Auto Hair Card Extraction for Smooth Hair with Differentiable Rendering

ZHONGTIAN ZHENG, LIGHTSPEED, China
TAO HUANG, LIGHTSPEED, USA
HAOZHE SU, LIGHTSPEED, USA
XUEQI MA, LIGHTSPEED, China
YUEFAN SHEN, LIGHTSPEED, China
TONGTONG WANG, LIGHTSPEED, China
YIN YANG, University of Utah, USA
XIFENG GAO, LIGHTSPEED, USA
ZHERONG PAN, LIGHTSPEED, USA
KUI WU, LIGHTSPEED, USA

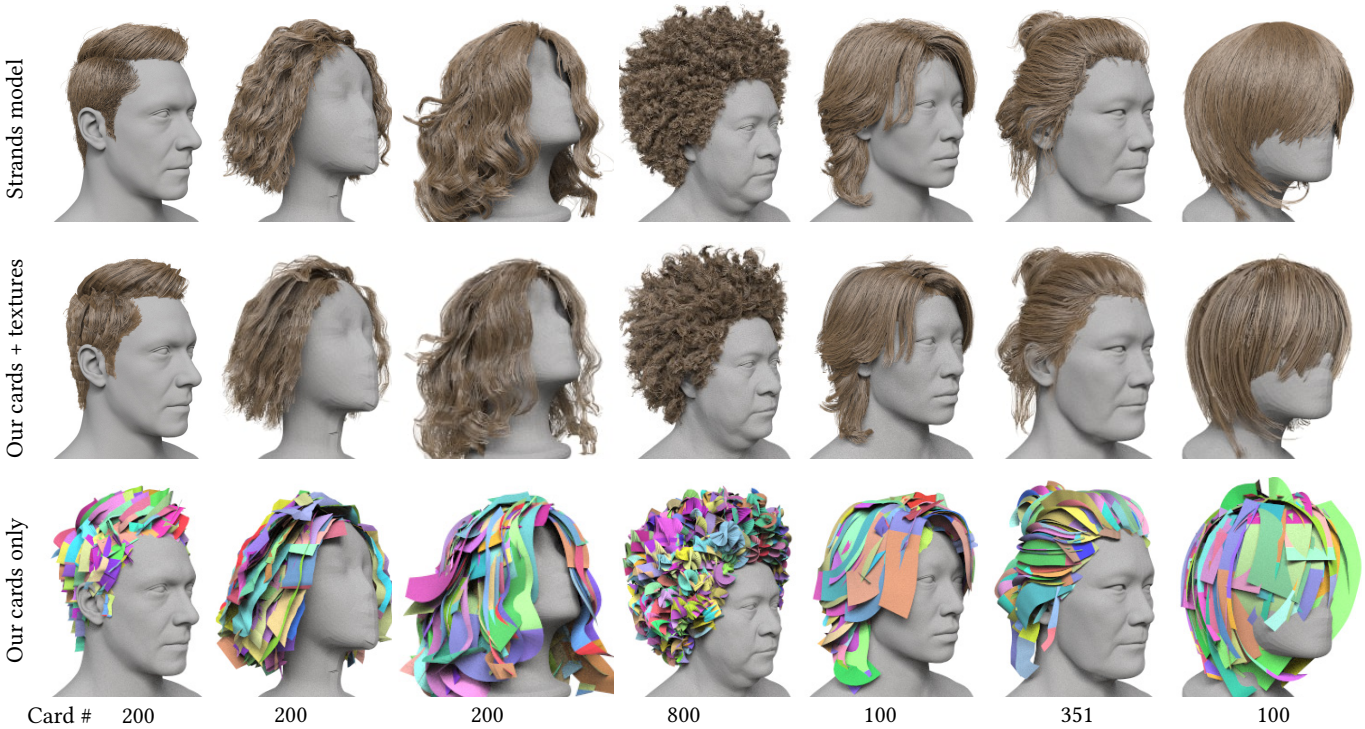

Fig. 1. *Our automatic pipeline can convert a wide variety of strand-based hairstyles with 40K strands into hair card models using a small number of cards (bottom, depending on hairstyles) and 32 individual card textures, while preserving high visual fidelity. Here, we show the input strand model, our generated cards only, and cards rendered with textures.*

Authors' addresses: Zhongtian Zheng, zhengzhongtian@pku.edu.cn, LIGHTSPEED, Shenzhen, Guangzhou, China; Tao Huang, tao_huang@ucsb.edu, LIGHTSPEED, Los Angeles, CA, USA; Haozhe Su, dyhard0520@gmail.com, LIGHTSPEED, Los Angeles, CA, USA; Xueqi Ma, qixuemaa@gmail.com, LIGHTSPEED, Shenzhen, Guangzhou, China; Yuefan Shen, yuefanshen@outlook.com, LIGHTSPEED, Shenzhen, Guangzhou, China; Tongtong Wang, wangtong923@gmail.com, LIGHTSPEED, Shenzhen, Guangzhou, China; Yin Yang, yangzzzy@gmail.com, University of Utah, Salt Lake City, UT, USA; Xifeng Gao, gxfxisha@gmail.com, LIGHTSPEED, Seattle, WA, USA; Zherong Pan, zherong.pan.usa@gmail.edu, LIGHTSPEED, Seattle, WA, USA; Kui Wu, walker.kui.wu@gmail.com, LIGHTSPEED, Los Angeles, CA, USA.

ACM 0730-0301/2025/12-ART237
https://doi.org/10.1145/3763295

Hair cards remain a widely used representation for hair modeling in real-time applications, offering a practical trade-off between visual fidelity, memory usage, and performance. However, generating high-quality hair card models remains a challenging and labor-intensive task. This work presents an automated pipeline for converting strand-based hair models into hair card models with a limited number of cards and textures while preserving the hairstyle appearance. Our key idea is a novel differentiable representation

where each strand is encoded as a projected 2D curve in the texture space, which enables end-to-end optimization with differentiable rendering while respecting the structures of the hair geometry. Based on this representation, we develop a novel algorithm pipeline, where we first cluster hair strands into initial hair cards and project the strands into the texture space. We then conduct a two-stage optimization, where our first stage optimizes the orientation of each hair card separately, and after strand projection, our second stage conducts joint optimization over the entire hair card model for fine-tuning. Our method is evaluated on a range of hairstyles, including straight, wavy, curly, and coily hair. To capture the appearance of short or coily hair, our method comes with support for hair caps and cross-card.

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

Additional Key Words and Phrases: Hair; hair card; differentiable rendering

ACM Reference Format:
Zhongtian Zheng, Tao Huang, Haozhe Su, Xueqi Ma, Yuefan Shen, Tongtong Wang, Yin Yang, Xifeng Gao, Zherong Pan, and Kui Wu. 2025. Auto Hair Card Extraction for Smooth Hair with Differentiable Rendering. *ACM Trans. Graph.* 44, 6, Article 237 (December 2025), 12 pages. https://doi.org/10.1145/3763295

1 INTRODUCTION

Hair plays a significant role in creating believable characters and immersive environments in films, video games, and virtual reality. However, the average human scalp has approximately 100,000 to 150,000 hair follicles. Although strand-based hair simulation and rendering have become increasingly popular in recent production pipelines due to their ability to realistically capture dynamic behavior and visual richness [Epic Games 2021; Hsu et al. 2023, 2024, 2025; Huang et al. 2023; Tafuri 2019], hair cards remain a widely adopted technique in real-time applications [Jiang 2016]. Hair cards are flat, textured quad strips that approximate the visual appearance of hair clusters. As illustrated in Fig. 1, hair cards can effectively reproduce the intricate structure of complex hair styles. Beyond their role in representing high-fidelity hair with a large number of textured planes [Jiang 2016], hair card representations are also widely used in level-of-detail (LoD), where strand-based models serve as a high-resolution representation, while hair cards can be used as a low-resolution approximation to significantly reduce rendering costs when hair is viewed from a distance. Additionally, mobile games and other low-performance platforms still rely heavily on card-based representations due to strict hardware constraints. Even on high-performance PC platforms, strand-based hair is typically reserved for main characters, while NPCs and background characters use hair cards to reduce runtime costs.

Unfortunately, the industry currently heavily relies on crafting hairstyles manually using hair cards. In this workflow, artists first create a set of hair card textures that contain varying numbers of strands and degrees of curvature. They then manually extrude each hair card from the scalp outward, following the hair flow while varying its length, width, and orientation to achieve a realistic appearance. Apparently, the fidelity quality of the hair card model is primarily bound by the number of cards and textures allowed. As a result, manually crafting a low-resolution hair card model, with a limited number of quads and card textures while maintaining high visual fidelity to the reference strand-based model, is a challenging and labor-intensive task, typically requiring several days to weeks of work from an experienced artist.

Most recent research works have focused on reconstructing strand-based hair models from single-view images [Wu et al. 2022; Zhang and Zheng 2019; Zheng et al. 2023; Zhou et al. 2018], multi-view inputs [Kuang et al. 2022; Wu et al. 2024b; Zhou et al. 2024], and CT scans [Shen et al. 2023]. However, there is a lack of work on converting these reconstructed strands into hair card representations suitable for real-time applications. Extracting hair cards from strand-based hair models presents several challenges. First, there is a strict budget on both the number of hair cards and on the size of texture, imposed by memory and computational efficiency constraints. Therefore, the extraction process must carefully balance visual fidelity with performance. Second, the hairstyle is highly irregular and chaotic, exhibiting significant variation in strand length, shape, and structure, making it challenging to generate a simplified representation while preserving visual details. Recently, Unreal Engine (UE) [Epic Games 2021] introduced an automatic hair card generation tool from strand-based models [Epic Games 2025]. However, the quality of its generated results remains insufficient for production use, often failing to preserve the appearance of the target hair strand model when the given hair card target is limited.

This paper presents the first automated pipeline that converts straight and wavy strand-based hair models into hair cards. The process begins with hair strand clustering, where strands are grouped based on a hair shape similarity metric. For each cluster, we optimize the orientation of the card geometry to minimize the strand projection error. To improve memory efficiency and runtime performance, we perform second-stage clustering on the hair textures, enabling texture sharing across multiple cards. Finally, we jointly optimize the hair card positions, strand shapes, and hair widths to minimize differences in tangent, depth, and coverage between our output cards and the original strand-based model. To this end, conventional RGB-image texture representations can lead to significant aliasing artifacts. Instead, we propose an explicit hair card representation in which each strand is projected into texture space as a 2D curve. These 2D curves serve as an intermediate representation that enables high-quality rendering and supports differentiable optimization. To enhance the visual fidelity for short and coily hairstyles, which is challenging for original hair card representation, our framework also supports baking a hair cap texture and generating a pair of crossed cards per hair cluster, enriching the volumetric appearance of the hair.

We evaluate our pipeline on a diverse range of hairstyles, including straight, wavy, curly, and coily hair, with a limited number of cards and textures. Experiments show that our automated pipeline surpasses both UE automatic solution and manual-crafted cards.

2 RELATED WORK

This section briefly reviews work on hair representation, hair modeling, and extracting planar representations from meshes.

Hair Representation. Both hair simulation and rendering are computationally expensive due to the large number of individual strands and their complex interactions. To reduce computational costs, researchers have developed various reduced representations over the years, including 2D strips (commonly referred to as *hair cards*) [Koh and Huang 2001; Ward et al. 2003], cubic lattice structures[Volino

and Magnenat-Thalmann 2006], short hair strips [Guang and Zhiyong 2002], and volumetric representations [Lee et al. 2019; Wu and Yuksel 2016]. During rendering, these reduced representations are expanded into full hair using baked textures or procedural functions. Hair cards remain widely used in the gaming industry due to their simplicity and efficiency [Jiang 2016]. For a comprehensive overview of hair rendering and simulation, we refer readers to the course by Bertails et al. [2008]. However, creating hair cards for multiple LoDs continues to be a labor-intensive process for artists, representing a significant bottleneck in production pipelines. Recently, Huang et al. [2025] presents a real-time framework to create and render hair LoD dynamically based on pre-clustered hairs.

Hair Modeling. Traditionally, artists have used tools like Maya XGen to create strand-based and card-based hair models. However, manually crafting hair remains a labor-intensive process. To simplify hair authoring, Yuksel et al. [2009] introduced *hair mesh*, a volumetric representation that provides high-level editing tools for artists. Sketch-based input methods [Fu et al. 2007; Shen et al. 2020] have also offered user-friendly interaction capability, allowing artists to design hair directly on the screen. For automatic hair modeling, researchers have explored extracting strand-based geometries directly from images. Earlier methods relied on heuristic-based approaches [Hu et al. 2017; Jakob et al. 2009; Kong and Nakajima 1998; Paris et al. 2008; Sun et al. 2021] or leveraged large 3D hair databases [Chai et al. 2016; Hu et al. 2015; Liang et al. 2018] for guidance. Recently, learning-based techniques have demonstrated accuracy and robustness in reconstructing simple hairstyles from single-view images [Wu et al. 2022; Zhang and Zheng 2019; Zheng et al. 2023; Zhou et al. 2018] and multi-view inputs [Kuang et al. 2022; Rosu et al. 2022; Sklyarova et al. 2023; Takimoto et al. 2024; Wu et al. 2024b; Zakharov et al. 2024; Zhou et al. 2024]. Wu et al. [2024a] present a geometric method to generate highly coiled hair. Beyond explicit modeling and reconstruction, hair synthesis methods enable hairstyle transfer from one to another using feature maps [Wang et al. 2009], further refined with learning-based features [Chen et al. 2024; He et al. 2025; Sklyarova et al. 2024; Zhou et al. 2023]. However, these techniques primarily aim to generate explicit strand geometries, which are computationally intensive and unsuitable for real-time applications. There are also learning-based methods using neural representations [Luo et al. 2024; Wang et al. 2023; Zheng et al. 2025]. While neural-based methods have shown significant progress in reconstructing and representing complex hair geometry, integrating these representations into industrial game engines in a highly efficient manner remains a major challenge.

Billboard Extraction. In addition to hair, arbitrarily oriented billboards (or impostors) with textures are widely used for rendering trees and forests [Behrendt et al. 2005] and large-scale scenes [Hladky et al. 2022; Lall et al. 2018]. To convert a 3D mesh to billboards, Décoret et al. [2003] proposed a greedy optimization algorithm that approximates the 3D model using a discretized plane parameterization in spherical coordinates. This method was later enhanced by Andujar et al. [2004] to produce better-fitting billboards. Silvennoinen et al. [2014] uses a similar concept to generate sets of planars as occluders. However, as hair presents unique challenges due to its extensive strand number and complex geometry, existing billboard generation methods cannot be applied directly.

3 PROBLEM STATEMENT AND OVERVIEW

In this section, we first provide the problem statement, followed by an overview of our pipeline.

Problem Statement. The input to our method is a strand-based hairstyle model $\mathcal{H} = \{\mathcal{S}_i\}$, where each strand $\mathcal{S}_i = \{\mathbf{s}_{i,j}\}$ is represented as a piecewise linear curve consisting of n^s uniformly distributed samples $\mathbf{s}_{i,j}$ (with the same distance between two consecutive samples on each strand), along with a corresponding head mesh $\mathcal{M}^{\text{head}}$. The output of our pipeline is a hair card model $\mathcal{M}^{\text{output}} = \langle \{C_i\}_{i=1}^{n^c}, \{\mathcal{T}_i\}_{i=1}^{n^t} \rangle$, composed of n^c hair cards and n^t textures to balance visual fidelity and performance constraints. In order to save texture space, users oftentimes choose $n^c \gg n^t$ such that multiple hair cards need to share the same texture. A hair card C_i refers to a quad strip with n^q consecutive quads. Our objective is to ensure that the synthesized hair card model $\mathcal{M}^{\text{output}}$ approximates the visual appearance of the input $\mathcal{H}$ as closely as possible.

Overview. As illustrated in Fig. 2, our pipeline begins with hair strand clustering (Sec. 4.1), where strands are grouped into n^c clusters based on a hair shape similarity metric. For each cluster, we then optimize the card orientation to minimize visual error, producing the initial card geometry (Sec. 4.2). Next, we introduce an explicit card texture representation by projecting the 3D curves onto a 2D texture space (Sec. 4.3). The generated hair textures are further clustered into n^t groups to facilitate texture sharing across cards (Sec. 4.4). Finally, we jointly optimize the geometry and strands of all hair cards to minimize the difference between $\mathcal{H}$ and $\mathcal{M}^{\text{output}}$ using differentiable rendering (Sec. 4.5). The optimized cards and strands are then used to generate the final textures via rasterization.

4 METHOD

We revise the steps of our pipeline in this section.

4.1 Strand Clustering

Our method begins by clustering the input strands $\mathcal{H}$ into a number of subsets so each cluster can be represented as a hair card. We use a clustering method proposed by [Wang et al. 2009]. Specifically, a hair shape similarity metric is defined as $\gamma(\mathcal{S}_i, \mathcal{S}_j) = \sum_{k=1}^{n^s} \|\mathbf{s}_{i,k} - \mathbf{s}_{j,k}\|_2^2 / n^s$ to quantify the sample-wise distance between two hair strands $\mathcal{S}_i$ and $\mathcal{S}_j$. Using this metric, we group $\mathcal{H}$ into clusters $\{\mathcal{G}_k\}$ with a k-mean clustering. Clearly, the center of each cluster is a mean strand denoted as $\bar{\mathcal{S}}_k = \{\bar{\mathbf{s}}_{k,j}\}$ with samples being $\bar{\mathbf{s}}_{k,j} = \sum_{\mathcal{S}_i \in \mathcal{G}_k} \mathbf{s}_{i,j} / |\mathcal{G}_k|$, where $\mathcal{G}_k \subseteq \mathcal{H}$ is a strand subset.

4.2 Card Geometry Initialization

Given the strand clusters $\mathcal{G}_k$, our next step is to initialize the hair card and prepare for further optimization. We first initialize an orthogonal frame along the mean strand and build initial card geometry. Then, we optimize orientation for each card to minimize projection error.

(a) Strands (b) Clustered strands (c) Optimized cards (d) Cards w/ exp. textures 19.28/0.148/0.026 (e) Texture reduction 18.54/0.161/0.032 (f) Final cards w/ textures **19.99/0.140/0.021**

Fig. 2. *Our pipeline: Given the strand-based hair model (a), we first cluster the strands based on their similarity (b). Then, we optimize the orientation for each card (c) and create the corresponding card texture with explicit hair strand representation (d). After hair texture reduction (e), we jointly optimize cards and strands to fine-tune the final result (f). ●/●/● indicate averaged PSNR ↑, LPIPS ↓, and coverage error ↓, respectively.*

Card Geometry Fitting. To initialize the card geometry, we need to build an orthogonal frame system $\{\mathbf{t}_{k,j}, \mathbf{n}_{k,j}, \mathbf{b}_{k,j}\}$ at each sample $\bar{\mathbf{s}}_{k,j}$ along the central strand $\bar{\mathcal{S}}_k$, where $\mathbf{t}_{k,j}, \mathbf{n}_{k,j}, \mathbf{b}_{k,j}$ are the tangent, normal, and binormal vectors, respectively. The problem of building a frame system for a 3D curve has been thoroughly studied, for which the standard construction is the Frenet-Serret formulas. But this formula suffers from singular configurations, and a more numerically stable choice is the Bishop formulas [Bergou et al. 2008], which have been adopted in strand-based hair simulation. Specifically, given the normal $\mathbf{n}_{k,1}$ for the first curve segment, the entire frame system can be computed by solving the Bishop formulas, and we refer readers to [Bergou et al. 2008] for the piecewise linear discretization. Given the frame system, we calculate the maximum distance from $\bar{\mathbf{s}}_{k,j}$ to all corresponding strand samples within the cluster along the binormal direction such that:

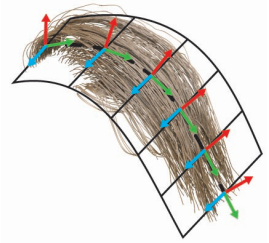

Fig. 3. *Example of hair cluster and its hair strip as well as per-sample frames.*

$$W_{k,j} = \max_{\mathcal{S}_i \in \mathcal{G}_k} \left| (\mathbf{s}_{i,j} - \bar{\mathbf{s}}_{k,j}) \cdot \mathbf{b}_{k,j} \right|. \tag{1}$$

Then, the card is constructed by connecting all $\mathbf{p}^{\pm}_{k,j} = \bar{\mathbf{s}}_{k,j} \pm W_{k,j}\mathbf{b}_{k,j}$ along the positive and negative $\mathbf{b}_{k,j}$ at each $\bar{\mathbf{s}}_{k,j}$ to form a quad strip with n^s quads. Finally, we downsample the quad strip to use only n^q quads, leading to our card geometry C_k as shown in Fig. 3.

Card Orientation Optimization. In the above discussion, we have assumed that the root normal $\mathbf{n}_{k,1}$ is given as the boundary condition of the Bishop formulas. This root normal $\mathbf{n}_{k,1}$ for the first card segment needs to be carefully computed to provide a good initial guess for further optimizations. We propose to optimize $\mathbf{n}_{k,1}$ by minimizing the difference between the strand cluster $\mathcal{G}_k$ and the hair card C_k. To this end, we introduce the projection operator $\mathbf{s}^{\Gamma}_{i,j} = \text{argmin}_{\mathbf{s} \in C_k} \|\mathbf{s} - \mathbf{s}_{i,j}\|$ as finding the closest point $\mathbf{s}$ on the quad strip of hair card C_k, that is closest to $\mathbf{s}_{i,j}$. We then define the projection error as follows:

$$L^{\text{proj}} = \sum_{\mathcal{S}_i \in \mathcal{G}_k} \sum_{\mathbf{s}_{i,j} \in \mathcal{S}_i} \|\mathbf{s}^{\Gamma}_{i,j} - \mathbf{s}_{i,j}\|. \tag{2}$$

Although the Bishop formulas are differentiable, the projection operator is non-differentiable, so gradient-based continuous optimization is non-available to minimize L^{proj}. Fortunately, the solution space is rather small. Since Eq. 2 is independent for different cards and due to the condition that $\mathbf{n}_{i,1}$ is orthogonal to $\mathbf{t}_{i,1}$, $\mathbf{n}_{i,1}$ essentially lies on a 3D circle. Therefore, we evenly sample a fixed number of potential $\mathbf{n}_{i,1}$ along the 3D circle and pick the direction leading to the smallest value of Eq. 2. At this point, we have initialized all the card geometries.

4.3 Explicit Hair Card Texture

In the standard rendering pipeline, textures are introduced for each hair card to represent various attributes in the form of RGB images. However, the hair strands are extremely thin, and rasterizing them into RGB textures would introduce severe aliasing error and noisy gradient information. To mitigate this issue, we propose a novel intermediary explicit hair card representation. Specifically, each hair strand $\mathcal{S}_i \in \mathcal{G}_k$ is first projected onto the quad strip C_k. Specifically, for each $\mathbf{s}_{i,j} \in \mathcal{S}_i$, we find the closest point $\mathbf{s}^{\Gamma}_{i,j}$ lying on the associated quad strips C_k. The corresponding 2D uv-coordinate $\mathbf{s}^{\text{uv}}_{i,j} = (u_{i,j}, v_{i,j}) \in [0,1]^2$ on the texture can be computed such that the 3D world position of each sample $\mathbf{s}_{i,j}$ can be reconstructed as:

$$\mathbf{s}_{i,j} = u_{i,j}\mathbf{p}^0_{i,j} + v_{i,j}\mathbf{p}^1_{i,j} + (1 - u_{i,j} - v_{i,j})\mathbf{p}^2_{i,j} + z\mathbf{n}^{\star}_{i,j}, \tag{3}$$

where $\mathbf{p}^{\bullet}_{i,j}$ are the vertices of the corresponding triangle face in the card geometry that contains $\mathbf{s}^{\Gamma}_{i,j}$, $\mathbf{n}^{\star}_{i,j}$ is the normal of that triangle, and $z \in \mathbb{R}$ is the displacement along $\mathbf{n}^{\star}_{i,j}$. Hence, each hair card texture can be explicitly represented as a set of points $\{\mathbf{s}^{\text{uv}}_{i,j}\}$ embedded within the uv-space. Using $\{\mathbf{s}^{\text{uv}}_{i,j}\}$ as our intermediary representation induces two remarkable features. First, this representation is amenable to end-to-end differentiable rendering, while it does not suffer from aliasing error induced by RGB images. Indeed, for differentiable rendering, given the card mesh and explicit hair card texture, 3D hair strands can be first recovered from Eq. 3. The user-defined per-strand hair width is denoted as w_i. Each and every

step of this procedure is differentiable, and a similar technique is proposed in [Sklyarova et al. 2023]. Further, our representation can be converted back to RGB images via rasterization. Finally, to avoid uv-space tangent distortion by the card geometry in world space, we decompose the tangent from strand geometry and store it as a per-vertex 3D attribute denoted as $\{\mathbf{t}_{i,j}^{\mathrm{3D}}\}$, which plays a central role in the hair appearance model. The renderer can also produce depth and coverage maps by utilizing depth and flat color as attributes for each vertex.

4.4 Texture Reduction

To facilitate texture sharing and improve both memory efficiency and runtime performance, users could optionally reduce the number of textures before packing them into a texture atlas. Clusters with similar strand counts and shapes are grouped to share the same texture. Specifically, after the per-cluster optimization, we allow users to provide the target texture number n^t. We then perform another round of k-means clustering to merge cards with similar textures. To this end, we first rasterize the UV-space strands into RGB textures and then compute the Learned Perceptual Image Patch Similarity (LPIPS) metric [Zhang et al. 2018] between all pairs of card textures. The LPIPS metric is used to guide our k-means clustering. For each cluster, only the texture closest to the k-mean center is retained, and it is reused across all hair cards within the same cluster. After the clustering, our joint optimization would then fine-tune the shared textures to collectively match the appearance.

4.5 Joint Optimization

After texture reduction, we perform another optimization with the visual losses and geometric regularization to match the collective visual appearance between the input strands $\mathcal{H} = \{\mathcal{S}_i\}$ and our final cards $\{C_i\}$ with uv-space strands $\{\mathbf{s}_{i,j}^{\mathrm{uv}}\}$ and tangent $\{\mathbf{t}_{i,j}^{\mathrm{3D}}\}$.

Optimization. Given the differentiable renderer, we fine-tune both the card geometry and strand textures for each cluster using gradient-based optimization by solving the following optimization:

$$\underset{\mathbf{p}_{k,j}^{\pm},\, \mathbf{s}_{i,j}^{\mathrm{uv}},\, \mathbf{t}_{i,j}^{\mathrm{3D}},\, w_i}{\operatorname{argmin}} \quad \sum_v \lambda^v L^v + \sum_g \lambda^g L^g, \tag{4}$$

where quad strip position $\mathbf{p}_{k,j}^{\pm}$, strand texture coordinates $\mathbf{s}_{i,j}^{\mathrm{uv}}$, tangent $\mathbf{t}_{i,j}^{\mathrm{3D}}$, and per-strand width w_i are all included as decision variables. To ensure high output quality, we combine visual loss terms $v \in \{\text{tangent},\ \text{depth},\ \text{dice}\}$ and geometry regularization terms $g \in \{\text{match},\ \text{collision}\}$, which are detailed below.

Visual Losses. While our ultimate goal is to produce a hair card model that looks identical to the input strand model, rendering complicated light interactions between a large number of hair segments is computationally intractable. In particular, the interaction between light and a hair strand is described by the Bidirectional Curve Scattering Distribution Function (BCSDF) denoted $f(\theta_i, \phi_i, \theta_o, \phi_o)$, as proposed by Marschner et al. [2003]. Here, $\langle\theta_i, \phi_i\rangle$ and $\langle\theta_o, \phi_o\rangle$ represent the incoming and outgoing light directions, respectively. The inclination angle $\theta_\bullet$ represents the deviation from the strand normal plane, while the azimuth angle $\phi_\bullet$ captures the orientation around the hair axis, both derived from the strand tangent. In order for more efficient optimization, instead of matching the final appearance, our optimization matches the tangent channel for shading and the depth channel for strand position along the view direction. In particular, tangent and depth losses are measured by the Mean-Squared Error (MSE) over all views as:

$$L^v = \int_{\mathbb{S}^2} \mathrm{MSE}\big(\mathcal{I}^v(\mathcal{H}, \omega), \mathcal{I}^v(\langle C_k, \mathbf{s}_{i,j}^{\mathrm{uv}}, \mathbf{t}_{i,j}^{\mathrm{3D}}, w_i\rangle, \omega)\big) d\omega, \tag{5}$$

where $\mathcal{I}^v(\bullet, \omega)$ is the rendering function that rasterizes the channel $v \in \{\text{tangent},\ \text{depth}\}$ of entities $\bullet$ under view direction ω. We further add the following dice loss [Sudre et al. 2017] for matching the silhouette:

$$L^{\mathrm{dice}} = \int_{\mathbb{S}^2} 1 - D\big(\mathcal{I}^{\mathrm{mask}}(\mathcal{H}, \omega), \mathcal{I}^{\mathrm{mask}}(\langle C_k, \mathbf{s}_{i,j}^{\mathrm{uv}}, w_i\rangle, \omega)\big) d\omega, \tag{6}$$

where $D(A, B) = 2|A \cap B|/(|A| + |B|)$ is the Sørensen-Dice coefficient to quantify the similarity between two masks, A and B, where $|A|$ and $|B|$ are the total number of pixels in A and B, and $|A \cap B|$ is the common area between A and B. For rendering the binary mask image, we set all pixels to one in a zero background. Note that we do not need the tangents $\mathbf{t}_{i,j}^{\mathrm{3D}}$ for mask rendering.

Geometric Regularization. For accurate visual appearance, we treat $\mathbf{t}_{i,j}^{\mathrm{3D}}$ as a separate decision variable that is not bound to the strand geometry. However, we still encourage the geometric definition of tangent is consistent with the optimized tangent direction, via the regularization:

$$L^{\mathrm{match}} = \sum_{\mathcal{G}_k} \sum_{\mathcal{S}_i \in \mathcal{G}_k} \sum_{j=1}^{N^s} \left\| \mathbf{t}_{i,j}^{\mathrm{3D}} - \frac{\mathbf{s}_{i,j+1} - \mathbf{s}_{i,j}}{\|\mathbf{s}_{i,j+1} - \mathbf{s}_{i,j}\|} \right\|^2. \tag{7}$$

Finally, we introduce a collision loss to prevent hair cards from penetrating the head mesh, formulated as:

$$L^{\mathrm{collision}} = \sum_{\mathcal{G}_k} \sum_{j=1}^{N^s} \sum_{\bullet \in \{+,-\}} \| \min(0, \mathrm{SDF}(\mathbf{p}_{k,j}^{\bullet}, \mathcal{M}^{\mathrm{head}}))\|^2, \tag{8}$$

where SDF is the signed distance from a point to the head mesh.

Texture Baking. As our final step, after optimization, we rasterize the UV-space strands and tangents into RGB-format textures to generate the final tangent, depth, and alpha maps, as shown in Fig. 4. Additionally, we bake an ambient occlusion (AO) texture to enhance visual detail and store local lighting information on the hair cards. We convert explicit hair card texture into 3D tubes in UV-space and illuminate them with a directional light along the z-direction. Next, we perform standard offline ray-traced AO baking, computing surface color contributions after multiple light bounces. Finally, the resulting shaded surface is projected back onto the card plane to create the AO texture. Note that while it is possible to also optimize the z offset along the normal direction in Eq. 3, we found in practice that keeping it fixed during optimization can provide sufficient results.

4.6 Extensions

To enhance both the visual fidelity of our output and the practical usability of our pipeline, we introduce two additional optional features.

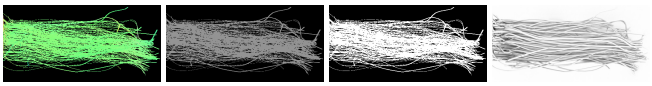

Fig. 4. *Example of our output hair card textures*

Crossed Cards. A common limitation of using billboards/cards is that when the view direction becomes nearly parallel to the card plane, the card appears invisible. To simulate hair volume and improve visual fidelity from multiple viewing angles, we introduce crossed hair cards during generation. Specifically, for each cluster, we generate a pair of hair cards placed at a 90-degree angle to each other, creating the illusion of volumetric hair using flat geometry, as shown in Fig. 5. In practice, during the card generation stage (Sec. 4.2), we simply create an additional card perpendicular to the primary card at each segment by rotating both $\mathbf{n}_{k,j}$ and $\mathbf{t}_{k,j}$ over 90 degrees, giving a crossed configuration.

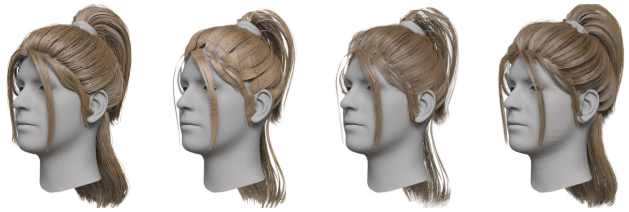

Fig. 5. *Compared with single cards from Unreal Engine auto tool [Epic Games 2025] and ours, our crossed cards can create a volumetric appearance, especially around the ponytail.*

Hair Cap. Since hair cards alone may not fully cover the scalp under certain camera angles or lighting conditions, particularly near the roots, we construct a hair cap, a base layer of geometry and texture applied to the scalp that simulates the appearance of short, dense hair, so it can efficiently represent root fuzz and base hair density with minimal computational cost. To create the hair cap mesh and its associated texture, we begin by projecting all hair roots onto their nearest faces on the head mesh $\mathcal{M}^{\text{head}}$. We then extract all faces containing at least one hair root, along with their one-ring neighboring faces, to form the hair cap mesh. In order to prevent z-fighting during rendering, we slightly extrude the vertices of the cap mesh along their normal directions by a small distance ϵ^{cap}. Finally, for each strand, we extract the first sub-segment with a fixed arc-length of ϵ^{root} and bake the sub-segment onto the scalp texture.

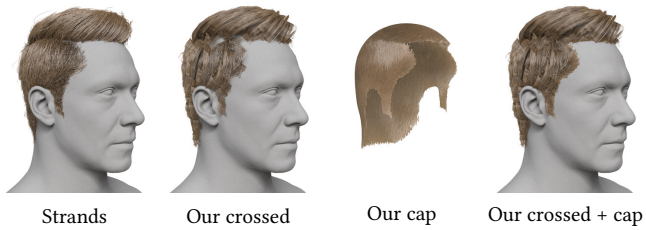

Fig. 6. *Our hair cap can improve the scalp coverage and hairline.*

Specifically, we bake the tangent, alpha, and AO into three texture channels as shown in Fig. 6.

5 IMPLEMENTATION DETAILS

This section provides a detailed description of the implementation of our rendering pipeline employed in both the optimization stage and the final results presentation.

Optimization Stage. After reconstructing all strand control points using Eq. 3, the strands are rasterized into camera-oriented quad strips with a user-defined strand width through Nvdiffrast [Laine et al. 2020]. Since our optimization is guided by loss functions defined over tangent, depth, and coverage, no hair BCSDF is employed for shading. Instead, the renderer produces these G-buffer maps with the corresponding attributes for evaluating the loss function.

Rendering Final Results. To ensure high-quality rendering results, we employ an offline path tracing pipeline for final image generation. Hair appearance is modeled using the classical formulation of hair BCSDF [Marschner et al. 2003], and the local frames are reconstructed from tangent and depth textures during BCSDF sampling and evaluation. Upon ray–card intersection, the alpha texture is used to determine the strand occupancy at the texel level. Hair-hair occlusion is separated into two stages: inter-card occlusion is directly resolved via ray tracing, while intra-card strand occlusion is approximated using our precomputed AO texture.

6 RESULTS

We conduct all experiments on a system equipped with an AMD Ryzen Threadripper 3970X 32-core CPU, 256 GB of memory, and an NVIDIA RTX 3090 GPU. Our framework is implemented in PyTorch and customizes Nvdiffrast [Laine et al. 2020] as our backbone differentiable render.

We evaluate our pipeline on a diverse set of hairstyles, including short, bangs, straight, ponytail, bun, wavy, curly, and coily, from MetaHuman [Epic Games 2021] and a dataset in previous high-fidelity hair reconstruction work [Shen et al. 2023]. All hair models are down-sampled to use 40K hair strands, each with $n^s = 32$ samples. Following the hair card texture settings used in MetaHuman, we set our output texture atlas size to 2048×2048, which accommodates $n^t = 4 \times 8 = 32$ individual hair card textures, each at a resolution of 512×256 by default. The card orientation optimization uses a screen size 128×128, while the joint optimization uses 256×256 to capture strand-level details. For card geometry fitting, we sample 20 candidate normals evenly distributed along a positive half-circle. During optimization, we uniformly sample 64 viewpoints on the unit sphere, oriented toward the origin, to balance between coverage and computational efficiency. We set n^q to 31 for standard cards and 15 for cross cards, except for Afro, we use 48. To assess the visual similarity to the input strand-based model, we use three averaged metrics, as presented in Table 1. PSNR ↑ for the shading evaluation, LPIPS ↓ for perceptual similarity, especially high-frequency hair details, and coverage error ↓ for the hair silhouette. We also use chamfer distance ↓ to evaluate the geometric accuracy. The average values of these metrics are computed over 12 uniformly distributed viewpoints around the hair.

Table 1. Statistics about card # used in both our method and Unreal Engine's auto cards, as well as the corresponding visual metric statistics, PSNR ↑, LPIPS ↓, coverage error ↓, and chamfer distance (CD) ↓. Crossed card is indicated by ×2.

			Unreal Engine [Epic Games 2025]				Ours			
Model	Fig. #	Card #	PSNR ↑	LPIPS ↓	Coverage ↓	CD ↓	PSNR ↑	LPIPS ↓	Coverage ↓	CD ↓
Ponytail	5	100×2	20.91	0.123	0.022	0.003	**22.70**	**0.095**	**0.010**	0.003
Short	6	100×2	22.34	0.112	0.014	0.007	**23.51**	**0.100**	**0.010**	0.004
Straight	7	100	17.39	0.230	0.038	0.005	**18.21**	**0.176**	**0.034**	0.005
Curly	9	400	18.54	0.153	0.038	0.005	**20.89**	**0.138**	**0.016**	0.005
Bangs	9	100	18.09	0.218	0.032	0.006	**20.47**	**0.154**	**0.018**	0.006
Blowout	9	200	17.16	0.200	0.038	0.005	**20.44**	**0.162**	**0.017**	0.005
Wavy	9	200	16.82	0.228	0.041	0.006	**19.89**	**0.166**	**0.016**	0.006
Bun	11	351	17.78	0.209	0.038	0.004	**19.99**	**0.140**	**0.021**	0.004
Braid	13	400	23.25	0.086	0.010	0.003	**24.90**	**0.077**	**0.007**	0.003
Fringe	14	100	18.47	0.166	0.036	0.007	**19.24**	**0.158**	**0.029**	0.008
Coily	17	400×2	17.90	0.174	0.032	0.007	**19.09**	**0.131**	**0.022**	0.007

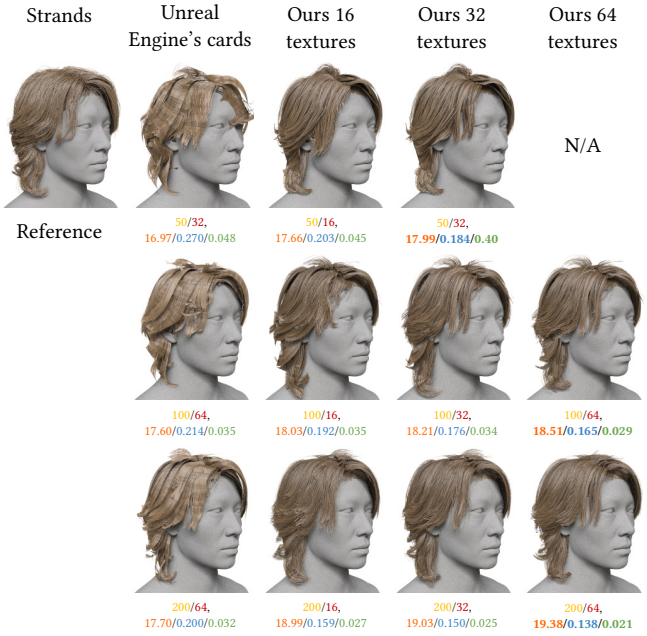

Fig. 7. *Comparison with results from UE automatic hair card generator [Epic Games 2025]. ●/●, ●/●/● indicate the number of cards, the number of textures, averaged PSNR ↑, LPIPS ↓, and coverage error ↓, respectively.*

Ablation Study on Card # and Texture #. Fig. 7 presents a comparison between our method and Unreal Engine's automatic hair card generator [Epic Games 2025] under varying numbers of cards and texture budgets. We evaluate results using 200, 100, and 50 cards, targeting low-resolution LoD hair generation. For textures, we test with 16 and 32 textures, following the industrial best practices for texture budgeting. As expected, visual similarity to the reference strand-based model decreases as the card and texture budgets are reduced. Therefore, the key to creating a good hair model lies in finding a balance between visual quality and geometric efficiency.

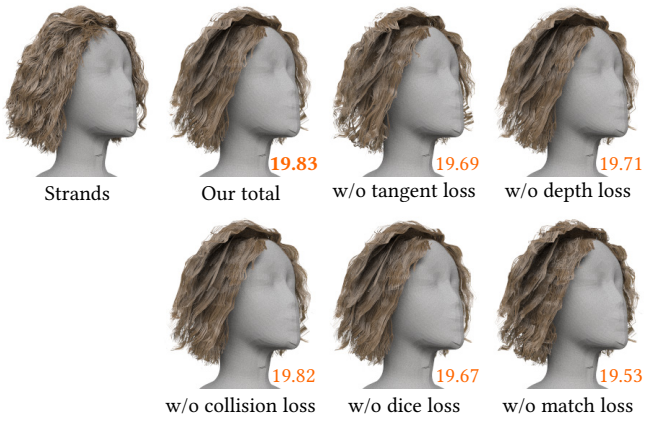

Fig. 8. *Ablation study on individual loss components. Removing any of the proposed loss terms either degrades the visual quality or increases the deviation from the strand-based model. ● indicates PSNR ↑.*

Nevertheless, our method consistently achieves higher visual fidelity in all three metrics, even when using fewer textures and only 50 cards.

Ablation Study on Losses. We further conduct an ablation study on all loss and regularization terms on a straight hairstyle from MetaHuman [Epic Games 2021] with 100 hair cards and 32 textures. As shown in Fig. 8, each component is necessary to achieve consistent and accurate results. Unfortunately, due to the subpixel nature of hair strands and their complicated geometries, traditional image-based metrics such as MSE and PSNR cannot fully capture the perceptual quality of hair. For instance, omitting tangent loss often results in unnatural zigzag strands. Similarly, allowing cards to intersect with the head may seem minor in visual metrics, but it causes noticeable artifacts during simulation. For the rest of the experiments, we adopt two sets of loss weights for different hair types. For straight hair, we use $\lambda^{\text{tangent}} = 10$, $\lambda^{\text{depth}} = 10$, $\lambda^{\text{dice}} = 5$, $\lambda^{\text{match}} = 3$, and $\lambda^{\text{collision}} = 1 \times 10^5$. For curly hair, we use a different configuration as $\lambda^{\text{tangent}} = 5$, $\lambda^{\text{depth}} = 15$, $\lambda^{\text{dice}} = 3$, $\lambda^{\text{match}} = 3$, and

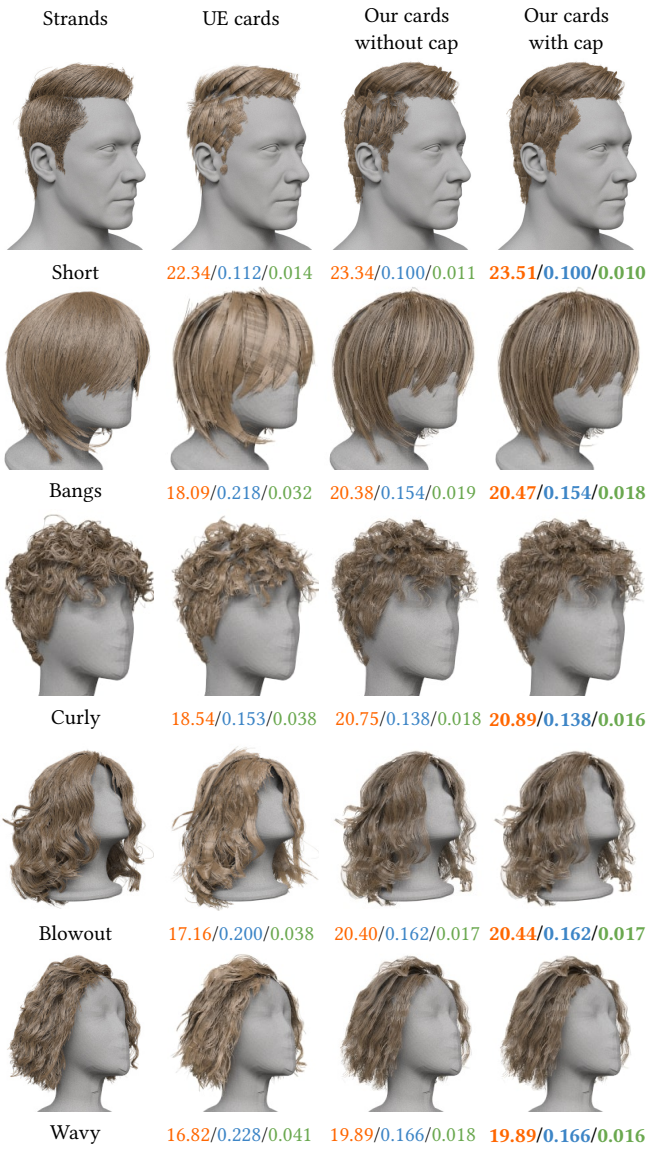

Fig. 9. *Our results outperform UE cards over a variety of hair styles, including short, bangs, wavy, curly, and blowout, regardless of whether with a hair cap or not.*

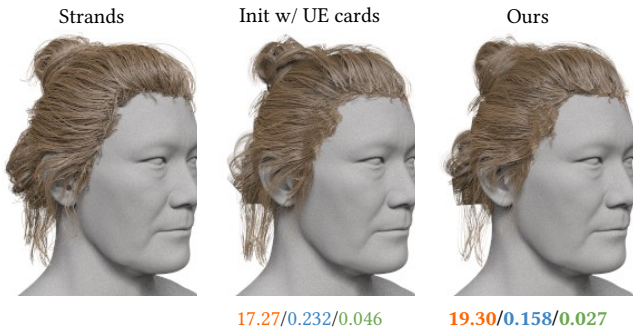

Fig. 10. *Ablation study on card initialization. Using UE cards as initialization degrades the visual quality.*

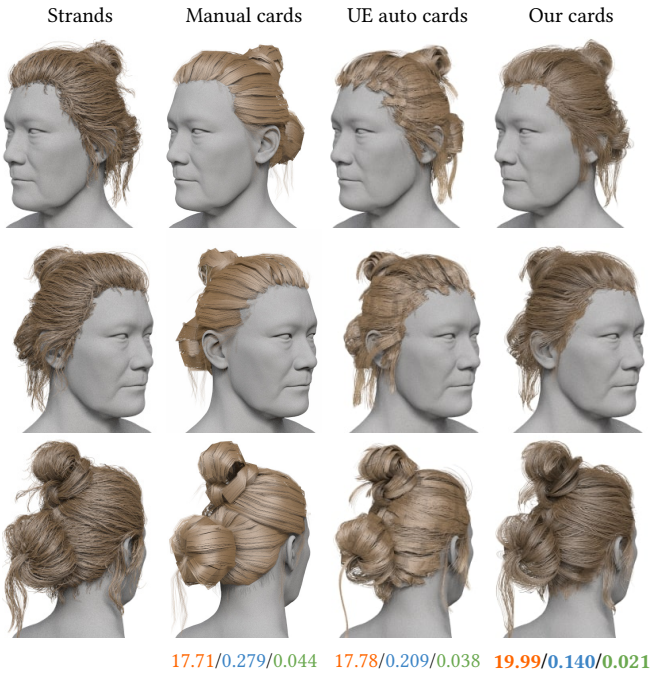

Fig. 11. *Comparison with Unreal Engine's auto-generated and manually crafted cards. Our method outperforms the other two in all metrics with the same 351 cards and 32 textures.*

$\lambda^{\text{collision}} = 1 \times 10^5$, to encourage more curl strands. Note that, due to the inherent geometric differences between straight and curly hair, we adopt two sets of configurations that work well across all examples. We believe this choice does not compromise the generalizability of our method.

Ablation Study on Hair Caps. Our results outperform Unreal Engine's cards over a variety of hair styles, including short, bangs, wavy, curly, and blowout, as shown in Fig. 9. The hair cap primarily improves scalp coverage and hairline, but does not affect overall hair silhouette. As for the geometry, hair cards inherently involve a trade-off between visual fidelity and geometric accuracy. This work primarily aims to reproduce the overall appearance within a constrained geometric budget. Hair geometry that is barely visible contributes little to appearance, so precise geometric accuracy in those regions is less critical. For this reason, our pipeline is appearance-driven, leveraging differentiable rendering to optimize hair cards for visual similarity rather than exact geometry reproduction.

Ablation Study on Card Initialization. We conduct an ablation study on card orientation optimization on Bun hairstyle from MetaHuman [Epic Games 2021] with 200 hair cards and 32 textures. As shown in Fig. 10, using cards generated from the UE automatic hair card generator as the initial shape for our optimization results in a degradation of visual quality, while our card geometry initialization leads to better optimized results.

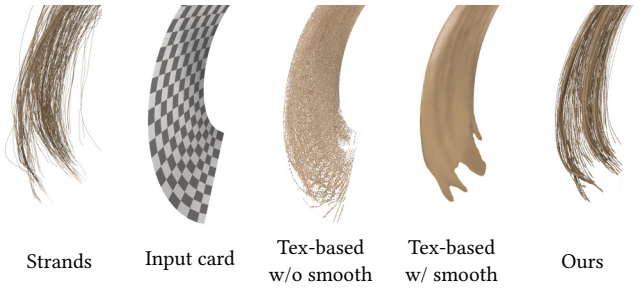

Fig. 12. *Given input card and hair views from eight directions, directly optimizing texture leads to noisy results, while adding smoothness loss eliminates high-frequency hair structure. Our strand-based optimization avoids the above issues effectively.*

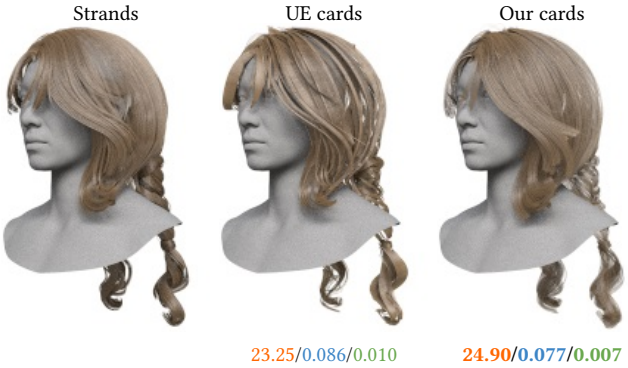

Fig. 13. *Our method applies equally well to complex structures such as knots and braids.*

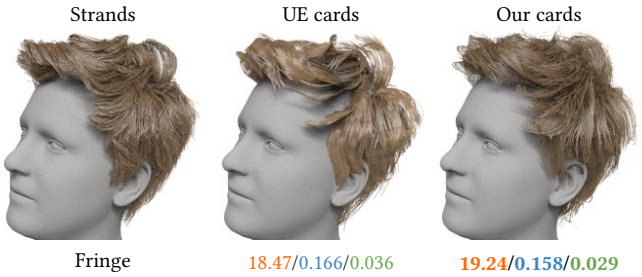

Fig. 14. *Hair card extraction for strand-based hairs from multi-view reconstruction [Zakharov et al. 2024].*

Comparison with UE Auto-generated and Manual-crafted Cards. We compare our results with the output of UE built-in automatic hair card generation tool [Epic Games 2025] and artist-crafted hair cards from MetaHuman [Epic Games 2021]. We compare with the lowest LoD hair card model from MetaHuman, which only contains 351 cards and 20 textures. As shown in Fig. 11, using the same set of 351 hair cards and 20 textures, our method achieves better visual fidelity to the input strand-based model than both Unreal Engine auto-generated and manual-crafted cards. While our higher PSNR partly benefits from more accurate AO textures, as we preserve the displacement during optimization, the lower LPIPS and coverage errors clearly demonstrate that our results more faithfully preserve hair occupancy and shape.

Fig. 15. *Comparison between simulating and rendering the input strand-based model and our output hair card model in Unreal Engine [Epic Games 2021].*

Comparison with Optimizing Texture Directly. We validate the necessity of our explicit strand representation by comparing it with a conventional differentiable rendering pipeline that directly optimizes hair card textures using Nvdiffrast [Laine et al. 2020]. In this experiment, we provide a fixed hair card geometry along with tangent images from eight view directions. We then use Nvdiffrast to optimize the texture mapped to the card so that its rendered appearance matches the given tangent images as closely as possible from those views. Although Nvdiffrast can produce hair textures with distinct shape features when optimizing for a single view direction, extending the optimization to multiple views leads to noisy and inconsistent results. Introducing a smoothness loss can eliminate noise at the cost of losing high-frequency details, as shown in Fig. 12. More importantly, this texture-based approach lacks the spatial structure for computing ambient occlusion accurately, as hair-to-hair spatial relationships are not preserved during texture optimization.

Braid Styles. Fig. 2 demonstrates that buns can be grouped based on their geometric and spatial similarity, a strategy that applies equally well to complex structures such as knots and braids. Fig. 13

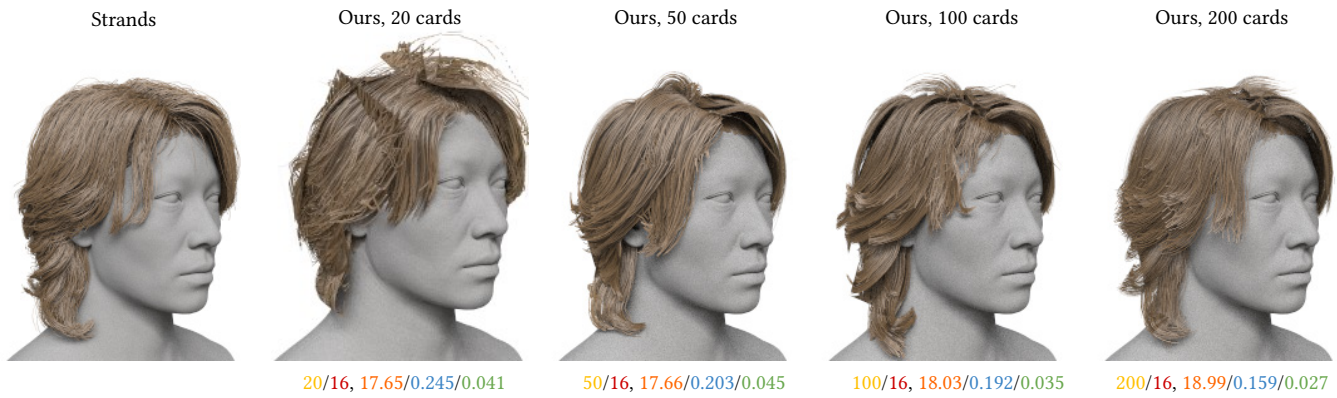

Fig. 16. *Using too few hair cards, for example, only 20, fails to adequately represent the hair structure, even for relatively simple straight styles. As the number of cards increases, the overall fidelity of the hair representation improves.* ●/●, ●/●/● *indicate the number of cards, the number of textures, averaged PSNR ↑, LPIPS ↓, and coverage error ↓, respectively.*

shows another example of braided hairstyles, where our method can produce hair cards that reproduce the braided style with a small number of hair cards.

Strand-based Hairs from Multi-view Reconstruction. To further validate the scope of application of our method, we test our method on the strand model generated using the multi-view reconstruction [Zakharov et al. 2024]. As shown in Fig. 14, our method can automatically extract hair cards from a reconstructed strand-based fringe hairstyle.

Comparison with Strand Model in Game Engine. Fig. 15 demonstrates that our hair cards can be used directly in Unreal Engine with real-time rendering and simulation. Specifically, we generate one guide hair per card for simulation and drive the deformation of the hair cards using linear blend skinning (LBS) as mentioned in [Hsu et al. 2024]. The strand-based model with approximately 50,000 strands requires 2.4 ms per frame for rendering, whereas our card model only takes 0.8 ms for 512 hair cards. Please refer to the supplemental video.

Cards Extraction Performance. Given input hair with 40K strands with $n^s = 32$ samples, the average process time of our card generation pipeline for cards with 100 cards and 32 textures is about 46 minutes, where joint optimization takes about 42% of the computation time. For a larger configuration of 400 cards with the same number of textures, the total runtime increases to approximately 1 hour, with joint optimization comprising roughly one-third of the computation time. In both cases, the optimization process converges within roughly 200 epochs. Memory cost is about 7 GB for 100 cards and 10 GB for 400 cards, respectively.

7 CONCLUSION

We have presented a fully automated pipeline for converting strand-based hair models into efficient and visually compelling hair card representations. By leveraging a differentiable rendering framework, our method first clusters the hair strands to initialize card geometry and then clusters the hair textures to share, thereby reducing memory cost. Finally, we conduct joint optimization over both card geometry and textures. A key contribution of our approach is the introduction of 2D curve-based texture encoding, which offers a resolution-independent representation that effectively captures fine strand details while remaining compatible with differentiable rendering. Our method supports a wide range of hairstyles and lengths, introducing mechanisms such as hair caps and crossed card generation to handle visually complex hair types, including short and coily styles. Additionally, the ability to share textures across LoDs makes our approach well-suited for real-time applications where memory efficiency and performance are critical.

Limitations. Although our pipeline demonstrates strong performance and visual fidelity across a range of hairstyles, several limitations remain. First, optimization is conducted in multiple substeps rather than as a fully end-to-end differentiable pipeline, which may limit global consistency. Given the already large number of variables to optimize (approximately 500K variables for the afro example), increasing the number of strand control points would indeed improve the representation capability of strands. However, this also enlarges the solution space, making optimization more prone to getting trapped in local minima. Second, the loss functions primarily rely on visible attributes such as tangent alignment, depth, and mask coverage. However, for complex hairstyles, especially those with dense or layered structures, interior strand arrangements play an important visual role and are not explicitly enforced. Besides, hair cards, as a low-cost representation, have an inherent limitation in that it is mostly for static and moderate animation in games. Under large deformation, it can cause occlusion artifacts. Third, our method assumes a uniform hair color throughout the entire hairstyle, which may not generalize well to stylized or multicolored hair. Finally, hair type with highly curved geometry, such as coily hair, remains a challenge. Theoretically, increasing the number of hair clusters and card geometries makes it easier to represent complex hairstyles. Ultimately, if we assign one card per strand, our pipeline would always succeed. However, our goal is to strike a balance between geometry and appearance. Fig. 16 shows that using too few hair cards, for example, only 20, fails to adequately represent the hair structure, even for relatively simple straight styles. As the number

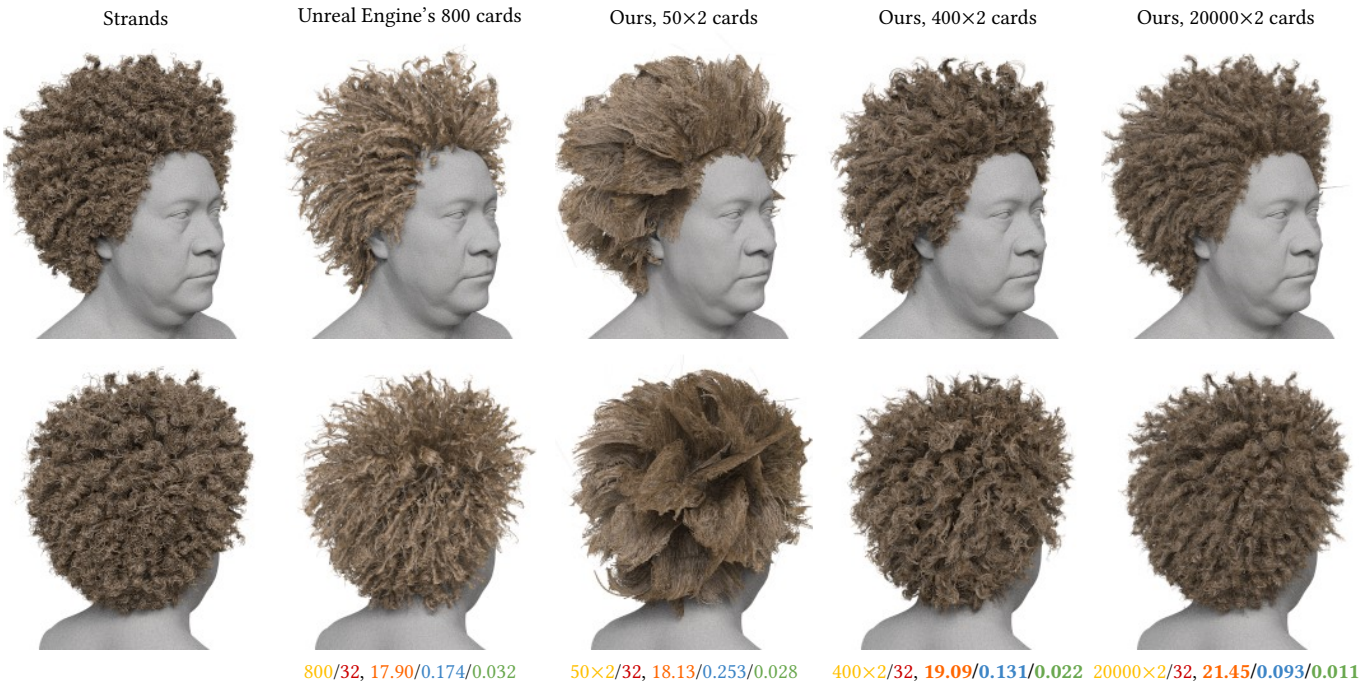

Fig. 17. *While our method can reproduce the appearance of coily hair with a relatively high number of cards (e.g., 20,000×2), it becomes challenging to preserve fidelity when the number of cards is limited.* ●/●, ●/●/● *indicate the number of cards, the number of textures, averaged PSNR ↑, LPIPS ↓, and coverage error ↓, respectively.*

of cards increases, the overall fidelity of the hair representation improves. Similarly, Fig. 17 demonstrates that our method struggles to capture complex coily structures when both the number of hair cards and control points are limited. However, with a sufficiently large number of cards (e.g., 20,000×2), even intricate coily patterns can be represented with fine detail.

REFERENCES

C. Andujar, P. Brunet, A. Chica, I. Navazo, J. Rossignac, and A. Vinacua. 2004. Computing Maximal Tiles and Application to Impostor-Based Simplification. *Computer Graphics Forum* 23, 3 (2004), 401–410.

S. Behrendt, C. Colditz, O. Franzke, J. Kopf, and O. Deussen. 2005. Realistic real-time rendering of landscapes using billboard clouds. *Computer Graphics Forum* 24, 3 (2005), 507–516.

Miklós Bergou, Max Wardetzky, Stephen Robinson, Basile Audoly, and Eitan Grinspun. 2008. Discrete Elastic Rods. In *ACM SIGGRAPH 2008 Papers* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 63, 12 pages.

Florence Bertails, Sunil Hadap, Marie-Paule Cani, Ming Lin, Tae-Yong Kim, Steve Marschner, Kelly Ward, and Zoran Kačić-Alesić. 2008. Realistic hair simulation: animation and rendering. In *ACM SIGGRAPH 2008 Classes* (Los Angeles, California) *(SIGGRAPH '08)*. Association for Computing Machinery, New York, NY, USA, Article 89, 154 pages.

Menglei Chai, Tianjia Shao, Hongzhi Wu, Yanlin Weng, and Kun Zhou. 2016. AutoHair: fully automatic hair modeling from a single image. *ACM Trans. Graph.* 35, 4, Article 116 (July 2016), 12 pages.

Yunlu Chen, Francisco Vicente Carrasco, Christian Häne, Giljoo Nam, Jean-Charles Bazin, and Fernando D De la Torre. 2024. Doubly Hierarchical Geometric Representations for Strand-based Human Hairstyle Generation. *Advances in Neural Information Processing Systems* 37 (2024), 89728–89751.

Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey. 2003. Billboard clouds for extreme model simplification. *ACM Trans. Graph.* 22, 3 (jul 2003), 689–696.

Epic Games. 2021. Unreal Engine. https://www.unrealengine.com

Epic Games. 2025. Hair Card Generator. [Online]. Available from: https://dev.epicgames.com/documentation/en-us/unreal-engine/hair-card-generator-for-grooms-in-unreal-engine.

Hongbo Fu, Yichen Wei, Chiew-Lan Tai, and Long Quan. 2007. Sketching hairstyles. In *Proceedings of the 4th Eurographics Workshop on Sketch-Based Interfaces and Modeling* (Riverside, California) *(SBIM '07)*. Association for Computing Machinery, New York, NY, USA, 31–36.

Yang Guang and Huang Zhiyong. 2002. A method of human short hair modeling and real time animation. In *10th Pacific Conference on Computer Graphics and Applications, 2002. Proceedings.* IEEE, New York, NY, USA, 435–438.

Chengan He, Xin Sun, Zhixin Shu, Fujun Luan, Soren Pirk, Jorge Alejandro Amador Herrera, Dominik Michels, Tuanfeng Yang Wang, Meng Zhang, Holly Rushmeier, and Yi Zhou. 2025. Perm: A Parametric Representation for Multi-Style 3D Hair Modeling. In *The Thirteenth International Conference on Learning Representations.* IEEE, New York City, NY, USA, 27 pages.

Jozef Hladky, Michael Stengel, Nicholas Vining, Bernhard Kerbl, Hans-Peter Seidel, and Markus Steinberger. 2022. QuadStream: A Quad-Based Scene Streaming Architecture for Novel Viewpoint Reconstruction. *ACM Trans. Graph.* 41, 6, Article 233 (Nov. 2022), 13 pages.

Jerry Hsu, Tongtong Wang, Zherong Pan, Xifeng Gao, Cem Yuksel, and Kui Wu. 2023. Sag-Free Initialization for Strand-Based Hybrid Hair Simulation. *ACM Trans. Graph.* 42, 4, Article 74 (jul 2023), 14 pages.

Jerry Hsu, Tongtong Wang, Zherong Pan, Xifeng Gao, Cem Yuksel, and Kui Wu. 2024. Real-Time Physically Guided Hair Interpolation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2024)* 43, 4, Article 95 (07 2024), 11 pages.

Jerry Hsu, Tongtong Wang, Kui Wu, and Cem Yuksel. 2025. Stable Cosserat Rods. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25)*. Association for Computing Machinery, New York, NY, USA, Article 75, 10 pages.

Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. 2015. Single-view hair modeling using a hairstyle database. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 1–9.

Liwen Hu, Shunsuke Saito, Lingyu Wei, Koki Nagano, Jaewoo Seo, Jens Fursund, Iman Sadeghi, Carrie Sun, Yen-Chun Chen, and Hao Li. 2017. Avatar digitization from a single image for real-time rendering. *ACM Transactions on Graphics (ToG)* 36, 6 (2017), 1–14.

Li Huang, Fan Yang, Chendi Wei, Yu Ju (Edwin) Chen, Chun Yuan, and Ming Gao. 2023. Towards Realtime: A Hybrid Physics-Based Method for Hair Animation on GPU. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 3, Article 43 (aug 2023), 18 pages.

Tao Huang, Yang Zhou, Daqi Lin, Junqiu Zhu, Ling-Qi Yan, and Kui Wu. 2025. Real-time Level-of-detail Strand-based Rendering. *Computer Graphics Forum* 44 (2025),

13 pages.
Wenzel Jakob, Jonathan T. Moon, and Steve Marschner. 2009. Capturing hair assemblies fiber by fiber. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 1–9.
Yibing Jiang. 2016. The process of creating volumetric-based materials in uncharted 4. *Siggraph Courses: Advances in Real-Time Rendering in Games, Anaheim, CA* 0, 0 (2016), 0.
Chuan Koon Koh and Zhiyong Huang. 2001. A Simple Physics Model to Animate Human Hair Modeled in 2D Strips in Real Time. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation* (Manchester, UK). Springer-Verlag, Berlin, Heidelberg, 127–138.
Waiming Kong and Masayuki Nakajima. 1998. Generation of 3D Hair Model from Multiple Pictures. *The Journal of the Institute of Image Information and Television Engineers* 52, 9 (1998), 1351–1356.
Zhiyi Kuang, Yiyang Chen, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2022. DeepMVSHair: Deep Hair Modeling from Sparse Views. In *SIGGRAPH Asia 2022 Conference Papers* (Daegu, Republic of Korea) *(SA '22)*. Association for Computing Machinery, New York, NY, USA, Article 10, 8 pages.
Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph.* 39, 6, Article 194 (Nov. 2020), 14 pages.
Puneet Lall, Silviu Borac, Dave Richardson, Matt Pharr, and Manfred Ernst. 2018. View-Region Optimized Image-Based Scene Simplification. *Proc. ACM Comput. Graph. Interact. Tech.* 1, 2, Article 26 (Aug. 2018), 22 pages.
Minjae Lee, David Hyde, Michael Bao, and Ronald Fedkiw. 2019. A Skinned Tetrahedral Mesh for Hair Animation and Hair-Water Interaction. *IEEE Transactions on Visualization and Computer Graphics* 25, 3 (2019), 1449–1459.
Shu Liang, Xiufeng Huang, Xianyu Meng, Kunyao Chen, Linda G Shapiro, and Ira Kemelmacher-Shlizerman. 2018. Video to fully automatic 3d hair model. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–14.
Haimin Luo, Min Ouyang, Zijun Zhao, Suyi Jiang, Longwen Zhang, Qixuan Zhang, Wei Yang, Lan Xu, and Jingyi Yu. 2024. Gaussianhair: Hair modeling and rendering with light-aware gaussians. *arXiv preprint arXiv:2402.10483* 0 (2024), 19 pages.
Stephen R Marschner, Henrik Wann Jensen, Mike Cammarano, Steve Worley, and Pat Hanrahan. 2003. Light scattering from human hair fibers. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 780–791.
Sylvain Paris, Will Chang, Oleg I Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. 2008. Hair photobooth: geometric and photometric acquisition of real hairstyles. *ACM Trans. Graph.* 27, 3 (2008), 30.
Radu Alexandru Rosu, Shunsuke Saito, Ziyan Wang, Chenglei Wu, Sven Behnke, and Giljoo Nam. 2022. Neural Strands: Learning Hair Geometry and Appearance from Multi-view Images. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII* (Tel Aviv, Israel). Springer-Verlag, Berlin, Heidelberg, 73–89.
Yuefan Shen, Shunsuke Saito, Ziyan Wang, Olivier Maury, Chenglei Wu, Jessica Hodgins, Youyi Zheng, and Giljoo Nam. 2023. CT2Hair: High-Fidelity 3D Hair Modeling using Computed Tomography. *ACM Trans. Graph.* 42, 4, Article 75 (July 2023), 13 pages.
Yuefan Shen, Changgeng Zhang, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2020. Deepsketchhair: Deep sketch-based 3d hair modeling. *IEEE transactions on visualization and computer graphics* 27, 7 (2020), 3250–3263.
Ari Silvennoinen, Hannu Saransaari, Samuli Laine, and Jaakko Lehtinen. 2014. Occluder Simplification Using Planar Sections. *Comput. Graph. Forum* 33, 1 (Feb. 2014), 235–245.
Vanessa Sklyarova, Jenya Chelishev, Andreea Dogaru, Igor Medvedev, Victor Lempitsky, and Egor Zakharov. 2023. Neural Haircut: Prior-Guided Strand-Based Hair Reconstruction . In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE Computer Society, Los Alamitos, CA, USA, 19705–19716.
Vanessa Sklyarova, Egor Zakharov, Otmar Hilliges, Michael J. Black, and Justus Thies. 2024. Text-Conditioned Generative Model of 3D Strand-Based Human Hairstyles. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, New York City, NY, USA, 4703–4712.
Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and M. Jorge Cardoso. 2017. Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, M. Jorge Cardoso, Tal Arbel, Gustavo Carneiro, Tanveer Syeda-Mahmood, João Manuel R.S. Tavares, Mehdi Moradi, Andrew Bradley, Hayit Greenspan, João Paulo Papa, Anant Madabhushi, Jacinto C. Nascimento, Jaime S. Cardoso, Vasileios Belagiannis, and Zhi Lu (Eds.). Springer International Publishing, Cham, 240–248.
Tiancheng Sun, Giljoo Nam, Carlos Aliaga, Christophe Hery, and Ravi Ramamoorthi. 2021. Human Hair Inverse Rendering using Multi-View Photometric data. In *Eurographics Symposium on Rendering - DL-only Track*, Adrien Bousseau and Morgan McGuire (Eds.). The Eurographics Association, Groene Loper 3, 5612AE Eindhoven, The Netherlands, 179–190.
Sebastian Tafuri. 2019. Strand-based Hair Rendering in Frostbite. *ACM SIGGRAPH Courses: Advances in Real-Time Rendering in Games Course* 0, 0 (2019), 0.
Yusuke Takimoto, Hikari Takehara, Hiroyuki Sato, Zihao Zhu, and Bo Zheng. 2024. Dr. Hair: Reconstructing Scalp-Connected Hair Strands without Pre-Training via Differentiable Rendering of Line Segments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New York City, NY, USA, 20601–20611.
P. Volino and N. Magnenat-Thalmann. 2006. Real-time animation of complex hairstyles. *IEEE Transactions on Visualization and Computer Graphics* 12, 2 (2006), 131–142.
Lvdi Wang, Yizhou Yu, Kun Zhou, and Baining Guo. 2009. Example-Based Hair Geometry Synthesis. In *ACM SIGGRAPH 2009 Papers* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 56, 9 pages.
Ziyan Wang, Giljoo Nam, Tuur Stuyck, Stephen Lombardi, Chen Cao, Jason Saragih, Michael Zollhöfer, Jessica Hodgins, and Christoph Lassner. 2023. Neuwigs: A neural dynamic model for volumetric hair capture and animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, Los Alamitos, CA, USA, 8641–8651.
Kelly Ward, Ming C. Lin, Joohi Lee, Susan Fisher, and Dean Macri. 2003. Modeling Hair Using Level-of-Detail Representations. In *Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003) (CASA '03)*. IEEE Computer Society, USA, 41.
Haomiao Wu, Alvin Shi, A.M. Darke, and Theodore Kim. 2024a. Curly-Cue: Geometric Methods for Highly Coiled Hair. In *SIGGRAPH Asia 2024 Conference Papers* (Tokyo, Japan) *(SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 112, 11 pages.
Keyu Wu, Lingchen Yang, Zhiyi Kuang, Yao Feng, Xutao Han, Yuefan Shen, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2024b. MonoHair: High-Fidelity Hair Modeling from a Monocular Video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New York, NY, USA, 24164–24173.
Keyu Wu, Yifan Ye, Lingchen Yang, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2022. NeuralHDHair: Automatic High-fidelity Hair Modeling from a Single Image Using Implicit Neural Representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, New York, NY, USA, 1526–1535.
Kui Wu and Cem Yuksel. 2016. Real-Time Hair Mesh Simulation. In *Proceedings of the 20th ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Redmond, Washington) *(I3D '16)*. Association for Computing Machinery, New York, NY, USA, 59–64.
Cem Yuksel, Scott Schaefer, and John Keyser. 2009. Hair Meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2009)* 28, 5, Article 166 (2009), 7 pages.
Egor Zakharov, Vanessa Sklyarova, Michael Black, Giljoo Nam, Justus Thies, and Otmar Hilliges. 2024. Human Hair Reconstruction with Strand-Aligned 3D Gaussians. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part XVI* (Milan, Italy). Springer-Verlag, Berlin, Heidelberg, 409–425.
Meng Zhang and Youyi Zheng. 2019. Hair-GAN: Recovering 3D hair structure from a single image using generative adversarial networks. *Visual Informatics* 3, 2 (2019), 102–112.
Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric . In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 586–595.
Yang Zheng, Menglei Chai, Delio Vicini, Yuxiao Zhou, Yinghao Xu, Leonidas Guibas, Gordon Wetzstein, and Thabo Beeler. 2025. GroomLight: Hybrid Inverse Rendering for Relightable Human Hair Appearance Modeling. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. IEEE Computer Society, Los Alamitos, CA, USA, 16040–16050.
Yujian Zheng, Zirong Jin, Moran Li, Haibin Huang, Chongyang Ma, Shuguang Cui, and Xiaoguang Han. 2023. HairStep: Transfer Synthetic to Real Using Strand and Depth Maps for Single-View 3D Hair Modeling. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, New York, NY, USA, 12726–12735.
Yuxiao Zhou, Menglei Chai, Alessandro Pepe, Markus Gross, and Thabo Beeler. 2023. GroomGen: A High-Quality Generative Hair Model Using Hierarchical Latent Representations. *ACM Trans. Graph.* 42, 6, Article 270 (Dec. 2023), 16 pages.
Yuxiao Zhou, Menglei Chai, Daoye Wang, Sebastian Winberg, Erroll Wood, Kripasindhu Sarkar, Markus Gross, and Thabo Beeler. 2024. Groomcap: High-fidelity prior-free hair capture. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–15.
Yi Zhou, Liwen Hu, Jun Xing, Weikai Chen, Han-Wei Kung, Xin Tong, and Hao Li. 2018. HairNet: Single-View Hair Reconstruction Using Convolutional Neural Networks. In *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI* (Munich, Germany). Springer-Verlag, Berlin, Heidelberg, 249–265.