

Curvature-Aware Multi-Scale Gaussian Appearance Model for Fabric Flyaways

ANONYMOUS AUTHOR(S)
SUBMISSION ID: 2348

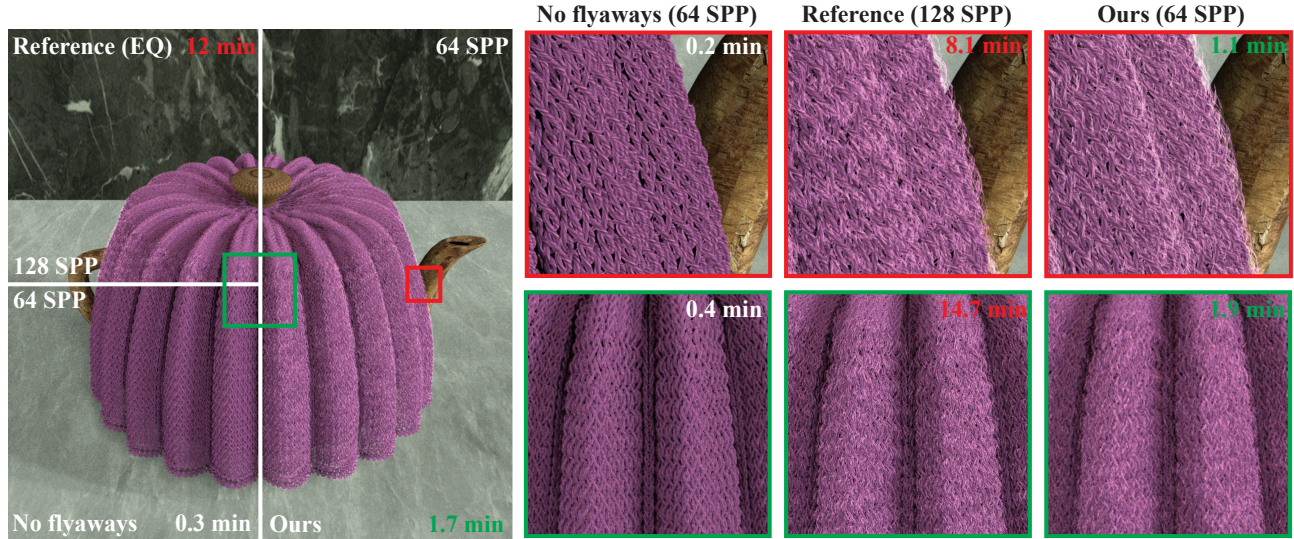


Fig. 1. We propose an appearance model for fabric flyaways using 3D Gaussian primitives, with adaptive multi-scale merging of Gaussian mixtures based on pixel footprint. The example shows a fuzzy knitted teapot, illustrating the visual impact of flyaways that are typically neglected in prior work. In far-view, the flyaways create a faint bright silhouette, known as the sheen effect and make the overall appearance less uniform, akin to real-world fabrics. Our method closely matches a reference rendered with explicit curve geometry, while achieving up to seven to 7.4 times faster performance on average across scales and avoiding expensive precomputation and storage costs by using a single tile-able patch of flyaways represented as Gaussians. Please refer to the supplementary for more evaluations.

Real-world fabrics exhibit imperfections arising from irregularities in yarn formation, fabric stitching, and everyday wear. These imperfections cause fine fibers to protrude from the fabric surface, producing a characteristic “fuzzy” appearance commonly referred to as flyaways. State-of-the-art approaches model flyaways as explicit curves or a volumetric medium layered on top of the base fabric. However, rendering such representations is computationally expensive and challenging. In this paper, we introduce a multi-scale model that represents flyaways using Gaussians stored in a tilable 3D texture. Modeling flyaways as Gaussians enables efficient aggregation based on the texture footprint: individual fibers are resolved in close-up views, while distant views naturally blend into a soft, hazy silhouette. Our approach integrates seamlessly with existing surface-based or curve-based fabric models that define the base layer. Unlike methods that generate thousands of explicit curve primitives—leading to complex bounding volume hierarchies and high samples-per-pixel requirements to reduce variance—our multi-scale formulation leverages the texture footprint to adaptively determine the number of Gaussians needed. As a result, our method achieves up to 7 to 40 times faster rendering while reducing storage requirements by up to five orders of magnitude compared to volumetric representations, and by over two orders of magnitude compared to explicit curve-based models, all while providing greater artistic and procedural control over the perceived fuzziness of fabrics.

1 Introduction

Hairy and fibrous materials are ubiquitous in the physical world and play an important role in visual realism. They appear as fuzz on cloth, peach fuzz on skin, body or facial hair on humans, and fur on animals. Although these fibers are often subtle and sparsely distributed, they strongly influence a material’s perceived softness, thickness, and realism, particularly at grazing angles and their effect on silhouettes. Even when individual fibers are not directly visible, their collective presence contributes to characteristic visual phenomena such as sheen, glow, and blurred outlines, making them essential for convincing rendering across a wide range of scales and viewing conditions.

Modeling such materials is fundamentally challenging because they do not fit cleanly into existing rendering abstractions. Unlike dense hair or fur, flyaway fibers are sparse, irregularly oriented, and weakly correlated with one another. At the same time, they cannot be treated as texture maps for textures fail to capture their volumetric extent and silhouette interaction. As a result, fibrous materials occupy an uncomfortable middle ground between volumetric representations and explicit curve-based models, inheriting the disadvantages of both.

Existing approaches typically resolve this tension by explicitly modeling flyaways as curves or fibers, borrowing techniques from hair and fur rendering. While physically accurate, such methods are expensive to generate, memory intensive, and costly to render, especially when high sampling rates are required to resolve thin geometry. Moreover, explicit representations scale poorly across viewing distances: fibers that are necessary for close-up realism become expensive when rendered across large surfaces. Conversely, faster alternatives using textures, shell maps, or volumetric grids trade efficiency for accuracy, often failing to reproduce correct silhouettes, directional variability, or scale-dependent appearance.

At the core of this problem lies the observation that flyaway fibers do not need to be explicitly modeled as individual curves at all scales. What ultimately matters is not the precise geometry of each fiber, but their aggregate geometric and optical effect within a pixel footprint. This observation motivates a representation that is continuous, analytically tractable, and naturally supports scale-dependent aggregation.

In this work, we propose a multi-scale Gaussian representation for flyaway fibers that replaces explicit curve geometry with a hierarchy of Gaussian primitives. At the finest scale, individual fibers are approximated by sequences of anisotropic Gaussians that closely match their geometric and shading behavior. As the viewing distance increases, these Gaussians are merged into coarser representations based on the pixel footprint, preserving the correct silhouette and appearance while dramatically reducing sampling cost. This formulation enables efficient, artifact-free rendering across scales, without resorting to discrete level-of-detail meshes or expensive volumetric integration.

Our approach integrates with existing surface- or curve-based fabric models, treating flyaways as an additional, lightweight layer with its own explicit geometry and appearance, influencing as well the perceived geometry and appearance of the underlying cloth. By leveraging the analytic properties of Gaussians and curvature properties of the base surface, we intersect the Gaussians against rays bending in accordance with the surface curvature, making the method suitable for high-quality offline rendering as well as scalable production pipelines. In summary, our contributions are:

- A novel multi-scale representation of flyaway fibers based on Gaussian primitives.
- An analytic merging strategy that adapts flyaway geometry and appearance to pixel footprint.
- A precomputation-free curvature-aware inverse displacement mapping strategy that enables updating the silhouettes and generalizes to texture layers, besides Gaussians.
- A rendering formulation that preserves silhouette and sheen effects while significantly reducing computational cost.

2 Related Work

Cloth rendering methods span surface-based, curve-based, aggregated or general learned representations, trading off efficiency against fiber-level fidelity, particularly for sparse flyaway fibers.

Curve/Volumetric Fabric Models. Volumetric fabric models [Zhao et al. 2012, 2014] represent fiber assemblies as heterogeneous volumes, achieving high visual fidelity via micro-CT imaging but incurring high memory and computational cost, particularly inefficient for flyaway fibers that occupy large regions of empty space and contribute only to sparse portion of these regions. Curve-based models [Khungurn et al. 2016] explicitly represent fiber level geometry using curved cylindrical primitives, enabling accurate microscopic detail at the cost of large storage and expensive multiple scattering simulation. Procedural models aggregate the fibers at different levels of abstraction, including ply-level [Montazeri et al. 2021, 2020] and yarn-level [Khattar et al. 2026b; Zhu et al. 2023a], reducing geometric complexity but remaining fundamentally curve-based, incurring significant overhead for sparse structures such as flyaway fibers.

Surface-Based Fabric Models. Surface-based approaches represent fabrics as a 3D base mesh, using BRDFs [Irawan and Marschner 2012; Sadeghi et al. 2013] to model appearance efficiently, but missing fiber-level effects such as transmission, masking-shadowing, parallax, and sheen. Recent models address several of these limitations, reproducing anisotropic highlights, transmittance, parallax and sheen either through precomputed texture maps [Zhu et al. 2024, 2023a] or texture-free implicit representations [Khattar et al. 2026a, 2025]. Despite these advances, surface-based methods cannot model flyaway fibers that extend beyond the base surface and therefore fail to reproduce silhouette softening or the characteristic fuzzy halo of real fabrics.

Gaussian Representations. Flyaway fibers form sparse volumetric structures extending beyond the base surface that are better suited to explicit representations, which 3D Gaussian Splatting (3DGS) [Kerbl et al. 2023] provides through fuzzy anisotropic Gaussians. The simplicity and flexibility of 3DGS has inspired Level-of-Detail solutions for large scale scenes [Kerbl et al. 2024; Liu et al. 2025], ray tracing extensions [Condor et al. 2025; Moenne-Loccoz et al. 2024; Wu et al. 2025], and hair [Zakharov et al. 2024] and garments [Rong et al. 2025]. We build on this foundation, introducing a multiscale Gaussian representation for flyaway fibers with an explicit appearance model for fabrics and ray tracing formulation.

3 Background and Challenges

Fabrics consist of two layers: (i) a base layer defined by a repeating tiling pattern, and (ii) flyaways, random fibers sparsely distributed protruding from the base layer, giving the fabric a fuzzy appearance. Recent advances in fabric rendering represent the base layer using explicitly generated curves [Khattar et al. 2026b; Zhu et al. 2023b] or a mesh [Jin et al. 2022; Khattar et al. 2026a, 2025; Zhu et al. 2024], where the tiling pattern is represented analytically or using texture maps. While effective for the base layer, these approaches produce unnaturally uniform fabrics that fail to capture flyaway imperfections arising from material properties, stitching errors, or wear and tear. At a geometric level, flyaways extend beyond the base layer and alter the fabric silhouette. At a visual level, they contribute a fuzzy appearance and, at grazing angles, produce a characteristic glow around the silhouette known as the sheen effect (Figure 2).

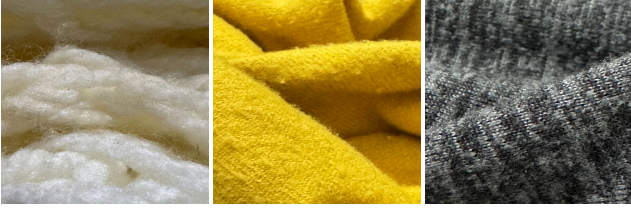


Fig. 2. **Photograph of flyaway fibers in textiles.** Protruding fibers alter both the silhouette and visual appearance of fabrics, producing characteristic fuzziness and sheen at grazing angles.

While [Zhu et al. 2024] model sheen analytically, their expression lacks randomness and does not account for silhouette changes.

Explicitly generating flyaways as curves is geometrically straightforward using tools like XGen Maya [Autodesk, INC. 2019], but rendering them is expensive due to high number of samples per pixel (SPP) and complex ray interactions [Khungurn et al. 2016]. An alternative is to represent the flyaways as 2D or 3D textures. 2D textures require fine ray steps to avoid parallax artifacts (“wall effect”) and cannot update the mesh silhouette, while 3D textures could reduce artifacts at the cost of higher memory and rendering time.

We represent flyaways as a tileable patch of Gaussian primitives between the mesh and base layer of the fabric. This allows the number of Gaussians to vary with pixel footprint via merging, reducing SPP requirements, while our curvature-aware inverse displacement mapping updates the mesh silhouette without explicit displacement. At the finest level each flyaway is represented by piecewise Gaussians along its tangent; at the coarsest, the entire patch can be represented with a single Gaussian.

4 Modeling Flyaways as Gaussians

As discussed in Section 3, our approach models flyaways as Gaussian primitives that form a patch tiled across the input fabric surface, which may consist of an entire mesh or individual curves. The pixel footprint of the surface determines how many Gaussians are needed to represent a flyaway patch, thereby influencing the aggregated appearance and overall visibility. This section focuses on representing flyaways at the finest level, where the pixel footprint is smaller than an individual flyaway. We describe how each flyaway curve can be represented using Gaussians (Section 4.1) and how their appearance is modeled to match that of curve-based flyaways (Section 4.2). Finally, we present our marching strategy against Gaussians, which accounts for the changing mesh silhouette in accordance with the local mesh curvature (Section 4.3).

4.1 Representing a Single Curve as Gaussians

A 3D curve is commonly discretized into line segments for efficient rendering and processing. We represent a curve as an ordered sequence of points, $\{P_i = (x_i, y_i, z_i) \mid i \in \{0 \dots n-1\}\}$ and a fixed radius r . To define a stable local frame along the curve, we associate a fixed normal direction with each point that is propagated using a Rotation Minimizing Frame (RMF) [Wang et al. 2008]. Each point is therefore additionally associated with a normal vector

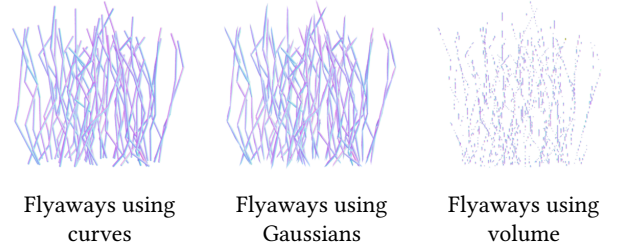


Fig. 3. **Normal maps for flyaways generated as curves v/s volumes v/s Gaussians.** For a small patch of flyaways, we visualize the normal maps when flyaways are represented using curves, volumes, and Gaussians, all with equal resolutions. The Gaussian flyaways closely match the geometry of the curve-based representation, demonstrating that Gaussians are a suitable replacement for modeling flyaways. In contrast, the volumetric representation, is generated by defining a voxel grid where with the normals are stored for voxels that lie on or inside the flyaways and requires significantly more storage to accurately capture the thin flyaways.

$\vec{n}_i = (n_i^x, n_i^y, n_i^z)$. A single 3D Gaussian, on the other hand, is represented using the mean position, μ , the degrees of rotations about the XYZ-axes, e_x, e_y, e_z (Euler angles) which make up the rotation matrix R and the scaling factors, s_x, s_y, s_z for the diagonal scaling matrix S . The analytical expression for a Gaussian is given by,

$$G(x, y, z) = \frac{1}{(2\pi)^{\frac{3}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2} ([x, y, z] - \mu)^T \Sigma^{-1} ([x, y, z] - \mu)}, \quad (1)$$

where (x, y, z) is a point in 3D space and $\Sigma = RSS^T R^T$ is the covariance matrix for the Gaussian. Geometrically, a Gaussian is identified to its one-sigma ellipsoid primitive.

To represent a constant function in 1D using a Gaussian mixture model (GMM), the amount of overlap between neighboring Gaussians must be carefully controlled. A common design principle is to choose the spacing and variance such that the sum of neighboring Gaussians is approximately constant at the one-sigma level. This ensures that the reconstructed function is smooth and minimally “bumpy”. In our formulation, the Gaussian parameters are therefore not learned but chosen analytically based on the curve direction, normal and radius. Therefore, each consecutive pair of curve points P_i and P_{i+1} together with their associated normals $\vec{n}_i$ and $\vec{n}_{i+1}$ can be substituted with a 3D Gaussian,

$$\begin{aligned} \mu &= \frac{1}{2} (P_i + P_{i+1}) \\ s_x, s_y, s_z &= \left\{ \frac{1}{2} \|P_{i+1} - P_i\|, r_i, r_i \right\} \\ e_x, e_y, e_z &= \left\{ 0, \arctan\left(\frac{-n_i^z}{\sqrt{(n_i^x)^2 + (n_i^y)^2}}\right), \arctan\left(\frac{n_i^x}{n_i^y}\right) \right\}. \end{aligned} \quad (2)$$

This allows us to represent a curve comprising N points using $N - 1$ Gaussians and extend it to generate a small patch of flyaways using Gaussians. In Figure 3, we visualize a comparison between the geometry of a small patch of flyaways generated using curves, volumes and Gaussians under the equal storage budgets to show that Gaussians are a suitable substitute to replace curves at the finest level.

4.2 Modeling the Appearance of Gaussians

Having represented each curve as a set of Gaussians and defined an ellipsoid primitive to represent each Gaussian, we use the Bi-directional Curve Scattering Function (BCSDF) model proposed by Montazeri et al. [2020] to model the appearance of Gaussian flyaways. We chose this model for its adaptability to surfaces beyond ply curves including yarns and fabric meshes [Khatter et al. 2026a,b, 2025]. Each Gaussian undergoes both reflection and transmission and the advantage of this appearance model is that they can be separated into two independent terms. In Mitsuba 3 [Jakob et al. 2022], Gaussians are implemented as geometric primitives, allowing us to explicitly compute ray intersections with the Gaussian’s entry and exit points. Thus, similar to curves, we treat gaussians as a surface primitive with an entry and exit point.

At the entry point x the reflected component is a combination of specular reflection f_x^S , expressed as a rough dielectric and backward multiple diffuse scattering, represented using Lommel-Seeliger (LS) [Wann Jensen et al. 2023]. At the exit point y , the transmitted component is a combination of specular transmission f_y^S , expressed as a rough dielectric and forward diffuse scattering, expressed as Lambertian diffuse. The complete appearance model is given by,

$$f(\omega_i, \omega_o) = \begin{cases} f_x^S(\omega_i, \omega_o) + r_d B \text{LS}(\omega_i, \omega_o) k_x^B & (\omega_i \cdot \omega_o > 0) \\ \tau(x, y) \left(f_y^S(\omega_i, \omega_o) + (1 - r_d) \frac{B}{\pi} k_y^B \right) & (\omega_i \cdot \omega_o < 0), \end{cases} \quad (3)$$

where B is the energy assigned to the light that undergoes multiple scattering, r_d is the reflection portion of the body energy, $\tau(x, y)$ is the attenuation factor for the rays traversing through the fiber volume from x to y and k_x^B and k_y^B are the diffuse reflection and transmission color, respectively.

4.3 Curvature-Aware Ray Marching against Gaussians

Generating Gaussian flyaways all over the fabric surface greatly increases storage requirements and requires costly regeneration whenever the silhouette or fuzziness changes. A tileable patch of Gaussians is on the other hand more flexible, but requires a displacement mapping strategy that supports 3D volumetric textures, updates the mesh silhouette, and avoids precomputation.

Shell maps [Jeschke et al. 2007; Porumbescu et al. 2005] require explicit construction of a tetrahedral offset surface, reintroducing storage overhead. Prism-free methods [Chen and Chang 2008; Oliveira and Policarpo 2005] avoid explicit offset surfaces but are limited to 2D height maps and require precomputed curvature and normal maps, both incompatible with our tileable representation. Critically, neither supports 3D volumetric textures necessary for Gaussian flyaways. We therefore propose a curvature-aware inverse displacement mapping that requires only local mesh curvature at runtime. Without curvature awareness, rays marching through texture space would travel in straight lines, diverging from curved surfaces and producing incorrect silhouettes.

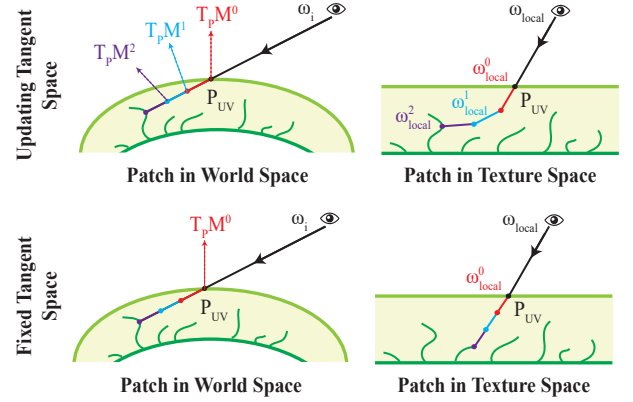


Fig. 4. **Our new curvature-aware inverse displacement mapping method.** We use the local curvature of a surface patch to update the local space, $T_P M^k$ with every marching step enabling the ray in texture space, ω_{local}^k to bend. If the ray does not intersect anything (bending away from the surface), then we mark it as no-hit which allows updating the mesh silhouette. State-of-the-art inverse displacement methods assume a fixed tangent space leading to incorrect intersections and missing silhouette for curved surfaces.

Taking inspiration from the Curvature-Aware Parallax Mapping (CAPM) [Khatter et al. 2026a] and the quadric-based silhouette rendering of Oliveira and Policarpo [Oliveira and Policarpo 2005], we propose a curvature-aware inverse displacement mapping requiring no precomputed curvature or normal maps. Unlike Oliveira and Policarpo, who fix the quadric at the fragment entry point, our key insight is to incrementally update the ray’s local frame at each marching step using curvature coefficients derived at runtime from the Weingarten map, causing the ray to continuously bend along the surface and naturally enabling silhouette updates when it diverges.

When an incoming ray, ω_i intersects the mesh at a point with position P_{uv} and normal N_{uv} , the partial derivatives w.r.t. U and V ($\frac{dP_{uv}}{du}$, $\frac{dP_{uv}}{dv}$, $\frac{dN_{uv}}{du}$, $\frac{dN_{uv}}{dv}$) describe how the surface normal varies with local curvature. These derivatives form a 2D matrix known as the shape operator (Weingarten Map) [Thorpe 1979]. The shape operator provides the curvature coefficients, κ_1 , κ_2 along the directions, $\frac{dP_{uv}}{du}$, $\frac{dP_{uv}}{dv}$ for a local surface patch. Using these curvature coefficients, we approximate the patch with a quadric surface, $z = \frac{1}{2}(\kappa_1 x^2 + \kappa_2 y^2)$, defined in the local space, $T_P M = \{\frac{dP_{uv}}{du}, \frac{dP_{uv}}{dv}, N_{uv}\}$. During ray marching at each step k , we offset the current position, P_{uv}^k by small factors Δu and Δv along $\omega_i^{T_P M^k}$. These offsets, together with the quadric approximation, are used to compute an updated local space T_P^{k+1} into which the incident ray is transformed. Changing the local space causes the ray to bend, this bending determines whether it intersects the 3D Gaussian primitives or not, enabling updates to the mesh silhouette.

Our idea is illustrated in Figure 4 and more details can be found in Section 6.1.

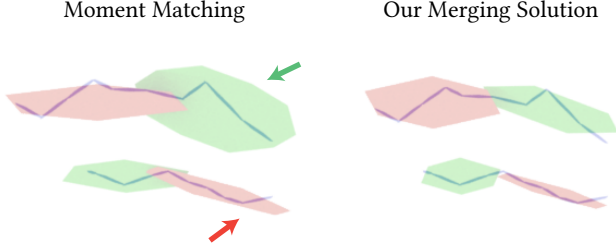


Fig. 5. **Comparing moment matching merging algorithm against ours.** We visualize two Gaussian flyaways, each composed of nine Gaussian ellipsoids rendered as ellipsoid meshes, which are merged into a two Gaussians per flyaway. Since the Gaussians do not overlap and have varying orientations, moment matching leads to variance explosion (as shown with arrows for overlapping clusters). In contrast, our merging approach enforces tighter bounds on the merged distribution by using the true bounds of the Gaussians and avoids this issue.

5 Multi-Scale Modeling and Appearance of Gaussians

The geometry and shading model described in Section 4, is suitable for near-field renderings when the camera is close enough that individual flyaways are thicker than the pixel footprint. As the camera is moved further away, the pixel footprint increases, covering multiple fibers or multiple patches increasing the variance of the normals of flyaways, requiring increased samples per pixel (SPP) impacting the total render time. To address this, we introduce a multi-scale geometry and appearance inspired by Kerbl et al. [2024]. Unlike this previous work, we construct the hierarchy using a bottom-up merging strategy, combining both geometric and appearance properties at each level, rather than refitting with a reduced budget of Gaussians for each level.

We first describe how multiple Gaussians are decimated and how we construct a texture pyramid similar to mip-map [Ewens et al. 1998; Williams 1983a] (Section 5.1). During this bottom-up aggregation, we combine geometric properties and precompute auxiliary quantities required for appearance evaluation, such as the area ratio between consecutive levels. At runtime, the corresponding level of the mip-map pyramid is selected with a heuristic and the roughness and opacity parameters of the aggregated Gaussians at that level are updated using the precomputed auxiliary data (Section 5.2).

5.1 Gaussian Decimation

For N Gaussians with varying means and covariances, there is extensive literature on merging them together [Goldberger and Roweis 2004; Jakob et al. 2011]. These merging algorithms primarily focus on minimizing the Kullback-Leibler-Divergence [Csiszár 1975] which is well suited for when Gaussians are close i.e. when the distance between their means is comparable to the standard deviation or otherwise geometrically when the area of overlap between the Gaussians is $\geq 50\%$. However, this assumption does not hold true for Gaussian flyaways. As shown in Figure 5, classical moment-matching methods tend to overemphasize overlap and inflate the

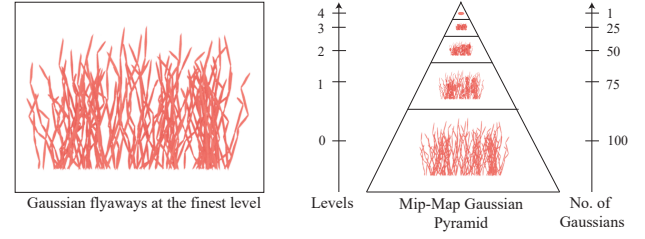


Fig. 6. **Mip-map pyramid.** We visualize a mipmap-style pyramid constructed for a small patch of flyaways represented by Gaussians. As the pyramid level increases, the number of Gaussians is progressively reduced. At the finest level, each ordered pair of points along a curve is represented by an individual Gaussian, while at the coarsest level, a single Gaussian represents the entire patch.

covariance, failing to preserve the tight spatial extent and orientation of individual primitives. This limitation becomes particularly severe for strongly anisotropic Gaussians, which motivates our design. Thus, we propose a new decimation algorithm that takes into account the mean position, orientation and extent of all the Gaussians that are to be combined to form a single Gaussian.

Our decimation algorithm is analogous to creating an object oriented bounding box. Given a set of Gaussians to be combined, we define the mean of the merged Gaussian as the average of their centers. Each covariance matrix is decomposed into a rotation matrix and a scaling matrix. The rotation matrices encode the orientation of each Gaussian, and we approximate the orientation of the merged Gaussian by averaging these rotations. The scaling matrices are used to compute the minimum of the lower bounds and the maximum of the upper bounds across all one- σ Gaussians, which then define the new scale of the merged primitive. Figure 5 illustrates how our approach handles different configurations when merging Gaussian flyaways in comparison to the classical moment-matching algorithm. Additionally, to determine which Gaussians should be grouped at coarser levels, we apply K-means clustering. The clustering relies on a similarity measure defined by the Euclidean distance between Gaussian mean positions. This encourages grouping of spatially nearby Gaussians while remaining robust to cases where the primitives are highly anisotropic or only weakly overlapping.

More details on how Gaussians are merged can be found in Section 6.2.

5.2 Aggregated Appearance of Gaussians

Using our merging algorithm, we construct a mip-map pyramid for a patch of Gaussians where each level reduces the Gaussian count by approximately 10% by K-means clustering on mean positions, from individual per-segment Gaussians at the finest level to a single Gaussian at the coarsest (Figure 6).

Opacity and Roughness of Merged Gaussians: The Gaussian flyaways to be merged can be oriented in different directions and do not necessarily overlap, meaning that the resulting merged Gaussian can contain empty regions. Thus, the opacity of the merged Gaussian is defined as the fraction of volume occupied by the constituting Gaussian flyaways.

The merged Gaussian is obtained through geometric aggregation; however, this process also affects the parameters of the appearance model [Khattar et al. 2026b; Zhu et al. 2022]. While geometric aggregation preserves the correctness of the resulting normal distribution, it increases the intensity of specular highlights, requiring adjustments to the roughness coefficients of specular components. This behavior is expected, as aggregating multiple fibers introduces increased directional disorder, analogous to microfacet distributions, which manifests as higher effective surface roughness. To address this, we adopt LEAN Mapping [Olano and Baker 2010]. Specifically, for each clustered Gaussian, the roughness covariance matrix is updated by incorporating the variance of the normal distributions of all underlying Gaussians contained within the cluster, as shown below,

$$\Sigma = \begin{bmatrix} \frac{1}{n} \Sigma(\tilde{b}_n \cdot x^2) - (\Sigma \tilde{b}_n \cdot x)^2 & \frac{1}{n} \Sigma(\tilde{b}_n \cdot x \tilde{b}_n \cdot y) - \Sigma \tilde{b}_n \cdot x \Sigma \tilde{b}_n \cdot y \\ \frac{1}{n} \Sigma(\tilde{b}_n \cdot x \tilde{b}_n \cdot y) - \Sigma \tilde{b}_n \cdot x \Sigma \tilde{b}_n \cdot y & \frac{1}{n} \Sigma(\tilde{b}_n \cdot y^2) - (\Sigma \tilde{b}_n \cdot y)^2 \end{bmatrix}, \quad (4)$$

where $\tilde{b}_n$ is the normal of the n^{th} Gaussian, projected to the $z = 1$ plane. Finally, the updated covariance matrix is expressed as,

$$\Sigma_{rough} = \Sigma + \begin{bmatrix} \beta_x & 0 \\ 0 & \beta_y \end{bmatrix}, \quad (5)$$

for a surface with roughness coefficients β_x and β_y .

Level Selection and Interpolation Using Ray Differentials: Using ray differentials [Igehy 1999], we can determine the footprint of each ray in the texture space, also known as the texture footprint, $\mathcal{T}$. During construction of the texture pyramid, we record the average projected area of the Gaussian kernels at each level along the surface normal, $A_{levels}[i]$. At render time, the appropriate mip-map level is then selected by comparing the ray’s texture footprint against these recorded areas, as expressed below,

$$L = \arg \max_i \left(\log(\mathcal{T}) - \log(A_{levels}[i]) \right). \quad (6)$$

To avoid texture popping [Heckbert 2007; Williams 1983b] and the cost of runtime Gaussian merging, we independently shade two adjacent mip-map levels and linearly blend the resulting outputs, rather than interpolating between the Gaussian representations themselves.

6 Implementation Details

Our proposed method is implemented as a custom BSDF plugin in Mitsuba 3 [Jakob et al. 2022] and we additionally update the Path Replay Backpropagation Integrator to account for the change in mesh silhouette.

6.1 Curvature-Aware Inverse Displacement Mapping

Our inverse displacement mapping approach requires no additional input texture maps or updates to the mesh geometry to estimate the coefficients of the quadric for the local surface patch.

The individual steps for our approach are summarized in Algorithm 1 and 2 in the supplementary document. We first determine the local quadric approximation of the surface patch using the UV partials and define the initial local space using $T_P M = \{ \frac{dP_{uv}}{du}, \frac{dP_{uv}}{dv} \}$ and N_{uv} . Then, for a given set of Gaussians, we define a depth function that returns the depth of a Gaussian for a given UV -value.

For each step, we offset the UV values using the depth function and estimate the resulting change in the surface normal based on the quadric approximation. This updated normal is then used to redefine the local space, enabling us to bend the incoming ray according to the local curvature. The process is repeated until the ray intersects either the underlying texture or exits the surface. If no intersection occurs, the ray is allowed to pass through the surface, altering its silhouette.

In practice, the incident rays are intersected with the base surface once and our approach takes at most a thousand steps to converge without repeating intersections.

6.2 Merging Two Gaussians

Our merging algorithm allows creation of a tight bounding Gaussian using the mean position, μ and covariance matrix, Σ for all Gaussians that have been selected to be merged at a coarser level.

We consider the case of merging two Gaussians. Consider G_1 and G_2 with means and covariances, μ_1, μ_2 and Σ_1, Σ_2 respectively. Following the classical moment matching solution, the merged Gaussian G' can be expressed with means and covariances, μ and Σ ,

$$\begin{aligned} \mu &= \frac{1}{2}(\mu_1 + \mu_2) \\ \Sigma &= \frac{1}{2}(\Sigma_1 + (\mu_1 - \mu) \cdot (\mu_1 - \mu)^T) + \\ &\quad \frac{1}{2}(\Sigma_2 + (\mu_2 - \mu) \cdot (\mu_2 - \mu)^T). \end{aligned} \quad (7)$$

If G_1 and G_2 do not overlap and are oriented in different directions then new scale matrix of G' will be incorrect, often referred to as variance explosion. Our approach, on the other hand aims to decompose the covariance matrices and determine the new orientation and scaling factors for the merged Gaussian, G' .

The mean position of G' is same as Equation 7. However, for the new covariance matrix, we first perform eigen decomposition of the two matrices to obtain the rotation and scaling matrices such that $\Sigma_1 = R_1 S_1 S_1^T R_1^T$ and $\Sigma_2 = R_2 S_2 S_2^T R_2^T$. The orientation of G' can be determined by converting R_1 and R_2 to quaternions, q_1 and q_2 which can be averaged to obtain the new rotation matrix, R from the averaged quaternion. The scaling matrices S_1 and S_2 give the 1-sigma bounds for two Gaussians that can be used to determine the new bounds for G' by taking the minimum of the minima and the maximum of the maxima to get the new scaling matrix S . Using R and S we can find the new covariance matrix to obtain the merged G' that tightly envelops the two Gaussians completely. While we described the merge operator for two Gaussians for illustration purposes, it naturally extends to N Gaussians by computing the average of all means, averaging orientations via quaternions, and taking the component-wise minimum and maximum of the scaling matrices to define the merged covariance.

7 Results

In this section, we present our results for different settings. We compare our model against state-of-the-art ray tracing methods and real photographs. We further evaluate individual components of our approach to demonstrate its flexibility across different scenarios. All of our results are generated on an Intel Core i9 processor 14900K.

We omit comparison against rasterization-based approaches [Xu et al. 2019] as their performance and trade-offs are not directly comparable to our ray-tracing solution.

7.1 Comparison to Flyaways as Curves and Volumes

In Figure 7, we compare our method against curve-based (used as reference) and volume-based flyaway representation generated along a fabric surface under environment and point lighting. For curve-based representation, thousands of flyaways are modeled as individual curves and for volume-based representation, the flyaways are represented inside a voxel grid that envelops the base surface as a participating medium. Our model relies only on a small patch of Gaussian flyaways that are tiled over the fabric surface. This makes a direct quantitative comparison with the reference challenging, as flyaways reproduced over the fabric surface and using a single tile cannot be an exact match. Nevertheless, we evaluate our method’s ability to capture appearance attributes such as sheen, fuzziness, and fabric wear in Figure 1 and 7, and the supplementary material.

In far-view, our multi-scale model accounts for the sheen effect along the fabric surface due to flyaways protruding outwards and in close-up, the individual strands of flyaways become visible showcasing the subtle change in silhouette. Curve-based representation also preserves these details across varying views but their storage and performance profile scales with the number of flyaways generated on the surface. In Figure 7 (a) and (b) there are different numbers of curves for the plain and twill weave fabrics (10000 and 30000 respectively) to show different levels of fuzziness and the varying storage requirements and render times across the two. Volume-based representation, maintain a similar performance and storage constraints as the size of the voxel grid remains fixed (1000^3 for the generated results). However, the random orientation and thin structure of flyaways require high-resolution voxel grids, increasing storage to several gigabytes. As shown in Figure 7(a), sparse flyaways are often missed in the volume representation, producing a blocky, cloudy silhouette. With dense flyaways in Figure 7(b), multiple flyaway strands collapse into single voxels without proper aggregation, weakening the sheen. Our tile-able method addresses these drawbacks while providing a subtle silhouette across the fabric edge and a fuzzy appearance with strong sheen in close-ups.

7.2 Comparison to Photos and Surface-Based Models

We compare our method against a real photograph in Figure 8 to demonstrate that representing flyaways using tile-able Gaussians produces a sheen effect that more closely resembles real fabrics than the surface-based approximation proposed by Zhu et al. [2024]. Furthermore, our hybrid approach is compatible with any surface-based representation and enables silhouette updates by creating small bumps along the fabric. In contrast, Zhu et al. [2024] preserves the fabric silhouette and incurs large precomputation and storage costs, in the order of gigabytes due to the use of Anisotropic Spherical Gaussians (ASGs) and Summed Area Tables (SATs). Our method, by comparison, relies only on a tile-able 3D texture of Gaussian flyaways which requires less than one megabyte, resulting in significantly lower overhead.

7.3 Multi-Scale Evaluation

In Figure 1 and 9, we evaluate the efficiency of our multi-scale model by comparing it against the near-field model in Equal Samples (EQ) and Equal Time (ET) render configurations. In far-view renderings, our model is 2 times faster than the near-field approach to converge to the same result. As defined in Equation 6, an appropriate level is selected from the mipmap Gaussian pyramid, and we perform interpolation at the shader level (avoiding runtime Gaussian merging) between the chosen level and the immediately finer level to prevent texture popping and ensure smooth transitions with changing views. In close-up views, the multi-scale produces the same result as the near-field model but shading and interpolation between two intermediate levels introduces additional overhead compared to the near-field approach.

Please refer to the supplementary video to observe the seamless transition across changing views.

7.4 Edit-ability

In Figure 10, we demonstrate the flexibility of our model in allowing control over the density and depth of flyaways. Unlike state-of-the-art methods, which require costly regeneration of curves or volumetric representations, our approach enables density adjustment by simply varying the number of repeated Gaussian flyaway tiles in the 3D texture space. By changing the *UV*-offsetting parameter, *c* we can efficiently control the depth of Gaussian flyaways while using the same tile. Moreover, the configuration of flyaways (from looping to sticking out) can be efficiently updated by generating a new patch comprising only a few hundred Gaussians. This enables our approach to significantly reduce preprocessing and render time compared to previous approaches.

Please refer to the supplementary for details on how a single tile is sufficient for achieving a fuzzy silhouette and appearance.

8 Discussion and Conclusion

Concavities and High Curvature. As an inverse displacement approach, our method inherits known issues such as shearing, warping, and self-intersections that are exacerbated in regions of high curvature or concave geometry. Integrating adaptive shell constructions [Wang et al. 2023] by explicit creation of a tetrahedral shell or adaptively updating the curvature coefficients into our framework can address high-curvature artifacts.

Merging Strategies. The current hierarchical merging strategy clusters Gaussians based solely on spatial proximity, which can over-blur visually distinct features by ignoring differences in orientation, anisotropy, or appearance. Incorporating more expressive distance metrics that account for these factors could yield more faithful merges while preserving efficient hierarchical evaluation.

Conclusions. We presented a multi-scale Gaussian model for fabric flyaways stored in a tileable 3D texture, combining curvature-aware inverse displacement mapping with a bottom-up mipmap merging scheme. The method achieves efficient rendering of silhouette softening, fuzziness, and grazing-angle sheen across scales, while remaining compatible with existing surface- and curve-based fabric models and offering direct control over flyaway density and depth.

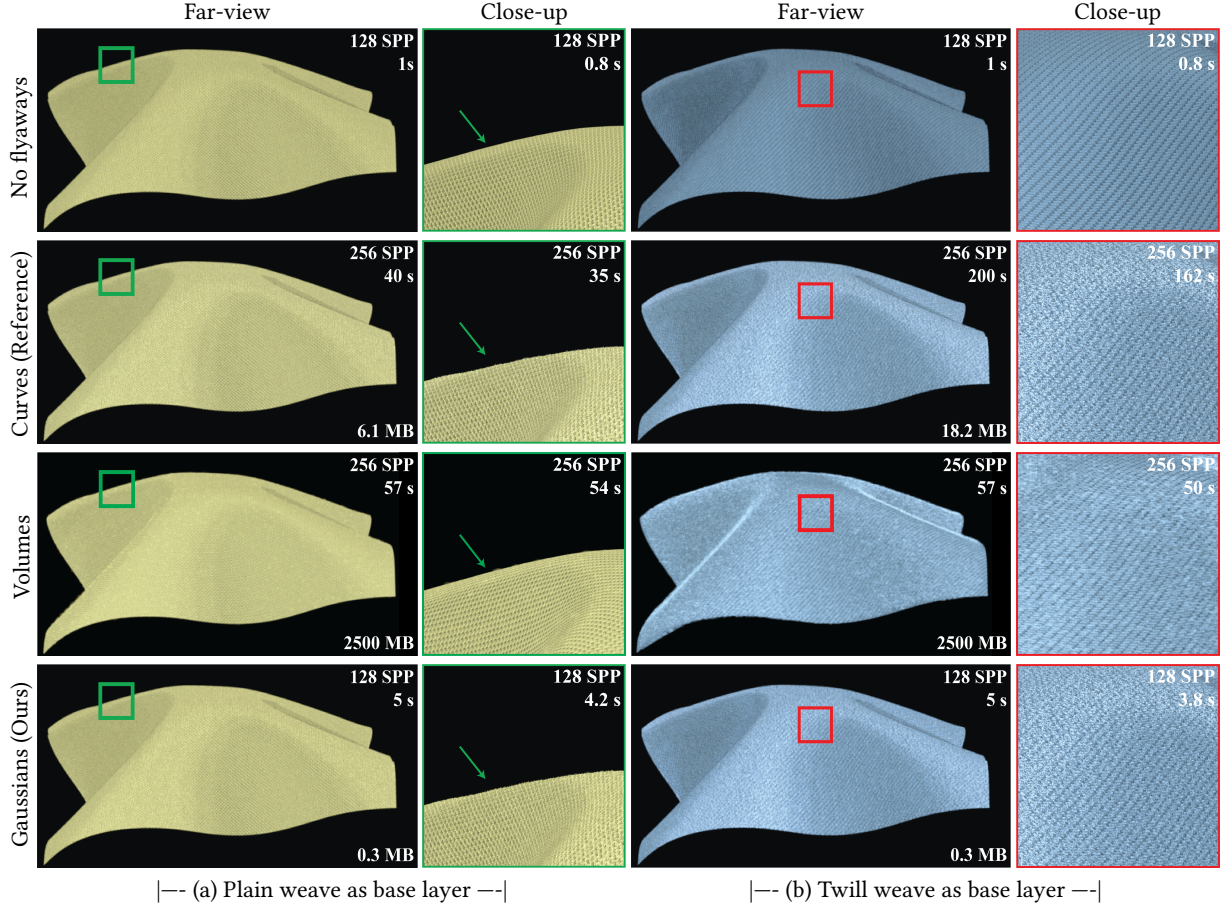


Fig. 7. **Comparison against curve-based and volume-based flyaway representations under environment and point lighting.** We render two different flyaway configurations at different zoom levels to highlight the faint change in silhouette and sheen effect (a) A plain weave fabric with small amount of fuzz that produces a faint silhouette in far-view and tiny flyaways can be seen sticking out over the fabric silhouette in close-up. (b) A twill weave fabric with a large amount of fuzz that adds imperfections and noticeable sheen to the fabric in both close-up and far-view renderings. Our multi-scale approach provides a close match to state-of-the-art representations for flyaways, while addressing their limitations in regards to render times and storage requirements. NOTE: Aggregation of flyaways into voxels leads to incorrect / missing sheen.

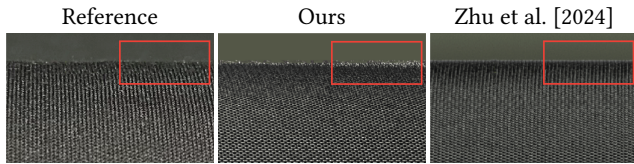


Fig. 8. **Comparison of the sheen effect with a reference photograph.** We compare our hybrid model with a real photograph, where the base layer is modeled following Khattar et al. [2025]. Our hybrid approach captures the sheen effect more realistically, and the random tiling process introduces natural variation in sheen intensity. In contrast, Zhu et al. [2024] model sheen analytically, which results in a continuous and spatially uniform sheen along the silhouette.

References

Autodesk, INC. 2019. *Maya*. <https://autodesk.com/maya>

- Ying-Chieh Chen and Chun-Fa Chang. 2008. A Prism-Free Method for Silhouette Rendering in Inverse Displacement Mapping. In *Computer Graphics Forum*, Vol. 27. Wiley Online Library, 1929–1936.
- Jorge Condor, Sebastian Speierer, Lukas Bode, Aljaz Bozic, Simon Green, Piotr Didyk, and Adrian Jarabo. 2025. Don't Splat your Gaussians: Volumetric Ray-Traced Primitives for Modeling and Rendering Scattering and Emissive Media. *ACM Trans. Graph.* 44, 1, Article 10 (Feb. 2025), 17 pages. doi:10.1145/3711853
- Imre Csiszár. 1975. I-divergence geometry of probability distributions and minimization problems. *The annals of probability* (1975), 146–158.
- Jon P Ewins, Marcus D Waller, Martin White, and Paul F Lister. 1998. Mip-map level selection for texture mapping. *IEEE Transactions on Visualization and Computer Graphics* 4, 4 (1998), 317–329.
- Jacob Goldberger and Sam Roweis. 2004. Hierarchical clustering of a mixture model. *Advances in neural information processing systems* 17 (2004).
- Paul S Heckbert. 2007. Survey of texture mapping. *IEEE computer graphics and applications* 6, 11 (2007), 56–67.
- Homan Igehy. 1999. Tracing ray differentials. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 179–186.
- Piti Irawan and Steve Marschner. 2012. Specular reflection from woven cloth. *ACM Trans. Graph.* 31, 1, Article 11 (Feb. 2012), 20 pages. doi:10.1145/2077341.2077352
- Wenzel Jakob, Christian Regg, and Wojciech Jarosz. 2011. Progressive expectation-maximization for hierarchical volumetric photon mapping. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1287–1297.

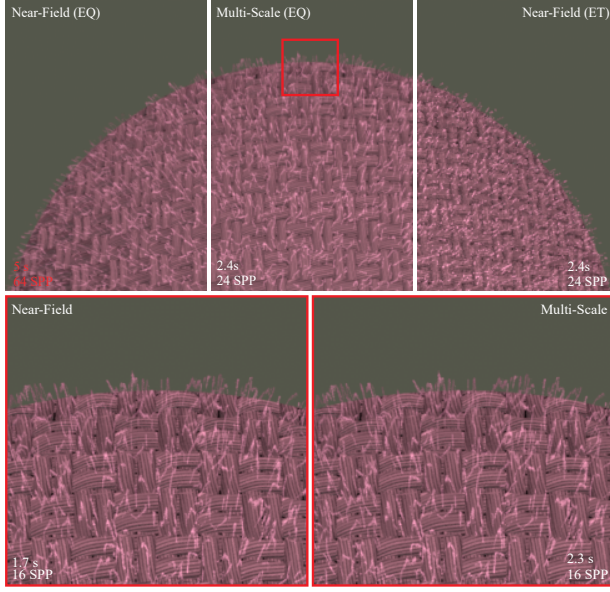


Fig. 9. **Multi-scale comparison with the near-field model for equal quality and equal time.** We compare our multi-scale mip-map Gaussian flyaways to the near-field model. In far view, the multi-scale reduces the variance by selecting a suitable level, reducing the number of ray samples required. Since we interpolate between the two consecutive levels at shading time, our model adds a small overhead. Please refer to the supplementary video for a seamless transition from close-up to far-view.

- Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, Merlin Nimier-David, Delio Vicini, Tizian Zeltner, Baptiste Nicolet, Miguel Crespo, Vincent Leroy, and Ziyi Zhang. 2022. *Mitsuba 3 renderer*. <https://mitsuba-renderer.org>.
- Stefan Jeschke, Stephan Mantler, and Michael Wimmer. 2007. Interactive smooth and curved shell mapping. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*. 351–360.
- Wenhua Jin, Beibei Wang, Milos Hasan, Yu Guo, Steve Marschner, and Ling-Qi Yan. 2022. Woven Fabric Capture from a Single Photo. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) (SA '22). Association for Computing Machinery, New York, NY, USA, Article 33, 8 pages. doi:10.1145/3550469.3555380
- Bernhard Kerbl, Georgios Kopanas, Thomas Leimkuehler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Trans. Graph.* 42, 4, Article 139 (July 2023), 14 pages. doi:10.1145/3592433
- Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. 2024. A Hierarchical 3D Gaussian Representation for Real-Time Rendering of Very Large Datasets. *ACM Trans. Graph.* 43, 4, Article 62 (July 2024), 15 pages. doi:10.1145/3658160
- Apoorv Khattar, Jean-Marie Aubry, Ling-Qi Yan, and Zahra Montazeri. 2026a. A Texture-Free Multi-Scale Model for Surface-Based Rendering of Knitted Fabrics. In *Computer Graphics Forum*. Wiley Online Library, e70356.
- Apoorv Khattar, Junqiu Zhu, Jean-Marie Aubry, Emiliano Padovani, Marc Droske, Ling-Qi Yan, and Zahra Montazeri. 2026b. A multi-scale yarn appearance model with fiber details. *Computational Visual Media* (2026).
- Apoorv Khattar, Junqiu Zhu, Ling-Qi Yan, and Zahra Montazeri. 2025. A Texture-Free Practical Model for Realistic Surface-Based Rendering of Woven Fabrics. In *Computer Graphics Forum*, Vol. 44. Wiley Online Library, e15283.
- Pramook Khungurn, Daniel Schroeder, Shuang Zhao, Kavita Bala, and Steve Marschner. 2016. Matching Real Fabrics with Micro-Appearance Models. *ACM Trans. Graph.* 35, 1, Article 1 (Dec. 2016), 26 pages. doi:10.1145/2818648
- Yang Liu, Chuanchen Luo, Lue Fan, Naiyan Wang, Junran Peng, and Zhaoxiang Zhang. 2025. Citygaussian: Real-time high-quality large-scale scene rendering with gaussians. In *European Conference on Computer Vision*. Springer, 265–282.
- Nicolas Moenne-Loccoz, Ashkan Mirzaei, Or Perel, Riccardo de Lutio, Janick Martinez Esturo, Gavriel State, Sanja Fidler, Nicholas Sharp, and Zan Gojcic. 2024. 3D Gaussian Ray Tracing: Fast Tracing of Particle Scenes. *ACM Trans. Graph.* 43, 6, Article 232 (Nov. 2024), 19 pages. doi:10.1145/3687934

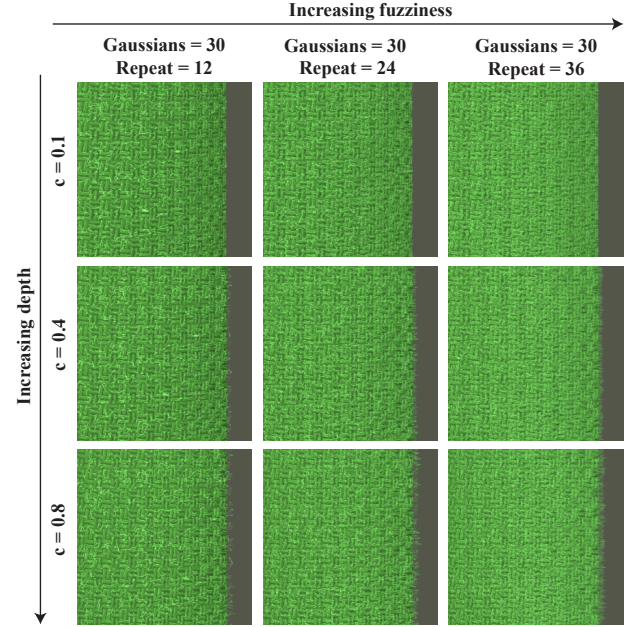


Fig. 10. **Customizing flyaway properties.** We demonstrate the editability of our method, which enables control over the number and height of flyaways by adjusting the repeat count of the tile-able Gaussian flyaways and modifying step size. Changing the depth causes the flyaways to protrude out around the surface silhouette and changing the fuzziness causes the merging of Gaussians to create a hazy layer on top of the fabric surface with individual flyaways hard to perceive. In contrast, modeling flyaways as curves would require regenerating a different set of curves—each with its own number and configuration—whenever changes are needed, increasing both preprocessing overhead and final rendering time.

- Zahra Montazeri, Søren Gammelmark, Henrik Wann Jensen, and Shuang Zhao. 2021. Practical Ply-Based Appearance Modeling for Knitted Fabrics. In *Eurographics Symposium on Rendering - DL-only Track*, Adrien Bousseau and Morgan McGuire (Eds.). The Eurographics Association. doi:10.2312/sr.20211297
- Zahra Montazeri, Søren B. Gammelmark, Shuang Zhao, and Henrik Wann Jensen. 2020. A practical ply-based appearance model of woven fabrics. *ACM Trans. Graph.* 39, 6, Article 251 (Nov. 2020), 13 pages. doi:10.1145/3414685.3417777
- Marc Olano and Dan Baker. 2010. Lean mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*. 181–188.
- Manuel M Oliveira and Fabio Policarpo. 2005. An efficient representation for surface details. *Instituto de Informatica UFRGS* (2005).
- Serban D. Porumbescu, Brian Budge, Louis Feng, and Kenneth I. Joy. 2005. Shell maps. *ACM Trans. Graph.* 24, 3 (July 2005), 626–633. doi:10.1145/1073204.1073239
- Boxiang Rong, Artur Grigorev, Wenbo Wang, Michael J. Black, Bernhard Thomaszewski, Christina Tsalicoglou, and Otmar Hilliges. 2025. Gaussian Garments: Reconstructing Simulation-Ready Clothing with Photo-Realistic Appearance from Multi-View Video. In *International Conference on 3D Vision (3DV)*. <https://ribosome-rbx.github.io/Gaussian-Garments/>
- Iman Sadeghi, Oleg Bisker, Joachim De Deken, and Henrik Wann Jensen. 2013. A practical microcylinder appearance model for cloth rendering. *ACM Trans. Graph.* 32, 2, Article 14 (April 2013), 12 pages. doi:10.1145/2451236.2451240
- JA Thorpe. 1979. The Weingarten Map. In *Elementary Topics in Differential Geometry*. Springer, 53–61.
- Wenping Wang, Bert Jüttler, Dayue Zheng, and Yang Liu. 2008. Computation of rotation minimizing frames. *ACM Transactions on Graphics (TOG)* 27, 1 (2008), 1–18.
- Zian Wang, Tianchang Shen, Merlin Nimier-David, Nicholas Sharp, Jun Gao, Alexander Keller, Sanja Fidler, Thomas Müller, and Zan Gojcic. 2023. Adaptive Shells for Efficient Neural Radiance Field Rendering. *ACM Trans. Graph.* 42, 6, Article 259 (2023), 15 pages. doi:10.1145/3618390
- Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. 2023. *A Practical Model for Subsurface Light Transport* (1 ed.). Association for Computing

1027	Machinery, New York, NY, USA. https://doi.org/10.1145/3596711.3596747	1084
1028	Lance Williams. 1983a. Pyramidal parametrics. <i>SIGGRAPH Comput. Graph.</i> 17, 3 (July 1983), 1–11. doi:10.1145/964967.801126	1085
1029	Lance Williams. 1983b. Pyramidal parametrics. In <i>Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques</i> (Detroit, Michigan, USA) (<i>SIGGRAPH '83</i>). Association for Computing Machinery, New York, NY, USA, 1–11. doi:10.1145/800059.801126	1086
1030	Qi Wu, Janick Martinez Esturo, Ashkan Mirzaei, Nicolas Moënne-Loccoz, and Zan Gojcic. 2025. 3DGUT: Enabling Distorted Cameras and Secondary Rays in Gaussian Splatting. In <i>Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)</i> . 26036–26046.	1087
1031	Chao Xu, Rui Wang, Shuang Zhao, and Hujun Bao. 2019. Multi-scale hybrid micro-appearance modeling and realtime rendering of thin fabrics. <i>IEEE transactions on visualization and computer graphics</i> 27, 4 (2019), 2409–2420.	1088
1032	Egor Zakharov, Vanessa Sklyarova, Michael J Black, Giljoo Nam, Justus Thies, and Otmar Hilliges. 2024. Human Hair Reconstruction with Strand-Aligned 3D Gaussians. In <i>European Conference on Computer Vision (ECCV 2024) (LNCS)</i> . Springer Cham.	1089
1033	Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. 2012. Structure-aware synthesis for predictive woven fabric appearance. <i>ACM Trans. Graph.</i> 31, 4, Article 75 (July 2012), 10 pages. doi:10.1145/2185520.2185571	1090
1034	Shuang Zhao, Wenzel Jakob, Steve Marschner, and Kavita Bala. 2014. Building volumetric appearance models of fabric using micro CT imaging. <i>Commun. ACM</i> 57, 11 (Oct. 2014), 98–105. doi:10.1145/2670517	1091
1035	Junqiu Zhu, Christophe Hery, Lukas Bode, Carlos Aliaga, Adrian Jarabo, Ling-Qi Yan, and Matt Jen-Yuan Chiang. 2024. A Realistic Multi-scale Surface-based Cloth Appearance Model. In <i>ACM SIGGRAPH 2024 Conference Papers</i> (Denver, CO, USA) (<i>SIGGRAPH '24</i>). Association for Computing Machinery, New York, NY, USA, Article 89, 10 pages. doi:10.1145/3641519.3657426	1092
1036	Junqiu Zhu, Adrian Jarabo, Carlos Aliaga, Ling-Qi Yan, and Matt Jen-Yuan Chiang. 2023a. A Realistic Surface-based Cloth Rendering Model. In <i>ACM SIGGRAPH 2023 Conference Proceedings</i> (Los Angeles, CA, USA) (<i>SIGGRAPH '23</i>). Association for Computing Machinery, New York, NY, USA, Article 5, 9 pages. doi:10.1145/3588432.3591554	1093
1037	Junqiu Zhu, Zahra Montazeri, J Aubry, Lingqi Yan, and Andrea Weidlich. 2023b. A Practical and Hierarchical Yarn-based Shading Model for Cloth. In <i>Computer Graphics Forum</i> , Vol. 42. Wiley Online Library, e14894.	1094
1038	Junqiu Zhu, Sizhe Zhao, Lu Wang, Yanning Xu, and Ling-Qi Yan. 2022. Practical level-of-detail aggregation of fur appearance. <i>ACM Trans. Graph.</i> 41, 4, Article 47 (July 2022), 17 pages. doi:10.1145/3528223.3530105	1095
1039		1096
1040		1097
1041		1098
1042		1099
1043		1100
1044		1101
1045		1102
1046		1103
1047		1104
1048		1105
1049		1106
1050		1107
1051		1108
1052		1109
1053		1110
1054		1111
1055		1112
1056		1113
1057		1114
1058		1115
1059		1116
1060		1117
1061		1118
1062		1119
1063		1120
1064		1121
1065		1122
1066		1123
1067		1124
1068		1125
1069		1126
1070		1127
1071		1128
1072		1129
1073		1130
1074		1131
1075		1132
1076		1133
1077		1134
1078		1135
1079		1136
1080		1137
1081		1138
1082		1139
1083		1140