtain a distance to the component centerline h [Marschner et al. 2003] for BCSDF. We first map a query point from UV space to canonical space so that the rachis becomes the vertical line $x = 0.5$.

Rachis. For $\boldsymbol{P} = (x, y)$, the distance to the rachis is

$$d_{rachis} = |x - 0.5|. \tag{6}$$

Barbs. We compute barb distances in a local barb frame. For the left vane, the barb normal and direction are $\boldsymbol{n}_b = (\cos\alpha_b, \sin\alpha_b)$ and $\boldsymbol{v}_b = (-\sin\alpha_b, \cos\alpha_b)$. We project the point as $\boldsymbol{P}_b = (x_b, y_b) = (\boldsymbol{P}\cdot\boldsymbol{v}_b, \boldsymbol{P}\cdot\boldsymbol{n}_b)$. With barb count along the rachis N_b, the distance to the nearest barb centerline is

$$d_{barb} = \left| y_b - \frac{\lfloor y_b N_b + 0.5 \rfloor}{N_b} \right|. \tag{7}$$

The two neighboring barbs that bracket the point are located at $y_{barb}^{below} = \lfloor y_b N_b \rfloor / N_b$ and $y_{barb}^{above} = \lceil y_b N_b \rceil / N_b$, which we use to evaluate the two barbule groups attached to these barbs.

Barbules. We define the barbule frame by its normal $\boldsymbol{n}_{bl}$ and direction $\boldsymbol{v}_{bl}$, and project the point as $\boldsymbol{P}_{bl} = (x_{bl}, y_{bl}) = (\boldsymbol{P}\cdot\boldsymbol{v}_{bl}, \boldsymbol{P}\cdot\boldsymbol{n}_{bl})$. For the barb at position y_{barb}, its attached barbule group introduces a phase shift $\delta^{bl} = (0, y_{barb})\cdot\boldsymbol{n}_{bl}$. We offset the coordinate by $y' = y_{bl} - \delta^{bl}$. The barbule period is $p_{bl} = d_{bl}\sin(\alpha_{bl})$, and the distance to the nearest barbule is

$$d_{barbule} = \left| y' - \lfloor \frac{y'}{p_{bl}} + 0.5 \rfloor p_{bl} \right|. \tag{8}$$

We evaluate $d_{barbule}$ for both neighboring barbs (below and above). When viewed from the front, distal barbules occlude proximal barbules, so we test the distal group first.

5.2 Spatial-Domain Range Query

Feather micro-structures (barbs and barbules) are often far below pixel scale, so explicit geometric intersection or stochastic sampling can introduce severe aliasing and noise, and tends to flicker under animation. We therefore seek a deterministic range query that estimates, for each pixel footprint, the expected coverage of four feather components: rachis w_r, barbs w_b, distal barbules w_{bd}, and

proximal barbules w_{bp} which enable stable anti-aliasing without Monte Carlo sampling. As the pixel footprint may have an arbitrary shape which is unsuitable for query, we model the pixel footprint as an anisotropic 2D Gaussian kernel $G(\boldsymbol{x}; \mu, \Sigma)$ with $\boldsymbol{x} \in \mathbb{R}^2$ in UV coordinates, mean $\mu \in \mathbb{R}^2$, and symmetric positive semidefinite covariance $\Sigma \in \mathbb{R}^{2\times 2}$. This Gaussian can be simply mapped to our canonical space according to the rachis, and we compute coverage by integrating this kernel against strip-like indicator functions derived from our procedural feather geometry.

For efficient integral calculation, *all spatial-domain queries are performed in a local frame aligned with the current barb.* Given the unit barb normal $\mathbf{n}_b$ and unit barb direction $\mathbf{v}_b$ (with $\mathbf{n}_b \perp \mathbf{v}_b$), we define

$$M = \begin{pmatrix} \boldsymbol{n}_b^\top \\ \boldsymbol{v}_b^\top \end{pmatrix}, \quad \boldsymbol{z} = \begin{pmatrix} s \\ t \end{pmatrix} = M\boldsymbol{x}, \tag{9}$$

so that s measures displacement across barbs (along $\mathbf{n}_b$) and t follows the barb direction (along $\mathbf{v}_b$). Under this orthonormal change of coordinates,

$$\mu_z = M\mu = \begin{pmatrix} \mu_s \\ \mu_t \end{pmatrix}, \qquad \Sigma_z = M\Sigma M^\top = \begin{pmatrix} \sigma_s^2 & \sigma_{st} \\ \sigma_{st} & \sigma_t^2 \end{pmatrix}, \tag{10}$$

where

$$\sigma_s^2 = \boldsymbol{n}_b^\top \Sigma \boldsymbol{n}_b, \quad \sigma_t^2 = \boldsymbol{v}_b^\top \Sigma \boldsymbol{v}_b, \quad \sigma_{st} = \boldsymbol{n}_b^\top \Sigma \boldsymbol{v}_b. \tag{11}$$

Barb contribution. We model barbs as an infinite periodic stripe family in the s direction with period p_b and strip half-width r_b. Let the strip centers be $c_{k_b} = k_b p_b$ for $k_b \in \mathbb{Z}$ (we translate the rachis so that the reference barb is centered at $s = 0$). We denote by $\mathcal{G}[q_1, q_2; \mu, \sigma] = \Phi\left(\frac{q_2-\mu}{\sigma}\right) - \Phi\left(\frac{q_1-\mu}{\sigma}\right)$ the probability mass of Gaussian $\mathcal{N}(\mu, \sigma^2)$ over the interval $[q_1, q_2]$, where $\Phi(\cdot)$ is the standard normal CDF. The expected coverage of the barb family under the Gaussian footprint is

$$w_b = \sum_{k_b \in \mathbb{Z}} \int_{s_o}^{s_e} G_s(s)ds = \sum_{k_b \in \mathbb{Z}} \mathcal{G}[s_o, s_e; \mu_s, \sigma_s], \tag{12}$$

where $s_o = c_{k_b} - r_b$, $s_e = c_{k_b} + r_b$. In practice, we truncate the sum to stripes within a few standard deviations of the footprint center by $|c_{k_b} - \mu_s| \le m\sigma_s$ with $m = 4$.

Barbule contribution. A single barbule segment inside a barb gap is modeled as a tilted strip of half-width r_{bl} with unit normal $\mathbf{n}_{bl}$ and phase ϕ, where ϕ is determined by the barbule index within its group. In the barb-aligned frame (s, t), the signed distance to the barbule axis can be written as

$$\boldsymbol{n}_{bl} \cdot \boldsymbol{x} = a\,s + b\,t, \quad a = \boldsymbol{n}_{bl} \cdot \boldsymbol{n}_b, \; b = \boldsymbol{n}_{bl} \cdot \boldsymbol{v}_b. \tag{13}$$

The barbule occupies the region $|\mathbf{n}_{bl} \cdot \boldsymbol{x} - \phi| \le r_{bl}$ and is truncated to a finite interval $s \in [s_l, s_r]$ by the two adjacent barbs. For a fixed s, this condition induces an interval along t,

$$t \in [f_1(s), f_2(s)], \; f_{1,2}(s) = \mathrm{sort}\left(\frac{\phi - as - r_{bl}}{b}, \frac{\phi - as + r_{bl}}{b}\right), \tag{14}$$

where $\mathrm{sort}(\cdot,\cdot)$ returns ordered endpoints and handles the case $b < 0$ automatically. Under the Gaussian footprint, the exact contribution of this barbule reduces to a one-dimensional integral along s,

$$I_{barbule} = \int_{s_l}^{s_r} G_s(s)\mathcal{G}[f_1(s), f_2(s); \mu_{t|s}(s), \sigma_{t|s}]ds, \tag{15}$$

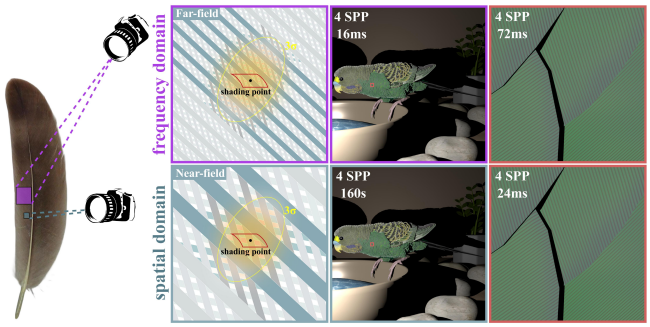

Fig. 8. Range query is performed in frequency or spatial domain during rendering. For far-field rendering when a pixel footprint covers at least four barbs, we choose frequency domain, otherwise we use spatial domain. The range query is conducted with a Gaussian representing a pixel footprint mapped to the canonical space, and the rendering results show that switching the domain can keep the quality while achieving good performance.

where $G_s(s) = \mathcal{N}(s; \mu_s, \sigma_s^2)$ is the marginal of the Gaussian along s, and $\mu_{t|s}(s)$, $\sigma_{t|s}$ are the conditional Gaussian parameters in the t direction.

A distal or proximal barbule group consists of a set of parallel barbules with indices $k_{bl} \in \mathbb{Z}$ and corresponding phases $\phi_{k_{bl}}$. The total contribution of the group within a barb gap $s \in [s_l, s_r]$ is the sum of individual barbule contributions,

$$w_{bl} = \sum_{k_{bl} \in \mathbb{Z}} I_{barbule}(\phi_{k_{bl}}), \tag{16}$$

where each term is evaluated using Eq. (15). In practice, the infinite sum is truncated to barbules whose centers lie within a few standard deviations of the Gaussian footprint. A convenient center index k_{bl}^0 (whose barbule axis is closest to the Gaussian mean) is estimated from the approximate alignment condition,

$$k_{bl}^0 \approx \frac{b\mu_t + a\mu_s - \phi_{base}}{p_{bl}}, \tag{17}$$

where p_{bl} denotes the barbule period of the group and ϕ_{base} is the phase of a reference barbule. We then enumerate a symmetric index range $k_{bl} \in [k_{bl}^0 - \Delta k_{bl}, k_{bl}^0 + \Delta k_{bl}]$, with Δk_{bl} chosen to cover the required physical support of the footprint. For disoccluded distal barbules, weight $w_{bd} = w_{bl}$.

Since distal barbules occlude proximal ones, the effective proximal contribution is obtained by subtracting the overlapped measure from the proximal group sum. For a proximal barbule index k_{bp} and a distal index k_{bd}, let $[f_{1,k}^{prox}(s), f_{2,k}^{prox}(s)]$ and $[f_{1,k}^{dist}(s), f_{2,k}^{dist}(s)]$ denote their respective t-intervals induced at a fixed s. The pairwise overlap is then:

$$O_{k_{bp},k_{bd}} = \int_{s_l}^{s_r} G_s(s)\mathcal{G}\left[f_{1,k}^{\max}, f_{2,k}^{\min}; \mu_{t|s}(s), \sigma_{t|s}\right]_+ ds, \tag{18}$$

where $f_{2,k}^{\min} = \min\{f_{2,k_{bp}}^{\mathrm{prox}}(s), f_{2,k_{bd}}^{\mathrm{dist}}(s)\}$ and $f_{1,k}^{\max} = \max\{f_{1,k_{bp}}^{\mathrm{prox}}(s), f_{1,k_{bd}}^{\mathrm{dist}}(s)\}$ select the upper and lower bounds of the intersection between the proximal and distal t-intervals at a fixed s, and $[\cdot]_+$ clamps non-overlapping cases to zero. The effective proximal group

contribution is therefore:

$$w_{bp} = \sum_{k_{bp}} I_{barbule(\phi_{k_{bp}})} - \sum\sum O_{k_{bp},k_{bd}}. \tag{19}$$

All integrals in Eq. (15) and Eq. (18) are evaluated using Gauss-Legendre quadrature.

5.3 Frequency-Domain Range Query

When observed from the far field, a single Gaussian footprint may overlap with a large number of barbules, making direct spatial-domain integration computationally expensive. To address this issue, we evaluate the integrals of an anisotropic Gaussian over the barbule geometry directly in the frequency domain. Exploiting the inherent periodic repetition of the feather structure, we model the barbs, distal barbules, and proximal barbules as three stripe families, denoted by b, bd, and bp, respectively.

In practice, we use a simple scale-based switch. For near-field footprints that are narrow in the cross-barb direction, direct spatial-domain evaluation is cheaper since the footprint overlaps only a few structures. Based on empirical timing measurements, the frequency-domain query becomes consistently more efficient once the cross-barb Gaussian width exceeds $\sigma_s \gtrsim 2p_b$, and we use this threshold to select between the two formulations.

As shown in Fig. 8, taking a feather's left vane as an example, each strip family with half-widths (r_b, r_{bl}) is characterized by a unit normal direction ($\mathbf{n}_b = (\cos(\alpha_b), \sin(\alpha_b))$, $\mathbf{n}_{bd} = (\cos(\alpha_b - \alpha_{bl}), \sin(\alpha_b - \alpha_{bl}))$, $\mathbf{n}_{bp} = (\cos(\alpha_b + \alpha_{bl}), \sin(\alpha_b + \alpha_{bl}))$) and a fundamental period ($p_b = \sin(\alpha_b)/N_b$, $p_{bl} = d_{bl}sin(\alpha_{bl})$) along that direction. Rather than introducing a full coordinate system, we use scalar projections

$$q = \mathbf{n}_b \cdot \boldsymbol{x}, \qquad g = \mathbf{n}_{bd} \cdot \boldsymbol{x}, \tag{20}$$

which separate the two distinct structural roles present in the geometry: periodicity across stripes and finite spatial extent along the parent b-stripes.

Along the parent direction q, subordinate stripe families (bd and bp) occupy a finite window of length L_Q within each b-interval. The window is positioned by an offset q_0 inside the interval. The placement of the stripe pattern itself is fixed by a base phase ϕ_{bd_0}(and analogously ϕ_{bp_0}), which defines a reference alignment within a chosen interval. Because the stripe directions $\mathbf{n}_b$ and $\mathbf{n}_{bd}$ are generally non-orthogonal, the stripe pattern undergoes a systematic lateral shift when moving from one b-interval to the next. This geometric effect is captured by the inter-interval phase shift δ_g^{bd} (and δ_g^{bp} for bp), which is fully determined by the stripe directions and the period p_b.

The combination of stripe periodicity along s and windowed repetition along q induces a discrete lattice of harmonic frequencies in the frequency domain. This lattice is indexed by a pair of integers (k_g, k_q), reflecting the two independent sources of repetition in the geometry. The index k_g enumerates harmonics of the stripe periodicity along g, while k_q indexes repetitions of the windowed stripe pattern across successive b-intervals along q. The index k_q does not introduce an additional geometric degree of freedom; it arises solely from the discrete repetition of the finite stripe window.

Each lattice node corresponds to a world-space frequency vector. For the bd-stripe family, this vector is

$$\boldsymbol{\omega}_{bd} = \omega_{k_g}^{bd}\, \mathbf{n}_{bd} + \omega_{k_q}^{bd}\, \mathbf{n}_b, \tag{21}$$

with

$$\omega_{k_g}^{bd} = \frac{2\pi k_g}{p_{bl}}, \qquad \omega_{k_q}^{bd} = \frac{2\pi k_q - \omega_{k_g}^{bd}\, \delta_g^{bd}}{p_b}. \tag{22}$$

An analogous construction applies to the bp-stripe family.

Gaussian strip integrals. To evaluate the contribution of a stripe family, we integrate the Gaussian against its indicator function. Using Plancherel's identity, the integral against the full bd-stripe family can be written as a sum over its discrete frequency lattice,

$$w_{bd} = \sum_{k_g,k_q} \widehat{G}(\boldsymbol{\omega}_{bd})\, \overline{bp_{bl}(k_g, k_q)}, \tag{23}$$

where $\widehat{G}$ denotes the Fourier transform of the Gaussian and $bp_{bl}(k_g, k_q)$ are geometry-dependent coefficients determined by stripe width, window length, and phase placement. Substituting the analytic Gaussian spectrum $\widehat{G}(\omega) = \exp(-i\,\omega \cdot \mu)\exp(-\frac{1}{2}\omega^\top \Sigma \omega)$ and the closed-form stripe coefficient bp_{bl} (a product of sinc envelopes and a phase term) into Eq. (23), and taking the real part, gives

$$w_{bd} = \sum_{k_g,k_q} \frac{2r_{bl}L_Q}{p_{bl}p_b}\, \mathrm{sinc}\left(\frac{\omega_{k_g}^{bd} 2r_{bl}}{2}\right) \mathrm{sinc}\left(\frac{\omega_{k_q}^{bd} L_Q}{2}\right) \cdot \exp\Big(-\tfrac{1}{2}\boldsymbol{\omega}_{bd}^\top \Sigma\, \boldsymbol{\omega}_{bd}\Big)\, \cos\Big(\omega_{k_g}^{bd}\phi_{bd_0} + \omega_{k_q}^{bd} q_0 - \boldsymbol{\omega}_{bd}\cdot\mu\Big). \tag{24}$$

The same structure applies to w_{bp} after replacing the corresponding stripe parameters. Following the same way, we can compute w'_{bp} by replacing the parameters of bp with bd.

Overlap between stripe families. To account for occlusion of bp by bd, we consider the integral of the Gaussian over the intersection of the two stripe families,

$$I_{overlap} = \iint G(x)\, \mathbf{1}_{bd}(x)\, \mathbf{1}_{bp}(x)\, dx. \tag{25}$$

In the frequency domain, the product of indicator functions becomes a convolution between their spectra. Since both $\widehat{\mathbf{1}_{bd}}$ and $\widehat{\mathbf{1}_{bp}}$ consist of discrete impulse trains supported on their respective harmonic lattices, the convolution reduces to pairwise interactions between lattice nodes, resulting in the double harmonic sum. Therefore, we have $w_{bp} = w'_{bp} - I_{overlap}$.

5.4 Masking

In grazing-angle views, masking changes the relative visibility of different microstructures, which in turn affects detail loss, perceived translucency, and the appearance and view-dependent color mixing of differently colored barbs; therefore, we model and quantify this masking effect as in Fig. 9. This figure illustrates a masking scenario where a barb occludes its own barbules when viewed from a grazing angle. Such occlusion alters the effective projected shape and visibility of the feather microstructures.

A feather fiber's (rachis, barb and barbule) cross-section is roughly circular or elliptical. When viewed head-on, it covers a width of approximately $2r$ (its diameter) in the feather's UV space. However,

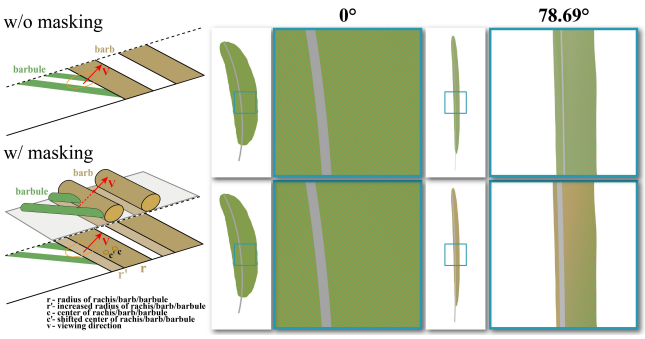

Fig. 9. Masking effect within our range query. Left we show the reason why we should consider masking. Barbules can be occluded by barbs for a 3D feather, and thus we should compute its effect with parameters given at the bottom left. Right we show some renderings with or without masking, it is obvious that yellow barbs occlude most green barbules viewing from a inclination angle.

when observed at a grazing angle, its projection widens due to foreshortening, potentially masking structures like barbules behind it.

Since rachis, barb and barbule share the same rule to form the masking effect, we take the barb as an example to illustrate our method without loss of generality. We quantify the masking by computing an effective increase in the projected width W' of the barb in UV space due to the view direction $\boldsymbol{v}$.

Let the actual barb radius be r, and its bitangent axis (perpendicular to the barb direction) be the unit vector $\boldsymbol{r}$. Then, $\boldsymbol{v_{proj}} \cdot \boldsymbol{r}$ is the cosine of the angle between the view direction (projected onto the feather surface) and the barb bitangent. The apparent elliptical cross-section width (major axis) of the barb is

$$W_{\text{proj}} = \frac{2r}{\left|\mathbf{v}_{\text{proj}} \cdot \mathbf{r}\right| + \varepsilon}. \tag{26}$$

To refine the masking effect, we further compute the additional projected half-width W' caused by the barb masking the view. Let $\boldsymbol{n}$ be the local normal to the feather surface. Then, the angle between the view direction and the surface normal is $\theta = \arccos(\boldsymbol{v} \cdot \boldsymbol{n})$, and the increased projected width is:

$$W' = \sqrt{\left(\frac{w_{proj}}{2}\right)^2 + (r \cdot \tan(\theta))^2} - \frac{w_{proj}}{2}, \tag{27}$$

and this additional width should be no larger than the barb gap $W' = min(W', \frac{d_{barb}}{\boldsymbol{v_{proj}} \cdot \boldsymbol{r}} - W_{proj})$. Given w' and the view-aligned projected width w, we define an updated effective barb radius in UV space as:

$$r' = \frac{(W_{proj} + W') \cdot (\boldsymbol{v_{proj}} \cdot \boldsymbol{r})}{2}, \tag{28}$$

and update the centerline position to:

$$\boldsymbol{c}' = \boldsymbol{c} + \frac{W_{proj}}{2} \cdot \boldsymbol{r}, \tag{29}$$

which shifts the barb center slightly along its axis to reflect the asymmetric expansion. We apply this change of width and center to all the feather parts for the masking effect. Thanks to the consideration of barbule overlap during our range query, no additional step is required to accomplish the masking.

6 Appearance Representation

In this section, we will introduce how we model and render the feather appearance. We first define a new per-fiber scattering BCSDF model (Sec. 6.1). We extend existing fur fiber scattering models to account for *non-iridescent structural coloration*. In Sec. 6.2 we derive an analytic formulation for efficient range queries over the appearance model, together with a corresponding importance sampling strategy. These components constitute an efficient appearance model for vivid feather rendering.

6.1 Feather Scattering Model

As discussed in Sec. 3.2, in the real world feathers exhibit three primary forms of coloration: pigment-based color, non-iridescent structural color, and iridescent structural color. Because iridescent effects rely on wave optics and are expensive to integrate into our real-time system, in this section we focus only on pigment-based color and non-iridescent structural color. Pigment-based color feathers have a microstructure similar to that of animal fur, so we adopt the model of Yan et al. [2017] for these. For non-iridescent structural color, we develop a new BCSDF model to account for structural color reflections within a geometric-optics framework. Building on the double-cylinder fiber model of Yan et al. (Fig. 10), we introduce an internal reflective core to represent the medulla interface and derive a corresponding BCSDF lobe.

Under this assumption, the fiber BCSDF is expressed as a sum of four *unscattered* lobes: R, TT, TR_cT, and TRT. R, TT, and TRT are defined identically to those in Yan et al. [2017] (we include their definitions in the supplementary material for completeness), and TR_cT is our new component capturing a transmission–core-reflection–transmission event. Concretely, we model the medulla as a reflective core that produces a specular reflection at the cortex–medulla interface (denoted R_c) and does not allow any volumetric transmission.

Following the standard factorization into longitudinal and azimuthal components, the TR_cT BCSDF is written as:

$$S_{\text{TR}_c\text{T}}(\theta_i, \theta_r, \varphi_i, \varphi_r, h) = \frac{M_{\text{TR}_c\text{T}}(\theta_i, \theta_r)\, N_{\text{TR}_c\text{T}}(h, \varphi_r - \varphi_i)}{\cos^2 \theta_i}. \tag{30}$$

The longitudinal component is modeled as a Gaussian distribution around a shifted specular direction (we use $G(x; \mu, \sigma)$ to denote a Gaussian at x with mean μ and standard deviation σ):

$$M_{\text{TR}_c\text{T}}(\theta_i, \theta_r) = G\big(\theta_r;\, -\theta_i + \alpha_{\text{TR}_c\text{T}},\, \beta_{\text{TR}_c\text{T}}\big)\,, \tag{31}$$

where $\alpha_{\text{TR}_c\text{T}}$ is the longitudinal shift for this path, and $\beta_{\text{TR}_c\text{T}}$ is its longitudinal roughness.

The azimuthal component is given by the product of an attenuation term $A_{\text{TR}_c\text{T}}(h)$ and a Gaussian distribution in the relative azimuth $\varphi = \varphi_r - \varphi_i$:

$$N_{\text{TR}_c\text{T}}(h, \varphi) = A_{\text{TR}_c\text{T}}(h)\, G\big(\varphi - \Phi_{\text{TR}_c\text{T}}(h);\, \beta_\varphi\big). \tag{32}$$

where $\Phi_{\text{TR}_c\text{T}}(h)$ is the azimuthal center (the direction of the core-reflected ray), and β_φ is the azimuthal roughness.

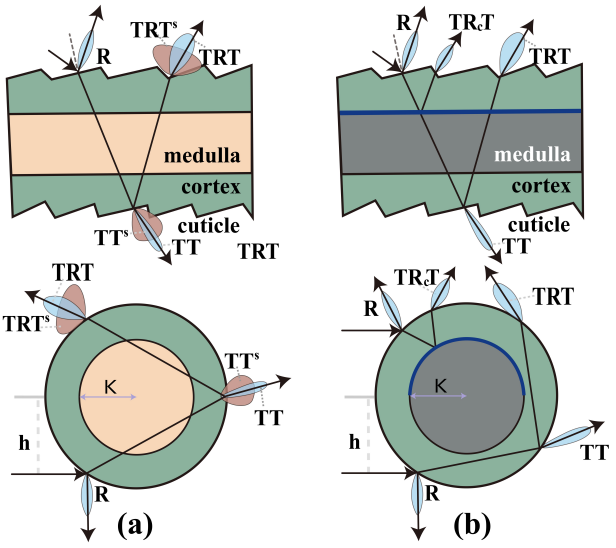

Fig. 10. BCSDF of Yan et al. [2017] and ours. (a) is the BCSDF of Yan et al., which treats the medulla as a scattering cylinder and has five lobes R, TT, TRT, TT^s, TRT^s in total. (b) is our BCSDF only considering the reflectance of medulla and thus has four lobes R, TT, TRT, TR_cT.

The attenuation term accounts for Fresnel factors and Beer–Lambert absorption along the path:

$$A_{\mathrm{TR_cT}}(h) \approx (1 - F_c)^2 \, R_{\mathrm{core}} \, \exp\left(-\frac{\sigma_{c,a}\, s_c(h)}{\cos\theta_d}\right). \tag{33}$$

Here, F_c is the Fresnel reflectance at the outer cuticle interface, so $(1 - F_c)^2$ represents the transmission into and out of the fiber. R_{core} is the specular reflectance at the medulla interface. $\sigma_{c,a}$ is the absorption coefficients in the cortex. The function $s_c(h)$ denotes the path length through the cortex (for a given h), and θ_d is the longitudinal difference angle used to correct for the inclined path length inside the fiber.

6.2 Unified Full-shading model for range query

During rendering, our full model is:

$$f(\omega_i, \omega_o) \;=\; f_s(\omega_i, \omega_o) \;+\; f_r(\omega_i, \omega_o)\,, \tag{34}$$

where f_s is the base surface component and f_r represents the additional grazing-angle scattering responsible for the perceived "fuzz."

We use a surface-based representation for feathers, and thus the BSDF $f_s(\omega_i, \omega_o)$ is composed of several components: the rachis BCSDF S_r, the barb BCSDF S_b, the barbule BCSDFs S_{bd} (above the barb) and S_{bp} (below the barb), and a delta transmission term through the feather. Formally, we write:

$$\begin{aligned} f_s(\omega_i, \omega_o) = \; & [\; w_r(\omega_o)\, S_r(\omega_i, \omega_o;\; h_r(\omega_o)) \\ & +\; w_b(\omega_o)\, S_b(T_b\omega_i, T_b\omega_o;\; h_b(\omega_o)) \\ & +\; w_{bd}(\omega_o)\, S_{bd}(T_{bd}\omega_i, T_{bd}\omega_o;\; h_{bd}(\omega_o)) \\ & +\; w_{bp}(\omega_o)\, S_{bp}(T_{bp}\omega_i, T_{bp}\omega_o;\; h_{bp}(\omega_o)) \\ & +\; w_t(\omega_o)\, \delta(1 + \omega_i \cdot \omega_o)\;]\; |\omega_i \cdot n|^{-1}\,. \end{aligned} \tag{35}$$

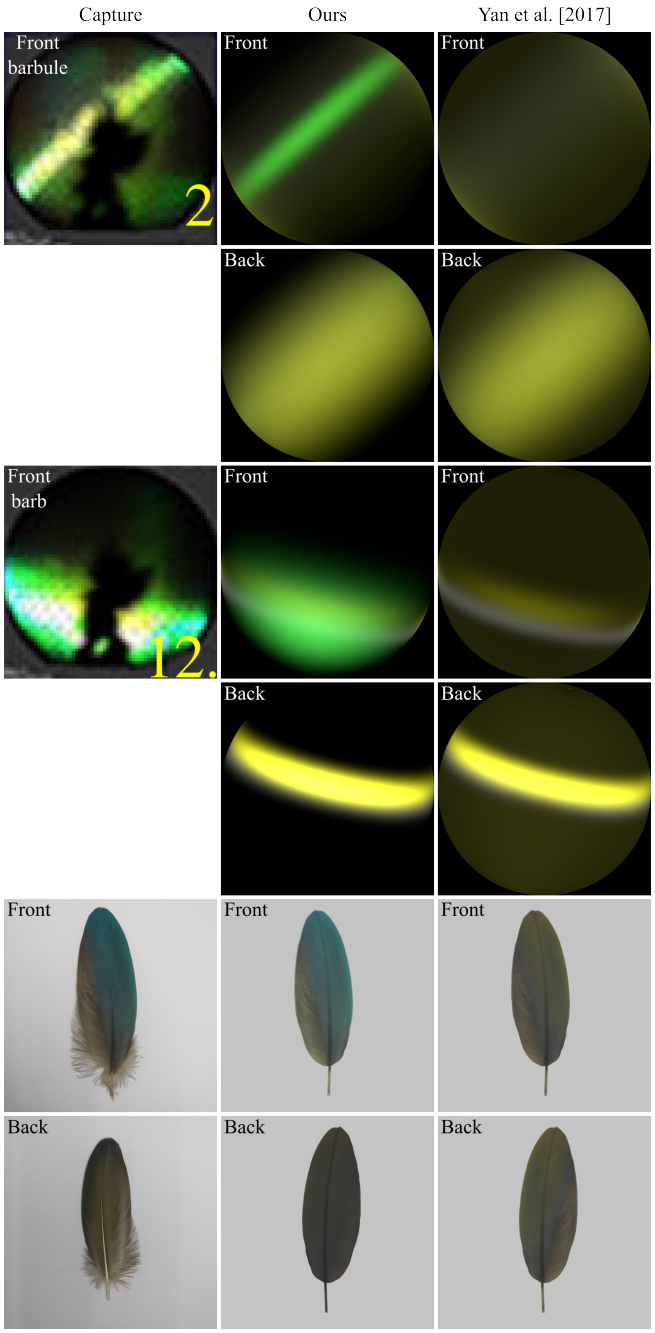

Fig. 11. The validation of our BCSDF. In the top four rows, we match the front barb or barbule reflectance capture from Harvey et al. [2013] using our BCSDF and that of Yan et al. [2017]. The dark area in the capture is the shadow cast by the camera. It is obvious that our BCSDF matches the capture better for both barbs (middle two) and barbules (top two), and our back BCSDF is more specular as we only consider reflectance of the medulla. Last two rows show a match of the front and back side of a feather. Our BCSDF match the feather well on both sides, while the BCSDF of Yan et al. fails.

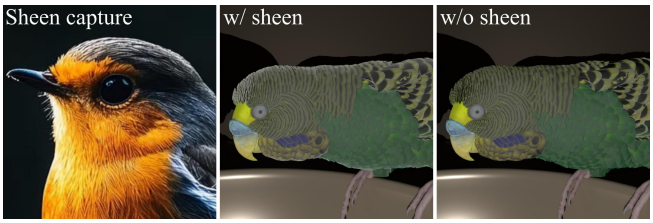

Fig. 12. The effect of our sheen terms. Similar to the real capture, the parrot rendering with sheen looks more fluffy especially at the grazing angles, making it more realistic than the rendering without sheen.

In this formulation, n is the surface normal. Each weight $w_p(\omega_o)$ (for example, $w_r(\omega_o)$ for the rachis) represents the contribution of that part and is computed using the Gaussian function introduced earlier. The term $w_t(\omega_o) = 1 - \big(w_r(\omega_o) + w_b(\omega_o) + w_{bd}(\omega_o) + w_{bp}(\omega_o)\big)$ represents the transparency ratio, and $\delta(\cdot)$ is the Dirac delta function. The operator T transforms a direction to the local frame of the given feather part, defined such that n is the local z-axis and the feather's tangent (pointing direction) is the local x-axis. Finally, h defines the integration range for multi-scale BCSDFs, which is from -1 to 1 in the range-query case, representing a far-field BCSDF.

Sheen term. To approximate the soft, fuzzy appearance of feathers, we augment the surface BSDF with a view-dependent sheen term f_r representing the additional grazing-angle scattering responsible for the perceived "fuzz", as in Fig. 12. We follow the approach of Zeltner et al. [2022] for this view-dependent sheen term.

6.2.1 Importance Sampling. Our importance sampling method operates on the fiber surface and is performed within a pixel footprint. It consists of two main steps: (1) selecting a BCSDF component by area ratio, and (2) selecting a lobe by intensity. This procedure is similar to that of d'Eon et al. [2011]. Specifically, our sampling can be summarized in the following four stages:

- **Select a BCSDF by area ratio.** As shown in Eq. (35), we treat BSDF as a mixture of multiple BCSDF lobes plus a delta transmission term. Each lobe's mixture weight is given by $w_p(\omega_o)$ (e.g., $w_r, w_b, w_{ba}, w_{bb}, w_t$), and these weights are normalized so that
$$\sum_p w_p(\omega_o) = 1\,.$$
Therefore, we select lobe p with probability
$$P(p) = w_p(\omega_o)\,.$$
- **Select a lobe by intensity.** Next, the candidate lobes are weighted according to the energy they carry. Since the longitudinal term M_p and the azimuthal distribution term D_p are both normalized, the energy carried by lobe p is determined by its attenuation factor A_p. We calculate A_p for each of the nine lobes (and compute their total $\sum_p A_p$). The probability of picking lobe p is then
$$P_{\text{lobe}}(p) = \frac{A_p}{\sum_q A_q}\,.$$

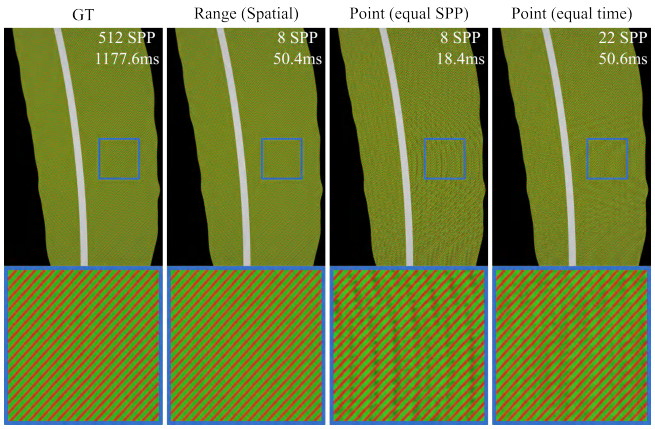

Fig. 13. Comparison of range query and point query. We use red for barbs and green for barbules. The renderings show that our range query result is similar to GT without aliasing, while both the results of equal SPP and time show obvious aliasing.

- **Sample an outgoing direction.** Similar to d'Eon et al. [2011], we then sample an outgoing direction (θ_r, ϕ_r) from the chosen lobe by separately sampling its longitudinal and azimuthal components according to their respective distributions (for example, a Gaussian or cosine distribution for the longitudinal component, and a Gaussian or uniform distribution for the azimuthal component).
- **Compute the PDF and sampling weight.** The final probability density function (PDF) for the sampled direction is the product of:
  - the BCSDF selection probability $P(p) = w_p(\omega_o)$ from the first step,
  - the lobe selection probability $P_{\text{lobe}}(p) = A_p / \sum_q A_q$ from the second step, and
  - the selected lobe's longitudinal and azimuthal PDFs from the third step (converted from (θ, ϕ) space to solid angle).

  The final sampling weight is the value of the selected lobe's BCSDF at the sampled direction divided by this overall PDF, i.e., divided by the product of the BCSDF selection, lobe selection, and directional sampling probabilities.

7 Results

In this section we show the results using our feather model. All scenes are rendered using path tracing on an NVIDIA GeForce RTX 3090 machine in the Falcor [Kallweit et al. 2022] framework in 1080P.

Our extraction pipeline. In addition to Fig. 4, we provide more results of our extraction pipeline in Fig. 14. The extracted rachis and contour fit the inputs well, which show the robustness of our extracting method. Moreover, we make a quantitative comparison on the rendering input under a normalized range from 0 to 1. For rachis, we sample the same number of points on them and calculate a mean distance. Its value 0.00552 means our extracted rachis is very similar to that of the input. For contour, we adopt the dice similarity coefficient commonly used in image segmentation, and its value

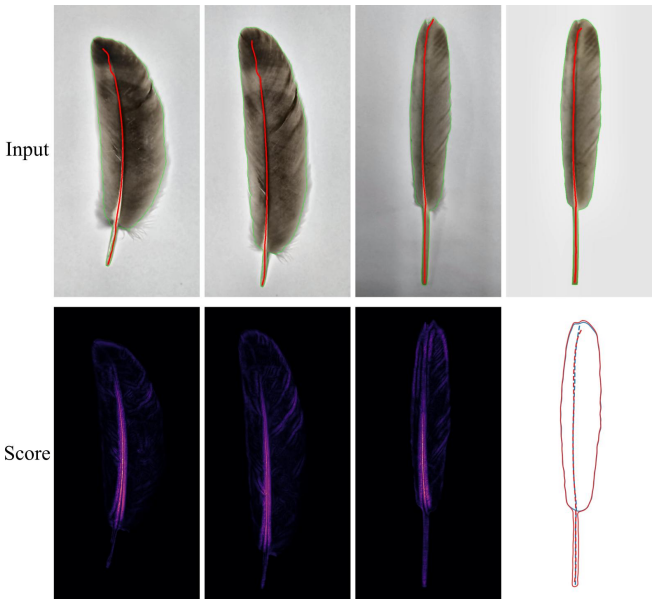

Fig. 14. Some more results of our extraction pipeline. The left two inputs are captures from the feather in Fig. 4 at different angles. The third input is another feather, and its rendering with extracted parameters is the last input. We take the extracted geometry as reference for quantitative evaluation. The extracted rachis and contour are drawn on the input in red and green, which show great fitting. In the second row, we display score maps used to decide rachis except the last input. Instead, we compare the extracted (red) and reference (blue) rachis and contour, which are nearly overlapping.

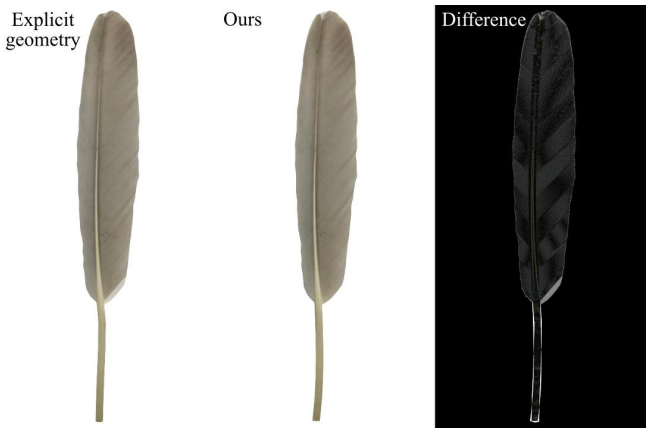

Fig. 15. Comparison with explicit curve-based geometry. Due to manual adjustment of geometry and UV, as shown in the difference, the explicit geometry does not match our representation in the canonical space well. But on the whole, our result is similar to the rendering of explicit geometry visually, which occupies 1.5 GB while we only take up 500 KB. Note that we use far-field shading model and only single scattering here.

0.9596 is very close to 1, which means our extracted contour is also very similar to the reference.

Match to real feather. We compared our results from two perspectives: (1) matching with the measured BCSDF, and (2) matching with the single photo we captured. We evaluate our method against real photographs and the BCSDF model of Yan et al. [2017]. As shown in Fig. 11, our approach more closely reproduces the captured appearance, indicating that the proposed BCSDF provides a more accurate approximation of real feather scattering than the previous work. Fig. 16 presents a qualitative comparison against photographic references (first column), followed by renderings under novel lighting and viewing conditions. Even in close-up views, our model reproduces fine-scale structure, translucency, and view-dependent effects with high fidelity. Additionally, we match some captures of feathers with different BCSDFs at different viewing angles in Fig. 17. Although our method can match captures well, some results reveal our shortcomings. As shown in Fig. 16, our albedo estimation process may cause baked-in shadows, leading to deviations from the true albedo. Additionally, our low pass Gaussian filter (standard deviation taken as 5 in image space) diminishes sharp features like black and white patch boundaries in the fifth row.

Our Multiscale Model. In Fig. 13, we show that our multiscale model can effectively prevent aliasing at the geometric level. In Fig. 8, we demonstrate our hierarchical strategy that chooses between the spatial domain and the frequency domain in the near-field and far-field based on efficiency. In the far-field, the spatial-domain range-query method can take up to 160 s, making real-time performance impossible; in contrast, the frequency-domain approach achieves the same effect in only 16 ms. We also show that the two approaches produce consistent results.

Memory. Finally, Figs. 18 and 19 showcase full feathered models of a parrot and an eagle. These examples demonstrate that our method scales from isolated feather appearance to complex, full-animal renderings while preserving details. Our method only takes about 0.02% memory compared to explicitly modeling detailed barbule-level geometry, while we show close appearance to the curve-based model in comparison, as shown in Fig. 15.

Performance. Our proposed method runs in real time. In Fig. 8 and Fig. 18, we demonstrate real-time performance (more than 55 FPS) under both simple lighting (a point light) and direct illumination (DI). In Fig. 16, we further show that rendering a single feather combined with our importance sampling strategy takes only 10 ms under complex environment lighting conditions. Note that Fig. 1 and Fig. 19 in the paper show global illumination results under complex lighting (environment lighting). These are computationally expensive, so the teaser is accumulated. We also further demonstrate our real-time performance in the accompanying video.

8 Conclusion

We have presented a practical, real-time feather appearance model that preserves both visual realism and computational efficiency. By leveraging a hierarchical procedural representation and an analytic level-of-detail approach, our method achieves high fidelity while drastically reducing complexity. We propose a new feather representation, including structural color BCSDF model and a sheen lobe enables our renderer to closely match the appearance of real

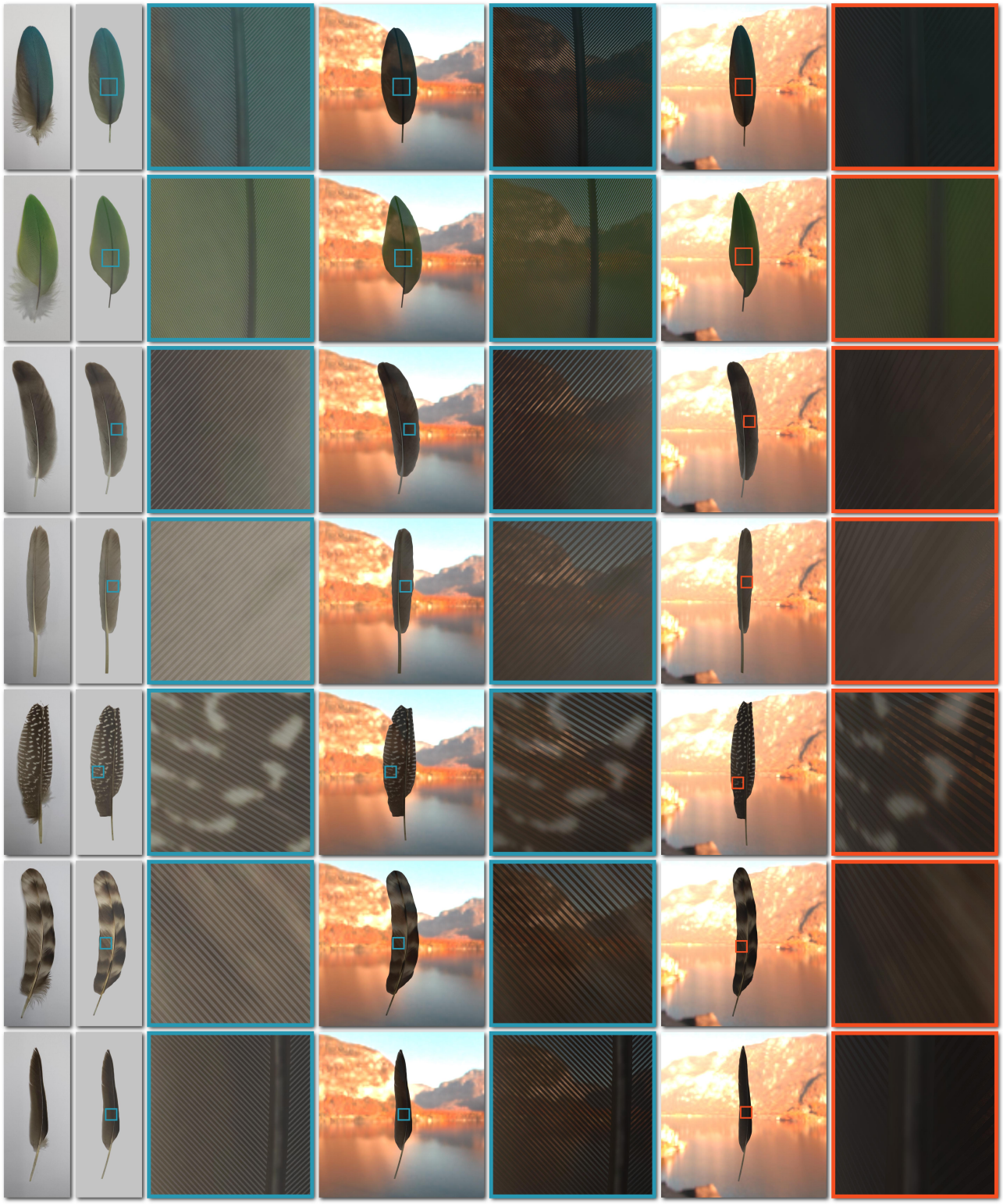

Fig. 16. Matching results of our method. The leftmost column shows seven feather captures, and right of them are our matching renderings with a crop, which prove that our method can match the capture well. We also display the renderings under an environment map with different viewing angles. Note that even under complex environment lighting conditions, with our importance sampling strategy, our rendering time is still under 10 ms for each feather with 16spp.

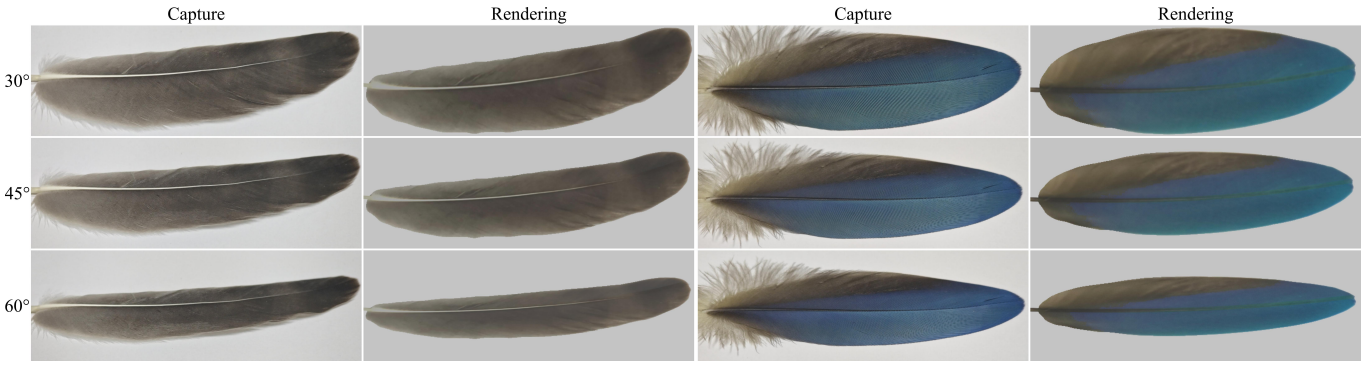

Fig. 17. Matching results of our method at three viewing angles. We match two feathers with different BCSDFs at gradually changing viewing angles under diffuse lighting, and the similar appearances prove the effectiveness of our method.

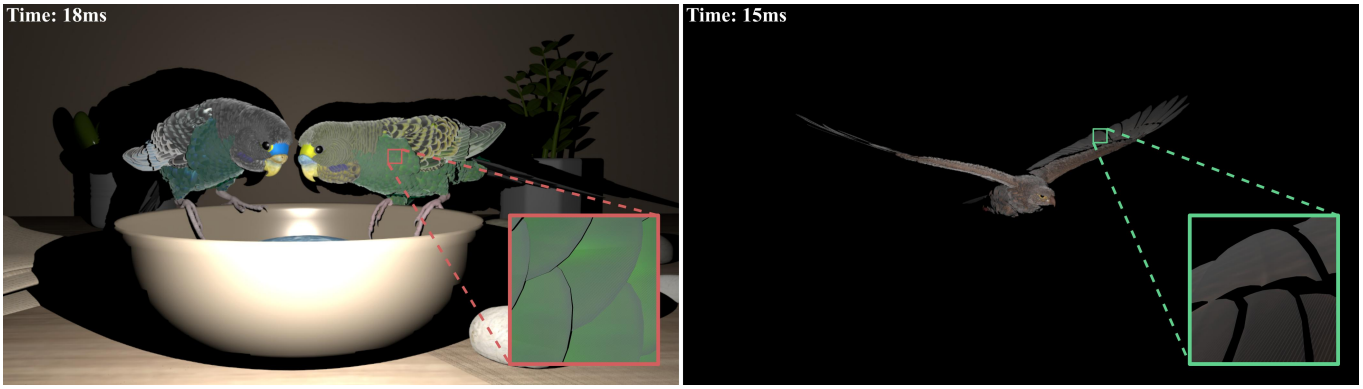

Fig. 18. The renderings of the whole birds under 4 SPP with only direct illumination. Our method supports real-time rendering of feathers, with only 18 ms for two parrots (6250 feathers) and 15 ms for an eagle (6350 feathers).

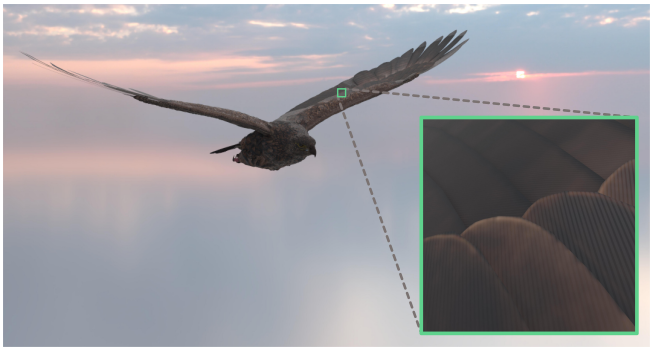

Fig. 19. Rendering of an eagle under an environment map. Our method achieves good renderings for both far-field and near-field views.

feathers, reproducing subtle effects like fluffy rim lighting and vivid non-iridescent colors.

Our results demonstrate significant performance gains: under simple lighting conditions, full bird scenes render more than 50 FPS with smooth, multi-scale transitions, using a memory footprint that is over 1/6000 of an explicit fiber model. Despite this extreme compression, the rendered output remains indistinguishable from real photographs, validating the accuracy of our appearance aggregation. The method's stable frame rates across varying view distances and lighting conditions make it well-suited for real-time applications such as games and VR. In summary, our work provides a practical solution for realistic feather rendering, delivering film-quality appearance at an interactive speed and bridging the gap between offline realism and real-time performance.

9 Limitations and Future Work

Barbule noise. The feather appearance typically exhibits two kinds of noise: one at the barb level and another at the finer-scale barbule level. In Sec. 4.3, we have already discussed barb noise: by introducing controllable perturbations to the feather contour and compositing noise across scales, we can reliably produce the "torn"/frayed look observed along real feather edges. In contrast, barbule noise manifests as more random, non-stationary micro-texture patterns. We currently use frequency-domain filtering (with band-limited assumptions) for multiscale control. To reproduce such effects, we can fall back to our spatial range-query model for evaluation, with a small speed cost. In the future, we plan to explore actual distributions

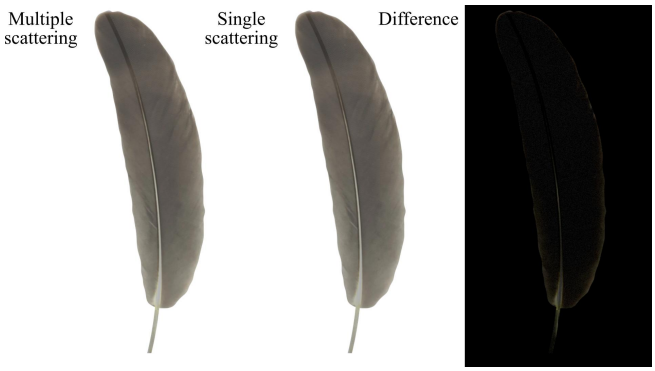

Fig. 20. Influence of multiple scattering. We compare renderings of a single feather with multiple scattering and single scattering. The difference shows that multiple scattering does not have much impact.

including variance-preserving filtering, blue-noise sampling, or a learned filter conditioned on local density, orientation distribution, and noise parameters.

Other complex feathers. Our representation models a feather on a plane. This makes it hard to represent out-of-plane structures such as down-like feathers. In practice, for a bird, down feathers are usually hidden under the outer plumage, so the visible feathers are almost entirely structured contour feathers. In addition to downy feathers and planar feathers, there are also contour feathers with clearly visible individual barbs and barbules that we do not address. Naive extensions of our presented method to support this may cause appearance and performance problems.

Wave optics in real-time. Our model focuses on geometric-optics and fiber scattering (extended lobes with structural parameters) to support real-time rendering under general illumination. Adding interference or diffraction typically requires wavelength-domain evaluation or heavier precomputation and caching. This would be hard to render under a real-time budget. We therefore do not include wave optics in the real-time system, though it can be added in offline rendering. In future work, we plan to investigate a more compact appearance model for wave-optic effects.

Multiple scattering. Multiple scattering in dense barbules is important for rim glow at grazing angles, backlighting, and energy distribution under HDR lighting. However, we do not consider it since multiple scattering has limited impact in common situations, as shown in Fig. 20. In future work, we consider using a lightweight neural network to better model these effects.

Acknowledgments

We thank the reviewers for the valuable comments. This work has been partially supported by Jing-Jin-Ji Regional Integrated Environmental Improvement-National Science and Technology Major Project (No. 2025ZD1202205); the National Natural Science Foundation of China (No. 62502285, No. 62272275); University of Melbourne 2026 Early Career Researcher Grant PRJ_026928; CCF-NetEase ThunderFire Innovation Research Funding (NO. CCF-Netease 202503).

References

Jessica Baron, Daljit Singh Dhillon, and Eric Patterson. 2021. Procedural Shading for Rendering the Appearance of Feathers. In *SIGGRAPH '21: Special Interest Group on Computer Graphics and Interactive Techniques Conference Posters*. 1–2. doi:10.1145/3450618.3469161

Jessica Baron, Daljit Singh Dhillon, N. Adam Smith, and Eric Patterson. 2022. Microstructure-based appearance rendering for feathers. *Computers & Graphics* (2022), 452–459. doi:10.1016/j.cag.2021.09.010

Jessica Baron and Eric Patterson. 2019. Procedurally generating biologically driven feathers. In *Computer Graphics International Conference*. Springer, 342–348.

Juli Broggi, Anna Gamero, Esa Hohtola, Markku Orell, and Jan-Åke Nilsson. 2011. Interpopulation variation in contour feather structure is environmentally determined in great tits. *PLoS One* 6, 9 (2011), e24942.

Isabelle Busch. 2024. Everything you ever wanted to know about feathers. https://www.isabellebusch.com/nature-blog/identifying-amp-cleaning-feathers.

Yanyun Chen, Yingqing Xu, Baining Guo, and Heung Yeung Shum. 2002. Modeling and rendering of realistic feathers. *ACM Transactions on Graphics* 21, 3 (2002), 630–636. doi:10.1145/566654.566628

Matt Jen-Yuan Chiang, Benedikt Bitterli, Chuck Tappan, and Brent Burley. 2016. A Practical and Controllable Hair and Fur Model for Production Path Tracing. *Computer Graphics Forum* 35, 2 (2016), 275–283. doi:10.1111/cgf.12830

Eugene d'Eon, Guillaume Francois, Martin Hill, Joe Letteri, and Jean-Marie Aubry. 2011. An energy-conserving hair reflectance model. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1181–1187.

A Roland Ennos, John RE Hickson, and ANNA Roberts. 1995. Functional morphology of the vanes of the flight feathers of the pigeon Columba livia. *Journal of Experimental Biology* 198, 5 (1995), 1219–1228.

Cristiano G. Franco and Marcelo Walter. 2002. Modeling and Rendering of Individual Feathers. In *15th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2002), 7-10 October 2002, Fortaleza-CE, Brazil*. IEEE Computer Society, 293–299. doi:10.1109/SIBGRA.2002.1167157

Cristiano G. Franco and Marcelo Walter. 2007. Direct Texture Synthesis of Feather Pigmentation Patterns. In *GRAPP 2007 - International Conference on Computer Graphics Theory and Applications*. 277–283. doi:10.5220/0002076602770283

Alejandro Frangi, W.J. Niessen, Koen Vincken, and Max Viergever. 1998. Multiscale Vessel Enhancement Filtering. *Lect Notes Comput Sci* 1496, 130–137. doi:10.1007/BFb0056195

Rasmus Haapaoja and Christoph Genzwürker. 2019. Mesh-Driven Generation and Animation of Groomed Feathers. In *SIGGRAPH '19 Talks*. 1–2. doi:10.1145/3306307.3328178

Todd Alan Harvey, Kimberly S Bostwick, and Steve Marschner. 2013. Directional reflectance and milli-scale feather morphology of the African Emerald Cuckoo, Chrysococcyx cupreus. *Journal of the Royal Society Interface* 10, 86 (2013), 20130391.

Sebastian Hendrickx-Rodriguez and David Lentink. 2025. The feather's multi-functional structure across nano to macro scales inspires hierarchical design. *Journal of the Royal Society Interface* 22, 225 (2025), 20240776.

Philipp Henzler, Niloy J. Mitra, and Tobias Ritschel. 2020. Learning a Neural 3D Texture Space from 2D Exemplars. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 8353–8361. https://openaccess.thecvf.com/content_CVPR_2020/papers/Henzler_Learning_a_Neural_3D_Texture_Space_From_2D_Exemplars_CVPR_2020_paper.pdf

Hideki Hirayama, Kazufumi Kaneda, Hideo Yamashita, and Yoshimi Monden. 2001. An accurate illumination model for objects coated with multilayer films. *Computers & Graphics* 25, 3 (2001), 391–400.

Weizhen Huang, Sebastian Merzbach, Clara Callenberg, Doekele Stavenga, and Matthias Hullin. 2022. Rendering Iridescent Rock Dove Neck Feathers. In *SIGGRAPH '22 Conference Proceedings*. 1–8. doi:10.1145/3528233.3530749

Simon Kallweit, Petrik Clarberg, Craig Kolb, Tom'aš Davidovič, Kai-Hwa Yao, Theresa Foley, Yong He, Lifan Wu, Lucy Chen, Tomas Akenine-Möller, Chris Wyman, Cyril Crassin, and Nir Benty. 2022. The Falcor Rendering Framework. https://github.com/NVIDIAGameWorks/Falcor https://github.com/NVIDIAGameWorks/Falcor.

Ares Lagae, Sylvain Lefebvre, Robert Cook, Tony DeRose, George Drettakis, David S. Ebert, John P. Lewis, Ken Perlin, and Matthias Zwicker. 2010. A Survey of Procedural Noise Functions. *Computer Graphics Forum* 29, 8 (2010), 2579–2600. doi:10.1111/j.1467-8659.2010.01827.x

Ares Lagae, Sylvain Lefebvre, George Drettakis, and Philip Dutré. 2009. Procedural Noise Using Sparse Gabor Convolution. *ACM Transactions on Graphics* 28, 3 (2009). doi:10.1145/1576246.1531360

Jerome Lengyel, Emil Praun, Adam Finkelstein, and Hugues Hoppe. 2001. Real-time fur over arbitrary surfaces. In *Proceedings of the 2001 symposium on Interactive 3D graphics*. 227–232.

Albert Julius Liu, Zhao Dong, Miloš Hašan, and Steve Marschner. 2016. Simulating the Structure and Texture of Solid Wood. *ACM Transactions on Graphics* 35, 6 (2016). doi:10.1145/2980179.2980255

Alfred Martin Lucas. 1972. *Avian anatomy: integument*. Vol. 2. US Agricultural Research Service.

Stephen R. Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. 2003. Light Scattering from Human Hair Fibers. *ACM Transactions on Graphics* 22, 3 (2003), 780–791. doi:10.1145/882262.882345

Nobuyuki Otsu et al. 1975. A threshold selection method from gray-level histograms. *Automatica* 11, 285-296 (1975), 23–27.

Juan Raúl Padrón-Griffe, Dario Lanza, Adrián Jarabo, and Adolfo Muñoz. 2024. A Surface-based Appearance Model for Pennaceous Feathers. *Computer Graphics Forum* 43, 7 (2024), e15235. doi:10.1111/cgf.15235

Darwyn R. Peachey. 1985. Solid Texturing of Complex Surfaces. *Computer Graphics (Proceedings of SIGGRAPH '85)* 19, 3 (1985), 279–286. doi:10.1145/325165.325246

Ken Perlin. 1985. An Image Synthesizer. *Computer Graphics (Proceedings of SIGGRAPH '85)* 19, 3 (1985), 287–296. doi:10.1145/325165.325247

John P. Rafferty. 2025. feather. https://www.britannica.com/science/feather.

Lisa Streit and Wolfgang Heidrich. 2002. A Biologically-Parameterized Feather Model. *Computer Graphics Forum* 21 (2002), 565–573. doi:10.1111/1467-8659.t01-1-00707

Satoshi Suzuki and Keiichi Abe. 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing* 30, 1 (1985), 32–46. doi:10.1016/0734-189X(85)90016-7

Jan Tinbergen, Bodo D Wilts, and Doekele G Stavenga. 2013. Spectral tuning of Amazon parrot feather coloration by psittacofulvin pigments and spongy structures. *Journal of Experimental Biology* 216, 23 (2013), 4358–4364.

Beibei Wang, Miloš Hašan, Nicolas Holzschuch, and Ling-Qi Yan. 2020. Example-Based Microstructure Rendering with Constant Storage. *ACM Transactions on Graphics* 39, 5 (2020), 1–12. doi:10.1145/3406836

Ling-Qi Yan, Henrik Wann Jensen, and Ravi Ramamoorthi. 2017. An efficient and practical near and far field fur reflectance model. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

Ling-Qi Yan, Chi-Wei Tseng, Henrik Wann Jensen, and Ravi Ramamoorthi. 2015. Physically-Accurate Fur Reflectance: Modeling, Measurement and Rendering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2015)* 34, 6, Article 185 (2015), 13 pages. doi:10.1145/2816795.2818080

Haiwei Yin, Biqin Dong, Xiaohan Liu, Tianrong Zhan, Lei Shi, Jian Zi, and Eli Yablonovitch. 2012. Amorphous diamond-structured photonic crystal in the feather barbs of the scarlet macaw. *Proceedings of the National Academy of Sciences* 109, 27 (2012), 10798–10801.

Yunchen Yu, Andrea Weidlich, Bruce Walter, Eugene d'Eon, and Steve Marschner. 2024. Appearance Modeling of Iridescent Feathers with Diverse Nanostructures. *ACM Transactions on Graphics* 43, 6 (2024). doi:10.1145/3687983

Tizian Zeltner, Brent Burley, and Matt Jen-Yuan Chiang. 2022. Practical multiple-scattering sheen using linearly transformed cosines. In *ACM SIGGRAPH 2022 Talks*. 1–2.