Transforming Unstructured Hair Strands into Procedural Hair Grooms

WESLEY CHANG, University of California San Diego, USA
ANDREW L. RUSSELL, University of California San Diego, USA
STEPHANE GRABLI, Meta Reality Labs, USA
MATT JEN-YUAN CHIANG, Meta Reality Labs, USA
CHRISTOPHE HERY, Meta Reality Labs, USA
DOUG ROBLE, Meta Reality Labs, USA
RAVI RAMAMOORTHI, University of California San Diego, USA
TZU-MAO LI, University of California San Diego, USA
OLIVIER MAURY, Meta Reality Labs, USA

(a) Target Strands (b) Our Groom (c) Guide Editing (d) Operator Editing

Fig. 1. We design a procedural pipeline that generates hair grooms using a small set of guide strands and artist-friendly hair grooming operators. (a) Given unstructured hair strand geometry, we propose optimization strategies to (b) fit the guide strands and grooming operator parameters of the procedural groom. We can then edit the hair shape and style by (c) modifying the guide strands and (d) changing operator parameters. Head model courtesy of ©Daniel Bystedt.

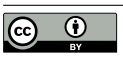
In recent years, reconstruction methods have been developed that can recover strand-level hair geometry from images. However, these methods recover a vast number of individual hair strands that are difficult to edit and simulate. Many methods also rely on neural priors to infer non-visible inner hair, which can result in poor inner hair structure for complex hairstyles, such as curly hair. We propose an inverse hair grooming pipeline that transforms the imperfect 3D strands from these reconstruction methods into procedural hair grooms that consist of a small set of guide strands and hair grooming operators, inspired by pipelines used by artists in popular 3D modeling tools such as Blender and Houdini. We take a probabilistic view of these hair grooms and design various optimization strategies and loss functions to optimize for the guide strands and operator parameters. Due to the proceduralism, our resulting grooms can naturally represent challenging hairstyles, have structurally sound inner hair, and are easily editable.

CCS Concepts: • **Computing methodologies → Parametric curve and surface models**.

Additional Key Words and Phrases: hair modeling, procedural modeling, grooming operators, optimization

ACM Reference Format:
Wesley Chang, Andrew L. Russell, Stephane Grabli, Matt Jen-Yuan Chiang, Christophe Hery, Doug Roble, Ravi Ramamoorthi, Tzu-Mao Li, and Olivier Maury. 2025. Transforming Unstructured Hair Strands into Procedural Hair Grooms. *ACM Trans. Graph.* 44, 4 (August 2025), 20 pages. https://doi.org/10.1145/3731168

Authors' Contact Information: Wesley Chang, wec022@ucsd.edu, University of California San Diego, USA; Andrew L. Russell, alrussell@ucsd.edu, University of California San Diego, USA; Stephane Grabli, sgrabli@meta.com, Meta Reality Labs, USA; Matt Jen-Yuan Chiang, mattchiang@meta.com, Meta Reality Labs, USA; Christophe Hery, chery@meta.com, Meta Reality Labs, USA; Doug Roble, droble@meta.com, Meta Reality Labs, USA; Ravi Ramamoorthi, ravir@ucsd.edu, University of California San Diego, USA; Tzu-Mao Li, tzli@ucsd.edu, University of California San Diego, USA; Olivier Maury, omaury@meta.com, Meta Reality Labs, USA.



ACM 1557-7368/2025/8-ART
https://doi.org/10.1145/3731168

1 Introduction

Recent advances in hair reconstruction have allowed us to accurately recover hair strand-by-strand from a video sequence [Sklyarova et al. 2023; Wu et al. 2024; Zakharov et al. 2024], enabling various applications in fields such as visual effects, gaming, and virtual reality. However, while these methods can reconstruct accurate hair strands, the resulting tens to hundreds of thousands of individual

strands are not intuitive to edit, limiting its usage in these downstream applications. Furthermore, hair grooms created by artists are rarely designed strand-by-strand. Instead, it is common to construct a small set of *guide strands*, which are instanced and interpolated into dense strands, and apply procedural *grooming operators* to modify the hair to achieve the desired hairstyle (e.g., a "curl" operator to make the hair curly). Guide hair and grooming operators are also crucial for artistic control of hair simulation to achieve artists' desired effects.

In this work, we present the first work on transforming unstructured 3D hair strand geometry obtained from hair reconstruction methods into procedural hair grooms. Given a set of hair curves (Figure 1 (a)) and a set of procedural grooming operators, we propose a pipeline to retrieve both the guide hair control vertices and the procedural grooming operator parameters (Figure 1 (b)). Our procedural hair is easy for an artist to edit to achieve different hairstyles (Figure 1 (c) and (d)). Since the number of guide strands and grooming operator parameters is small relative to the number of strands in the target geometry, we focus on reconstructing accurate overall hairstyles, rather than individual strand locations.

Designing optimization strategies to recover these guide strands and operator parameters is non-trivial due to a few major challenges: 1) Both the guide strands and operator parameters need to be optimized without assuming the other is known, even though there is ambiguity between the two. For example, a curly hair groom can be represented using curly guide strands, or a curl operator. 2) Hair grooming pipelines add random variations to individual strands to ensure that not all instanced strands follow the same shape. This, however, means that we cannot expect the strands in the procedural groom to match the strands in the target exactly, and so simple losses like a vertex-wise L^2 loss are not suitable. Rather, a loss that matches the overall hairstyle or distribution is needed. 3) Hair strands that are reconstructed from images often require data-driven priors to infer inner hair that is not visible from any view, and there are no guarantees these inner hairs are accurate. As such, the method must be robust to the imperfect target geometry.

We introduce several contributions to address these challenges:

- We design a procedural hair grooming pipeline (Section 4) that is both intuitive to edit and flexible enough to capture a wide variety of hairstyles.
- We develop a probabilistic view on procedural hair grooms (Section 5.1) to design loss functions that account for the random per-strand variations.
- We design strategies and loss functions to initialize and optimize for guide strands without assuming known grooming operator parameters (Section 5.2, Section 5.3, Section 5.4), by assuming guides are typically smooth and variations are added through operators.
- We design loss functions to optimize for grooming operator parameters that change overall hairstyles (Section 5.5). We also optimize for the random per-strand variations to better match complex real hair grooms (Section 5.6).
- We show that the combination of these items allows our method to handle imperfect target geometry, and ensure that our grooms are structurally sound, even in nonvisible hair (Section 5.7).

We validate our optimization pipeline on both synthetic procedural grooms (Section 7.2, Figure 7) and real hair reconstructions (Section 7.3, Figure 8), on a wide variety of hairstyles, including curly hair (Section 7.1, Table 1). Our method recovers procedural grooms that are both easy to edit and have reasonable inner hair strands, as shown by edits and simulations (Section 7.5, Figure 10).

2 Related Work

2.1 Procedural Modeling

Computer graphics has a long tradition of modeling objects using procedural programs and coarse structures, from textures and shaders [Cook 1984; Perlin 1985], grammars for shapes [Prusinkiewicz 1986], to character rigs [Baran and Popović 2007]. Our task is related to the task of inverse procedural modeling (e.g., [Cascaval et al. 2022; Du et al. 2018; Hu et al. 2019; Talton et al. 2011; Vanegas et al. 2012]) and automatic rigging [Baran and Popović 2007; Xu et al. 2020], where we share the goal of inferring the skeletal structure and retrieving the procedural program parameters from unstructured input.

To our knowledge, our work is the first to reconstruct procedural hair grooms from hair stands, although procedural grooming operators are widely used in hair modeling and rendering [Bhokare et al. 2024; Yuksel et al. 2009], and even in movie production [Hasenbring and Karlsson 2021; Ogunseitan 2022]. Unlike some works in inverse procedural modeling, we assume the types of our grooming operators are known, and we aim to infer the skeletal guide strands and the groom operators parameters from hair strands. The unique challenge in our setting is to design loss functions that can match the distributions of our procedural hair grooms and target hair strands. Due to the scarcity of training data for procedural hair grooms, we do not consider data-driven methods [Xu et al. 2020], but the combination with them could be exciting future work.

2.2 Hair reconstruction and generation

Our method is driven by the recent advances in hair reconstruction and generation. Given a video walkthrough of a person's head or even just a single image, we now have a plethora of methods capable of reconstructing 3D hair strands. Furthermore, there are data-driven generative models that can synthesize hair from random seeds, optionally guided by input text.

A large class of methods takes multiple views of a static subject, with uncontrolled lighting, and directly fits hair strands to the observation without collecting a large set of hair examples [Hu et al. 2014; Jakob et al. 2009; Kong and Nakajima 1998; Luo et al. 2012, 2013; Maeda et al. 2023; Nam et al. 2019; Takimoto et al. 2024; Wei et al. 2005; Zhou et al. 2024]. Some other methods collect a hair database to act as a prior for multi-view reconstruction [Liang et al. 2018; Rosu et al. 2022; Sklyarova et al. 2023; Wu et al. 2024; Zhang et al. 2017, 2018]. Some of these data-driven methods can even be used for generating hair from a single image input [Chai et al. 2016; Hu et al. 2015; Saito et al. 2018; Wu et al. 2022; Zhang and Zheng 2019; Zheng et al. 2023; Zhou et al. 2018], sometimes along with user input strokes for specifying the desired hairstyle [Chai et al. 2015, 2013]. Another class of methods rely on controlled lighting to get very high-quality output [Grabli et al. 2002; Paris et al. 2004,

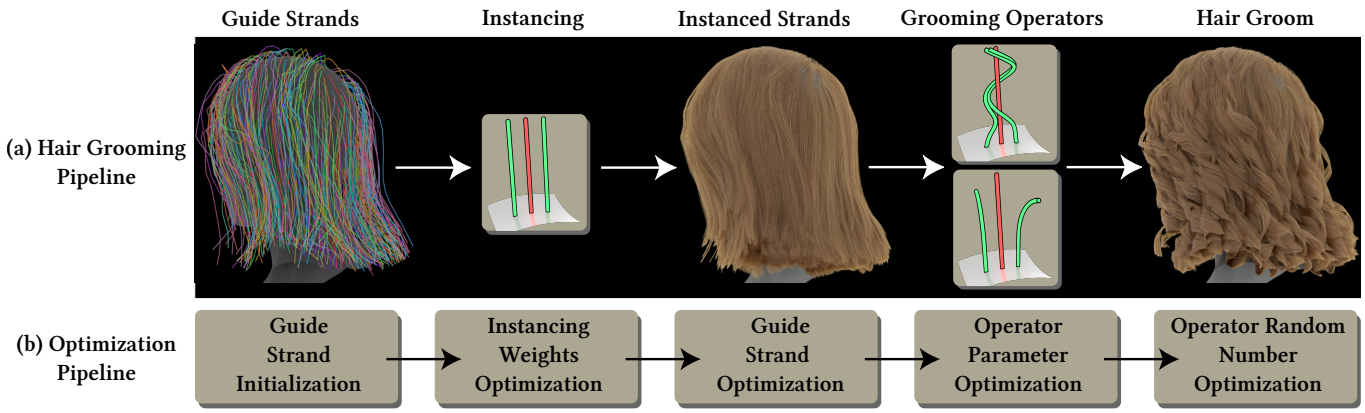


Fig. 2. Pipelines. (a) Our hair grooming pipeline takes a sparse set of guide strands, instances them to dense strands, and then procedurally deforms them using grooming operators to generate the hair groom. (b) To optimize the various parameters in the hair grooming pipeline, we initialize the guide strands using the target strands. We optimize the weights of the instancing operation and the shape of the guide strands. We then optimize both the operator parameters, which control the global hairstyle, as well as operator random numbers, to refine individual strands. Groom adapted from Hair Styles ©Daniel Bystedt.

2008; Sun et al. 2021]. Some methods directly synthesize hair by training a generative model [Chen et al. 2024; Zhou et al. 2023] or via examples [Wang et al. 2009; Weng et al. 2013].

Some of the methods above also reconstruct guide/coarse strands of the hair along the way [Chen et al. 2024; He et al. 2024; Luo et al. 2024; Takimoto et al. 2024; Zakharov et al. 2024]. However, these guide strands do not account for the procedural structure of the groom operators.

Our method is in principle agnostic to the type of method used for reconstruction and generation.

3 Overview

Problem setup. Our goal is to transform unstructured 3D hair strands (the target) into editable hair grooms. As such, we aim to recover a set of guide strands, as well as procedural grooming operators that deform the strands using a set of intuitive parameters.

We represent the strands in a hair groom as a collection of n polylines $\mathbf{x} = \{\mathbf{x}_1, \dots \mathbf{x}_n\}$, each with m vertices: $\mathbf{x}_i = \{x_{i,1}, \dots, x_{i,m}\}$, where n and m are determined by the target strands. In order to edit the groom more easily, we recover a sparse set of n_g ($500 \leq n_g \leq 3000$ in our work depending on the complexity of the target) guide strands $\mathbf{g}$, which are themselves strands with m vertices. The strands in the groom are then instanced by interpolating the guides, through an instancing operation: $\mathbf{x} = \text{instance}(\mathbf{g})$. The procedural grooming operators are parametric functions $P^{(z)}_{\boldsymbol{\theta}}$ that deform the strands: $\mathbf{x}^{(z+1)} = P^{(z)}_{\boldsymbol{\theta}}(\mathbf{x}^{(z)})$. An example is a bend operator, that is parameterized by a bending angle. Our procedural hair groom can therefore be described as:

$$\mathbf{x} = \mathbf{P}_{\boldsymbol{\theta}}(\text{instance}(\mathbf{g})), \tag{1}$$

where $\mathbf{P}_{\boldsymbol{\theta}}$ are the grooming operators applied to the instanced strands with parameters $\boldsymbol{\theta}$. $\mathbf{P}_{\boldsymbol{\theta}}$ can be written as composition of zero or more individual grooming operators $P^{(z)}_{\theta_z} \circ \cdots \circ P^{(1)}_{\theta_1}$. This forms a *chain* of operators. Production hair grooming tools often allow for highly expressive *graphs* of operators, but we implement a chain for simplicity. Figure 2 (a) illustrates this grooming pipeline, and we discuss it more in detail in Section 4.

The problem of recovering procedural hair grooms can then be posed as an optimization problem. Given target strands $\hat{\mathbf{x}}$, find the best set of guides $\mathbf{g}$ and grooming operator parameters $\boldsymbol{\theta}$ by solving the optimization problem:

$$\arg\min_{\mathbf{g},\boldsymbol{\theta}} \mathcal{L}(\mathbf{x}(\mathbf{g}, \boldsymbol{\theta}), \hat{\mathbf{x}}), \tag{2}$$

where $\mathcal{L}$ is a loss function. We denote target variables with $\hat{\cdot}$ to distinguish them from their estimated counterparts. In this work, we assume the set of operators is known, and only their parameters are optimized. We leave optimizing the topology of operators to future work. A major challenge of this work is the design of the loss $\mathcal{L}$ and how to optimize it.

Optimization Pipeline. Figure 2 (b) illustrates the overview of our optimization pipeline. We design dedicated strategies for the various parameters in the grooming pipeline and optimize each in turn. Our strategies are based off a probabilistic interpretation of the procedural hair grooms, discussed in Section 5.1. First, our method initializes guide strands by clustering and smoothing the target strands (Section 5.2). We then optimize the weights of the instancing operator to improve guide interpolation (Section 5.3). After, we optimize the guide strands without assuming correct grooming operator parameters (Section 5.4). Given the optimized guides, we optimize for the operator parameters that control the global hairstyle of the groom (Section 5.5), and per-strand random numbers to further refine the appearance of the groom (Section 5.6).

4 Hair Grooming Pipeline

Inspired by existing workflows in 3D modeling software, such as Blender [Blender-Online-Community 2018] and Houdini, we design a hair grooming pipeline that starts with guide strands, which are instanced to dense strands (Section 4.1). These instanced strands are further deformed by grooming operators (Section 4.2) to generate

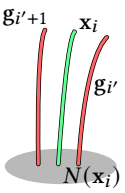
the full hair groom. Users can edit these grooms by modifying the guide strands and changing operator parameters.

4.1 Guides and Instancing

Since the average person can have a number of hair strands on the order of 10^5, to be able to design hair grooms more easily, artists frequently instead manipulate a sparse set of guide strands [Ward et al. 2007; Watanabe and Suenaga 1991], which reduce the number of strands by several orders of magnitude (typically in the hundreds, up to a few thousand). The final dense strands are generated by instancing the guide strands, using hair root positions and interpolating nearby guides. Mathematically, to generate strands $\mathbf{x}_i$, we take guides $\mathbf{g}_i$ and hair root positions $x_{i,1}$ and interpolate them as:

$$\bar{g}_{i,j} = \frac{\sum_{i' \in \mathrm{N}(\mathbf{x}_i)} w(x_{i,1}, g_{i',1}) g_{i',j}}{\sum_{i' \in \mathrm{N}(\mathbf{x}_i)} w(x_{i,1}, g_{i',1})}$$
$$x_{i,j} = x_{i,1} + \bar{g}_{i,j} - \bar{g}_{i,1}, \tag{3}$$

where $\mathrm{N}(\mathbf{x}_i)$ is the set of k closest guides to $\mathbf{x}_i$, and w weights the influence of each guide for each strand. By default, we choose a weight based on the distance of the root to the guide root in UV space: $w(x, g) = \exp\left(-||T(x) - T(g)||^2 / \sigma^2_{\text{instance}}\right)$ where $T : \mathbb{R}^3 \to [0, 1]^2$ is the UV-mapping of hair roots to texture coordinates on the scalp, as often computed in reconstruction work [Rosu et al. 2022], and σ_{instance} controls the size of the region of influence of each guide. However, for complex hairstyles seen in real hair strands, we can also opt to instead optimize for w, which we discuss in Section 5.3.



4.2 Operators

Figure 3 shows our grooming operators, each of which deforms strands procedurally. Some operators, particularly clump, bend, and curl, perform their deformation to follow an *operator* guide strand. In this section, we first discuss properties common to all operators, then describe each operator.

Operator guides. Some of the operators, namely clump, bend, and curl, deform the instanced strands around a guide strand to form their shape. However, instead of reusing the same guides used to instance them, which we refer to as simply the guide strands, we use a different set, which we call the *operator guides* $\mathbf{g}_{\text{op}}$. We choose the operator guides by sampling a fraction of the guide strands: $\mathbf{g}_{\text{op}} \subseteq \mathbf{g}$. In this way, the number of operator guides is not tightly coupled with the number of guide strands: the former is chosen based on the hairstyle (e.g., the number of curls in the groom), while the latter controls the resolution of the full dense groom. Each instanced strand is assigned only a single operator guide, as opposed to multiple guide strands in the guide interpolation. In this way, for the curl operator for example, all strands assigned to a particular operator guide will form a single curl around it. See Section 5.2 for more on operator guide sampling and assignment, and Figure 18 and Figure 19 for the effect of guides and operator guides.

Random parameters. Many of the operators not only have parameters controlling the deformation of all strands across the groom, but also *random* parameters which apply a random amount of deformation to each individual strand. For example, the curl operator has a curl frequency parameter θ and a random curl frequency parameter θ_{rand} so that the total frequency in a particular strand is $\theta + \theta_{\text{rand}} u$, where u is a random number. This provides a random variation to different strands to better model variation in real hair. See Appendix A for more details about the random numbers.

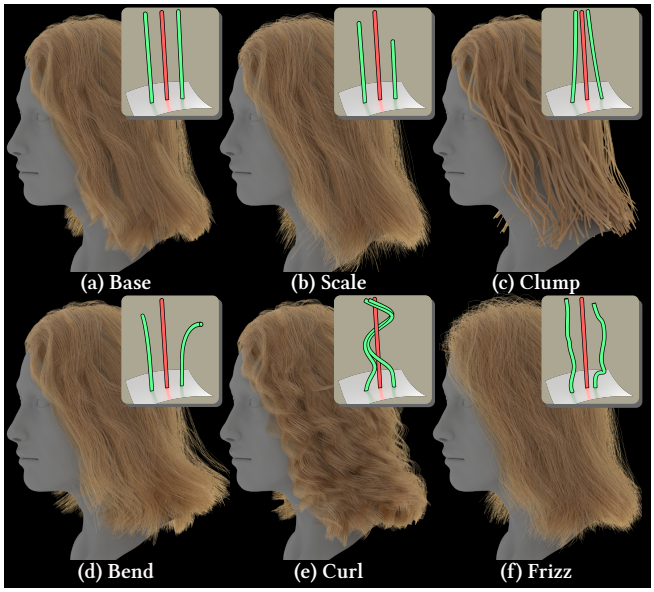


Fig. 3. Grooming Operators. (a) Given a base hair groom, (b)-(f) shows the effect of each individual operator. For each, a single red operator guide hair is shown in the middle, with two green instanced strands deformed by the operator. The render shows the overall hairstyle change due to each operator. Our pipeline combines operators to produce different grooms. Groom adapted from Hair Styles ©Daniel Bystedt.

Types. We provide full operator formulations in Appendix A.

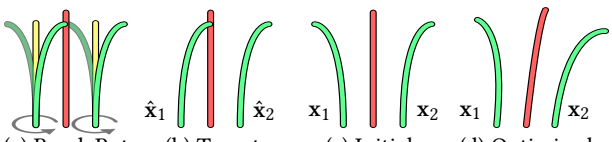
The scale operator (Figure 3 (b)) applies a random scaling to each instanced strand, parameterized by the random scale factor.

The clump operator (Figure 3 (c)) reduces the distance of each instanced strand to the operator guide towards the tip to form a hair clump shape. It blends the strand to the operator guide using a profile of the form $\exp(-ax)$ where a controls the profile shape.

The bend operator (Figure 3 (d)) bends each strand around a random axis perpendicular to the root direction of the operator guide, by an angle which increases linearly from root to tip. It is parameterized by the angle and bend start, which controls the distance along the strand to start the bend.

The curl operator (Figure 3 (e)) curls each strand around its operator guide. It is parameterized by the curl radius, frequency, a random frequency, which varies the frequency per curl, and curl start, which controls the distance along the strand to start the curl.

The frizz operator (Figure 3 (f)) applies Perlin noise [Perlin 1985] to each strand, to add noise to the groom. It is parameterized by the noise frequency and amplitude.



(a) Rand. Rot. (b) Target (c) Initial (d) Optimized

Fig. 4. Random Parameters and Naïve optimization. (a) The bend operator bends strands (green) by an angle with a random rotation around an axis specified by the operator guide strand (red). Note that if we average the strands of all possible rotations, the resulting strand (yellow) is similar in shape as the guide. (c) The initial state has the correct bend angle and guide strand shape as the target in (b). However, due to the different axis rotations, strands $\mathbf{x}_1, \mathbf{x}_2$ are different from $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$. (d) Naïvely minimizing the difference between $\mathbf{x}_1, \hat{\mathbf{x}}_1$ and $\mathbf{x}_2, \hat{\mathbf{x}}_2$ incorrectly bends the guide strand.

5 Optimization Strategies

To optimize the pipeline discussed in Section 4, we optimize different components in turn. Leveraging a probabilistic view on procedural hair grooms (Section 5.1), we initialize the guide strands (Section 5.2), optimize the instancing weights (Section 5.3), and then optimize the guide strands (Section 5.4). We further use the probabilistic view to optimize operator parameters (Section 5.5). Finally, to refine the groom to better match the target strands, we optimize the per-strand random numbers $\mathbf{u}$ in the operators (Section 5.6) as discussed in Section 4.2. We discuss structural integrity considerations when optimizing real hair grooms in Section 5.7. We provide details of the optimization including hyperparameters in Section 6.

5.1 A Probabilistic View on Procedural Hair Grooms

A core challenge in optimizing procedural hair grooms is that while the model contains a significantly smaller number of parameters for ease of editability, it typically cannot perfectly represent each individual strand in the target. Instead, the guide strands model the overall shape of the groom, while the operators add to the overall style and variation. Note that individual strand variation is added through the *random* operator parameters and their random numbers, and so a slightly different groom is generated depending on the random seed. As such, an optimization method needs to take into account a certain level of variance when comparing two strands: a strand at a given root location in the target does not always have the same shape as the strand at the same location in the procedural groom (See Figure 4). Rather, a local collection of strands in the model should, on average, look similar to the collection of strands in the target.

Thus, we turn to a probabilistic view on procedural hair grooms: given a set of random numbers $\mathbf{u}$, the grooming pipeline generates a hair groom, which is a realization from a *distribution* of hair grooms described by the guide strands and operator parameters. Using this, we rewrite Equation (1) in a per-strand form:

$$\mathbf{x}_i \sim \mathbf{P}_{\boldsymbol{\theta}}(\mathbf{g}, r_i), \tag{4}$$

where $r_i = x_{i,1}$ is the root location, and the instancing operation is hidden inside the set of operators $\mathbf{P}_{\boldsymbol{\theta}}$. In the next few sections, we use the idea that all strands, including target strands, can be seen as samples from a distribution $\mathbf{P}_{\boldsymbol{\theta}}$ to design various loss functions.

5.2 Guide Initialization

Guide clustering and smoothing. We manually select the number of guides n_g, and set the root locations from the target strands $r_i = \hat{x}_{i,1}$. Due to the variance in comparing strands, naïvely optimizing for the guides using, e.g., a vertex-wise L2 loss,

$$\mathcal{L}_{\mathbf{x}} = \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} (x_{i,j} - \hat{x}_{i,j})^2, \tag{5}$$

frequently causes variation to be baked into the guides, rather than the operators (Figure 4). Instead, we observe that the guide strands are similar to the *mean* of the distribution $\mathbf{P}_{\boldsymbol{\theta}}$: high-frequency variations are added using the operators, and the guides are similar to an average of all possible realizations of the strands (Figure 4 (a)). Thus, since strands are typically locally spatially similar besides the random variations, averaging a bundle of strands can be a guess for their guide strand. Therefore, we initialize the guide strands by taking the full set of n target strands, and applying K-means clustering with n_g clusters. The cluster centroids then form a sparse set of strands. However, these are not yet guide strands as they can still have, for example, a curl shape, and so we further smooth the strands to form our guide strands by solving the heat equation on the strands. In particular, given a strand $\mathbf{x}_i$, to smooth to a strand $\tilde{\mathbf{x}}_i$, we form the discrete Laplacian L:

$$[L]_{i,j} = \begin{cases} 1, & i = j = 1 \text{ or } i = j = m \\ 2, & i = j \text{ and } i, j \neq 1, i, j \neq m \\ -1, & j = i + 1 \text{ or } j = i - 1 \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

and iteratively solve $(I + \lambda_{\text{smooth}} L)\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)}$ for N_{smooth} steps, with fixed boundary conditions $\tilde{x}_{i,j'} = x_{i,j'}$, for $j' = \{1, \ldots, m_{\text{fixed}}\} \cup \{m\}$. In this way, we fix the first few vertices of the hair growing out of the scalp up to vertex m_{fixed} to prevent over-flattening at the top of the head, and the tip vertex to ensure the length of the strand does not shrink significantly during smoothing.

This is similar to the K-medoids clustering and discrete cosine transform (DCT) decomposition used in concurrent work by Chen et al. [2024], with the difference that we do not need the guide strands to be a subset of the target strands and directly solving the heat equation allows for more flexible boundary conditions.

Operator guide selection and assignment. To select the operator guides $\mathbf{g}_{\text{op}} \subseteq \mathbf{g}$, we select them by applying K-medoids clustering on $\mathbf{g}$ (where $n_{g_{\text{op}}}$ is also selected manually based on the groom). Note that we use K-medoids clustering here, as we choose the operator guides to be a subset of the guides (this does not have to be the case in general), and so we need the resulting medoids to be from the set of guides. To assign an operator guide to each instanced strand, by default we choose the closest operator guide based on root UV distance. This assignment may occasionally be problematic near hair-parting regions, where an instanced strand's blended guide $\bar{\mathbf{g}}_i$ and operator guide $\mathbf{g}_{\text{op}_i}$ are on different sides of the parting. This can cause the instanced strand to penetrate the head. To fix this, we reassign the instanced strand the next closest operator guide that does not cause head penetration.

5.3 Instancing optimization

We find that in real hair grooms, distance-based interpolation methods are insufficient, as root distance is a poor metric of hair similarity, which is in line with previous work [Zhou et al. 2023]. Target strands that are not well interpolated using a distance-based interpolation make it difficult to optimize for the guide strands discussed in the next section. Thus, we optimize the per-strand interpolation weights w in Equation (3) with a vertex-wise L2 loss (Equation (5)), i.e. we find $\arg\min_{w_{i,i'}} \mathcal{L}_{\mathbf{x}}$, where $w(x_{i,1}, g_{i',1}) = w_{i,i'}$, using gradient-based optimization. For this stage only, we use the non-smoothed cluster centroids as the guide strands with no grooming operators to optimize only for instancing. This is done as at this point, we do not know what the operator parameters should be, and so we let the guide strands fully describe the shape of the groom. With optimized weights, each strand can choose the weighting of local neighboring guides to best fit the target shape, without affecting the editability of the strands.

5.4 Guide Optimization

Loss function. To further refine the shape of the guides, we aim to design a loss that is robust to individual strand variation. We start by assuming the target can be perfectly represented using a procedural model: $\hat{\mathbf{x}}_i \sim \mathbf{P}_{\hat{\boldsymbol{\theta}}}(\hat{\mathbf{g}}, \hat{r}_i)$. In this case, using the idea that the guides are similar to the mean of $\mathbf{P}_\theta$, we can use a loss that minimizes expectations:

$$\mathcal{L}_{\mathbf{g}} = \frac{1}{n}\sum_{i=1}^{n} \left\| \mathbb{E}_{\mathbf{u}_i}[\mathbf{P}_{\boldsymbol{\theta}}(\mathbf{g}, r_i)] - \mathbb{E}_{\hat{\mathbf{u}}_i}[\mathbf{P}_{\hat{\boldsymbol{\theta}}}(\hat{\mathbf{g}}, \hat{r}_i)] \right\|^2, \tag{7}$$

where the expectation is taken over all random numbers $\mathbf{u}_i$ for each strand i. By taking the expectation of the strand, we aim to remove the random strand variations caused by the operators, revealing the overall shape due to the guides. To minimize this loss, we first assume $\bar{\mathbf{x}}_i = \mathbb{E}_{\mathbf{u}_i}[\mathbf{P}_{\boldsymbol{\theta}}(\mathbf{g}, r_i)]$ can be computed. This can either be done via Monte Carlo sampling with high sample count, or we find in most cases, well-approximated by operators with a known set of parameters. For example, the expectation of the bend operator for any angle is simply the identity operation. This is the case for most of our operators besides clump. We can then use a gradient-based approach and simplify the gradient of $\mathcal{L}_g$ to be:

$$\begin{aligned}
\frac{\partial \mathcal{L}_{\mathbf{g}}}{\partial \mathbf{g}} &= \frac{2}{n}\sum_{i=1}^{n} (\mathbb{E}_{\mathbf{u}_i}[\mathbf{P}_{\boldsymbol{\theta}}(\mathbf{g}, r_i)] - \mathbb{E}_{\hat{\mathbf{u}}_i}[\mathbf{P}_{\hat{\boldsymbol{\theta}}}(\hat{\mathbf{g}}, \hat{r}_i)]) \frac{\partial}{\partial \mathbf{g}} \mathbb{E}_{\mathbf{u}_i}[\mathbf{P}_{\boldsymbol{\theta}}(\mathbf{g}, r_i)] \\
&= \frac{2}{n}\sum_{i=1}^{n} (\bar{\mathbf{x}}_i - \mathbb{E}_{\hat{\mathbf{u}}_i}[\mathbf{P}_{\hat{\boldsymbol{\theta}}}(\hat{\mathbf{g}}, \hat{r}_i)]) \frac{\partial}{\partial \mathbf{g}} \bar{\mathbf{x}}_i \\
&= \frac{2}{n}\sum_{i=1}^{n} \mathbb{E}_{\hat{\mathbf{u}}_i}\left[(\bar{\mathbf{x}}_i - \mathbf{P}_{\hat{\boldsymbol{\theta}}}(\hat{\mathbf{g}}, \hat{r}_i)) \frac{\partial}{\partial \mathbf{g}} \bar{\mathbf{x}}_i \right],
\end{aligned} \tag{8}$$

by linearity of expectation. In practice, the target is not a procedural model, and so we only have access to a single realization $\hat{\mathbf{x}}_i \sim \mathbf{P}_{\hat{\boldsymbol{\theta}}}(\hat{\mathbf{g}}, \hat{r}_i)$. Thus, we cannot compute Equation (8) as the expectation cannot be computed. If we instead only compute the gradient using the single realization, the result tends to be noisy and cause variation to be baked into the guide strands. To avoid this, we observe that the guide strands tend to be smooth across the scalp, i.e. neighboring guides typically do not differ significantly. We can therefore use the single realization and apply a filtering operation $\mathcal{D}$ across the guide gradients:

$$\frac{\partial \mathcal{L}_{\mathbf{g}}}{\partial \mathbf{g}} \approx \frac{2}{n} \mathcal{D}\left(\sum_{i=1}^{n} (\bar{\mathbf{x}}_i - \hat{\mathbf{x}}_i) \frac{\partial}{\partial \mathbf{g}} \bar{\mathbf{x}}_i \right), \tag{9}$$

where $\mathcal{D}$ is a filtering operation *across* the gradients of *different* guide strands. In particular, we use mesh bilateral gradient filtering [Chang et al. 2024] as $\mathcal{D}$ to optimize towards smooth guides. This is done by triangulating [Barber et al. 1996] the guide roots in UV space $T(g_{i,1})$ to obtain a connectivity for the guides. We use the guide root directions $\frac{g_{i,2} - g_{i,1}}{||g_{i,2} - g_{i,1}||}$ for the bilateral data term.

Parameterization. To ensure smoothness along each guide strand, in addition to gradient filtering, we use the curvature parameterization by Crane et al. [2013] instead of directly optimizing vertex positions. They show that parameterizing based on curvature minimizes a fairing/bending energy $\frac{1}{2}\int \boldsymbol{\kappa}^2 \, \mathrm{d}l$, where $\boldsymbol{\kappa}$ is the curvature, leading to more robust and higher quality curve evolution. In particular, this parameterization includes the root position $x_{i,1}$ (which we do not optimize), root angles $\phi_{i,1}$ (in spherical coordinates), segment lengths $l_{i,j}$, and curvatures $\kappa_{i,j}$. We then integrate curvature to get segment directions, and integrate directions to get vertex positions:

$$\begin{aligned}
\phi_{i,j} &= \phi_{i,1} + \sum_{j'=1}^{j-1} \frac{1}{2}(l_{i,j'+1} + l_{i,j'})\kappa_{i,j'} \\
x_{i,j} &= x_{i,1} + \sum_{j'=1}^{j-1} l_{i,j'} d_{i,j'}(\phi_{i,j'}),
\end{aligned} \tag{10}$$

where $d_{i,j}$ is the segment direction in Cartesian coordinates.

5.5 Operator Parameter Optimization

Unlike guides, each of which modify the shape of strands in a local neighborhood, operators globally modify the groom style. As a result, instead of having only the neighborhood of strands to determine guide shape, we can use all strands simultaneously to determine the operator parameters. If we view the estimated strands $\mathbf{x}$ as n samples from the distribution $\mathbf{P}_{\boldsymbol{\theta}}$ (note this is not strictly a distribution, as each strand's $\mathbf{x}_i$ depends on root location r_i, but we find this is fine in practice) and the target strands $\hat{\mathbf{x}}$ as n samples from $\mathbf{P}_{\hat{\boldsymbol{\theta}}}$, then the task becomes finding parameters $\boldsymbol{\theta}$ to best match two sets of samples from two distributions. We categorize the operators based on *how* they affect the distribution: strand shape operators largely modify the shape of each strand, while strand correlation operators modify the correlation across strands. The two are optimized separately with different strategies.

Strand shape operators. The bend, curl, and frizz operators modify the *shape* along each strand. Like for the guides, optimizing using a vertex-wise loss is insufficient due to the variance inherent to the comparison between two strands as discussed previously. Instead, a better approach is to use a loss that attempts to match the distributions $\mathbf{P}_{\boldsymbol{\theta}}$ and $\mathbf{P}_{\hat{\boldsymbol{\theta}}}$ given their samples. In particular, instead of comparing strands directly, we compare them using the Sliced Wasserstein Distance [Rabin et al. 2012], based in optimal transport

theory for comparing distributions, which has been used for various tasks such as color transfer and texture synthesis [Heitz et al. 2021; Pitie et al. 2005]. Optimal transport has also been used to compute strand correspondences for example-based hair geometry interpolation [Weng et al. 2013]. Following the analysis of Heitz et al. [2021], we use a Sliced Wasserstein Loss of the form:

$$\mathrm{SW}\left(\mathbf{f}, \hat{\mathbf{f}}\right) = \frac{1}{s}\sum_{k=1}^{s}\left\|\mathrm{sort}(\{\langle \mathbf{v}_k, \mathbf{f}_i\rangle\}) - \mathrm{sort}(\{\langle \mathbf{v}_k, \hat{\mathbf{f}}_i\rangle\})\right\|^2, \tag{11}$$

where $\mathbf{f} = h(\mathbf{x})$ are features computed from the strands, $s = \dim \mathbf{f}_i$ and $\mathbf{v}_k$ are random vectors. For each set of n strand features, the loss first takes a slice of each s-dimensional feature, by projecting it to a scalar through a dot product with $\mathbf{v}_k$, then sorts the set of n projected scalars. A distance is then computed between the two sets of sorted projections using the L2 norm. We refer readers to the work by Heitz et al. [2021] for more information. Intuitively, this loss implicitly computes a correspondence between the features via the projection and sorting in order to match the distributions (See Figure 5).

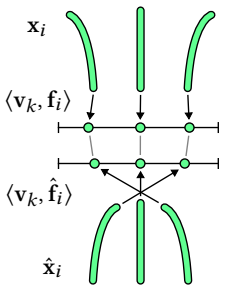


Fig. 5. Sliced Wasserstein Loss. To compare two sets of hair strands $(\mathbf{x}_i, \hat{\mathbf{x}}_i)$ without known strand correspondences, each strand's features $(\mathbf{f}_i, \hat{\mathbf{f}}_i)$ are projected to scalars $(\langle \mathbf{v}_k, \mathbf{f}_i\rangle, \langle \mathbf{v}_k, \hat{\mathbf{f}}_i\rangle)$. Both sets are sorted and compared using an L2 loss.

To optimize a variety of operators, we use the combination of a frequency loss and a strand segment loss:

$$\mathcal{L}_\theta = \mathrm{SW}\left(\left|\mathcal{F}\{\mathbf{e}_i\}\right|, \left|\mathcal{F}\{\hat{\mathbf{e}}_i\}\right|\right) + \lambda_e \mathrm{SW}\left(\mathbf{e}_i, \hat{\mathbf{e}}_i\right), \tag{12}$$

where $\mathbf{e}_i = \{e_{i,1}, \ldots, e_{i,m-1}\}$, $e_{i,j} = x_{i,j+1} - x_{i,j}$ are strand segments, and $|\mathcal{F}|$ is the absolute value of the 1D Fourier transform computed independently for each coordinate, i.e. $|\mathcal{F}\{\mathbf{e}_i\}| = \left\{\left|\mathcal{F}\{\mathbf{e}_i^c\}\right|\right\}$, where $\mathbf{e}_i^c = \{e_{i,1}^c, \ldots, e_{i,m-1}^c\}$, for $c = \{\mathrm{x}, \mathrm{y}, \mathrm{z}\}$. Intuitively, the second term in the loss measures the global shape of each strand, while the first term transforms strands to the frequency domain to better capture periodic shapes like frizz and curl.

Strand correlation operators. The scale and clump operators affect the *correlation* across strands: the scale operator randomly scales each strand so that neighboring strands have different lengths, while the clump operator pushes neighboring strands together around the guide. To measure strand correlations, we turn to Determinantal Point Processes (DPP) [Hough et al. 2006; Kulesza 2012; Macchi 1975], which characterize a set of points from a DPP using the determinant of a positive semidefinite kernel K. We omit the formal math of DPPs for brevity and refer interested readers to the introductions provided by Hough et al. [2006] and Kulesza [2012]. Intuitively, given a set of points $\mathbf{y} = \{y_1, \ldots y_n\}$, if the entries $[K]_{a,b} = \psi(y_a, y_b)$ measure a similarity between y_a and y_b, then $\det(K)$ can be used to measure the correlation between the points in $\mathbf{y}$: the determinant is large when points are distributed far apart and is small when points are clumped together. Thus, we can use this to define a loss:

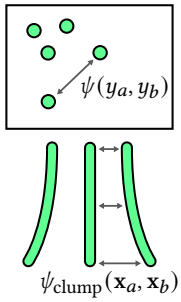


Fig. 6. DPP Loss. DPP theory enables quantifying point clustering through pairwise similarities $\psi(y_a, y_b)$. To measure hair clumpiness, we define a ψ_{clump} based on the distance between strands from root to tip.

$$\mathrm{DPPL}_\psi\left(\mathbf{x}, \hat{\mathbf{x}}\right) = \left(\log(\det[\psi(\mathbf{x}_a, \mathbf{x}_b)]) - \log(\det[\psi(\hat{\mathbf{x}}_a, \hat{\mathbf{x}}_b)])\right)^2, \tag{13}$$

where ψ is a suitable measure of similarity between strands, and log is used for robustness against numerical precision issues. For the scale operator, we compare strand lengths:

$$\psi_{\mathrm{scale}}(\mathbf{x}_a, \mathbf{x}_b) = \exp(-(l_a - l_b)^2/\sigma_{\mathrm{scale}}) \tag{14}$$

$$l_a = \sum_{j=1}^{m-1}\left\|x_{a,j+1} - x_{a,j}\right\|, \tag{15}$$

while for the clump operator, we compare a type of relative distance between strand vertices:

$$\psi_{\mathrm{clump}}(\mathbf{x}_a, \mathbf{x}_b) = \exp(-\mathrm{CD}(\mathbf{x}_a, \mathbf{x}_b)/\sigma_{\mathrm{clump}}) \tag{16}$$

$$\mathrm{CD}(\mathbf{x}_a, \mathbf{x}_b) = \frac{1}{m}\sum_{j=1}^{m}\frac{\left\|x_{a,j} - x_{b,j}\right\|}{\left\|x_{a,1} - x_{b,1}\right\|}, \tag{17}$$

where CD decreases as strands meet towards the tip (See Figure 6).

Computing this loss exactly is prohibitively expensive, as it requires forming K which is $O(n^2)$ in the number of strands, where n can be on the order of 10^5. Although it is possible to form a sparse matrix by setting elements in the matrix to 0 when strands have low similarities, we instead take the simpler approach of sampling a subset of strands $B_K \subseteq \{1, \ldots, n\}$. To sample this subset, we first select a random strand $\mathbf{x}_i$, and then select the nearest $|B_K| - 1$ strands in UV space. We then have $|B_K|$ strands in a local region of the head, which we use to compute K. Our losses are then

$$\mathcal{L}_{\theta_{\mathrm{scale}}} = \mathrm{DPPL}_{\psi_{\mathrm{scale}}}\left(\mathbf{x}_{B_K}, \hat{\mathbf{x}}_{B_K}\right) \text{ and } \mathcal{L}_{\theta_{\mathrm{clump}}} = \mathrm{DPPL}_{\psi_{\mathrm{clump}}}\left(\mathbf{x}_{B_K}, \hat{\mathbf{x}}_{B_K}\right). \tag{18}$$

Since we have losses specific to scale and clump, we alternate optimizing them.

5.6 Operator Random Number Optimization

While procedural hair grooming enables editability, a downside of the methods in the previous sections is that only the global characteristics of the groom are captured; local hair orientations may not be captured. Note that this is not specific to hair grooming, matching local features is a problem for other procedural modeling tasks (e.g., [Hu et al. 2019; Talton et al. 2011]) as well. To ameliorate this and improve the visual appearance of our optimized groom compared to the target, we propose to optimize the per-strand random numbers $\mathbf{u}_i$. Similar to the previous section, we use a combination of a frequency loss, direction loss, and this time a length loss as well

for the scale operator:

$$\mathcal{L}_{\theta_{\text{rand}}} = \frac{1}{n}\sum_{i=1}^{n}\left\|\left|\mathcal{F}\{\mathbf{d}_i\}\right| - \left|\mathcal{F}\{\hat{\mathbf{d}}_i\}\right|\right\|^2 + \frac{\lambda_{d_{\text{rand}}}}{n}\sum_{i=1}^{n} 1 - \langle\langle \mathbf{d}_i, \hat{\mathbf{d}}_i\rangle\rangle + \frac{\lambda_{l_{\text{rand}}}}{n}\sum_{i=1}^{n}(l_i - \hat{l}_i)^2, \quad (19)$$

where $\langle\langle\cdot,\cdot\rangle\rangle$ computes the dot product between each element of $\mathbf{d}_i$ and $\hat{\mathbf{d}}_i$, and $d_{i,j} = \frac{e_{i,j}}{||e_{i,j}||}$. This time we use an element-wise loss in order to optimize local features.

Using gradient-based optimization with this loss often gets stuck in local minima, as the procedural operators are typically very non-linear functions. As such, to help escape local minima, we add noise to the gradients during the optimization step:

$$\mathbf{u}_i^{(t+1)} = \mathbf{u}_i^{(t)} - \alpha\left(\frac{\partial \mathcal{L}_{\text{rand}}}{\partial \mathbf{u}_i} + \sigma_i^{(t)}\varepsilon\right), \quad (20)$$

where t is the iteration number, α is the learning rate, $\sigma_i^{(t)}$ are parameter and iteration dependent noise scaling factors, and each $\varepsilon \sim \mathcal{N}(0,1)$, the standard normal distribution. To automatically choose a $\sigma_i^{(t)}$ which is adaptive to the scale of the parameter gradients across a wide range of different parameters and grooms, we scale it according to the mean absolute gradient $\frac{1}{n}\sum_{i=1}^{n}\left|\frac{\partial \mathcal{L}_{\text{rand}}}{\partial \mathbf{u}_i}\right|$. To aid in jumping out of local minima, we additionally scale it by a factor of 1.5, which linearly decays to 0.5 at the end of the optimization after N_{rand} iterations. Altogether, we set:

$$\sigma_i^{(t)} = \left(1.5 - \frac{t-1}{N_{\text{rand}}-1}\right)\frac{1}{n}\sum_{i=1}^{n}\left|\frac{\partial \mathcal{L}_{\text{rand}}}{\partial \mathbf{u}_i}\right|, \quad (21)$$

for $t = 1, \ldots, N_{\text{rand}}$. Adding noise to the gradients has shown similar benefits in recent inverse rendering tasks [Fischer and Ritschel 2023; Kheradmand et al. 2024].

5.7 Structural Integrity

A challenge of unstructured strands obtained from reconstruction work (e.g., [Sklyarova et al. 2023; Wu et al. 2024]) is that they often have little structural guarantees. For example, the inner strands that are not visible to the camera need to be inferred from data-driven priors, which can often be lacking for complex grooms like curly hair. As a result, inner hair which should be curly may end up being too short or too straight. On the other hand, our instancing and operators guarantee some structure: all strands that are part of a curl are guaranteed to be curly. Furthermore, the gradient filtering and curvature parameterization make it unlikely for the strands to have sharp bends or change drastically spatially.

When the target strands $\hat{\mathbf{x}}$ have poor structure (e.g., short, straight strands in a curly groom), using them in the optimization can affect the recovered parameters. For example, straight strands in a curly groom will bias the curl operator radius and frequency to lower values. To address this, we remove these flawed strands from the optimization. In particular, if we denote $B_{\text{flawed}} \subset \{1, \ldots, n\}$ as indices of the flawed strands, then the strands we use during optimization are $\mathbf{x}_{\text{opt}} = \mathbf{x} \setminus \mathbf{x}_{B_{\text{flawed}}}$. Then, our loss functions become $\mathcal{L}(\mathbf{x}_{\text{opt}}, \hat{\mathbf{x}}_{\text{opt}})$. We choose $\mathbf{x}_{B_{\text{flawed}}}$ to be the strands that are not visible to the camera. We select these by taking the cameras used by the original reconstruction method, rasterizing the target strands, and removing any that are not visible from any view.

Note that even if strand $\hat{\mathbf{x}}_i$ is not used in the optimization, strand $\mathbf{x}_i$ can still be optimized due to our grooming pipeline. Neighboring target strands $\hat{\mathbf{x}}_{i'}$ that *are* used in the optimization will contribute gradients to $\mathbf{x}_{i'}$, which will be propagated to the guides, which are then instanced to $\mathbf{x}_i$. Even when no strands for a particular guide strand receive gradients, the gradient filtering ensures that the guide is still optimized to smoothly follow neighboring guides.

6 Implementation

We implement our system using PyTorch. We use Warp [Macklin 2022] to compute various distance and closest point queries from the hair to the head mesh and Faiss [Douze et al. 2024; Johnson et al. 2019] as well as Scikit-Learn for various clustering and nearest neighbor queries. The time it takes our method to optimize a single groom depends on the number of strands and operators. We list the times for each example run on a desktop with an i9-14900K CPU and an RTX 4090 GPU in Table 1.

Preprocessing Details. For synthetic data, we fit a FLAME model to the head mesh as described by Li et al. [2017]. We optimize the FLAME model shape and pose parameters by minimizing the L2 distance between a set of 51 facial landmarks, manually placed on the template mesh and the head mesh, and the L2 distance between the vertices of the template mesh and their closest point on the head surface. With the FLAME head mesh, we can then extract the scalp region from the parametric mesh in Blender, and generate a UV map, which is used to compute hair root distances and root triangulations. For the real examples from MonoHair and Gaussian Haircut, we directly use their fitted FLAME parametric head model. We also use their fitted camera parameters to extract the non-visible set of hair strands, as discussed in Section 5.7.

Initialization Details. We list the number of guides n_g and operator guides $n_{g_{\text{op}}}$ we choose for each example in Table 1. For the guide smoothing, we set $\lambda_{\text{smooth}} = 1$ and $N_{\text{smooth}} = 80$ by default and $m_{\text{fixed}} = \lfloor 0.05m \rfloor$. To initialize the grooming operators, we manually select a set of operators based on the hairstyle of the subject, and choose initial parameters within common ranges.

Optimization Details. For optimization, we follow the pipeline as discussed in Section 3.

We optimize the instancing weights for 2000 iterations with Adam with a learning rate of $3 \cdot 10^1$ and default PyTorch settings.

We optimize for the guides for 3000 iterations. We use the optimizer provided by [Chang et al. 2024], with learning rates of $5 \cdot 10^{-2}$ for $\kappa_{i,j}$, $1.5 \cdot 10^{-3}$ for $l_{i,j}$, and $5 \cdot 10^{-2}$ for $\phi_{i,1}$. We set $\lambda = 10$ and $\sigma_d = 0.3$. The remaining parameters follow their defaults.

For the operators, we alternate between optimizing the 3 different losses for: the strand shape operators, the scale operator, and the clump operator, each for 200 iterations at a time. If a groom does not have the particular operator, we skip optimizing it. We optimize each for a total of 3000 iterations, up to a total of 9000 iterations for the operator optimization stage. For all, we use the Adam optimizer [Kingma and Ba 2015] with a learning rate of $1 \cdot 10^{-2}$ and default PyTorch settings. We set $\lambda_{\mathbf{e}} = 4 \cdot 10^2$. For the DPP loss, we set

Table 1. Example Details. We list the local taxonomic labels [Meishvili et al. 2024], strand information, and optimization time for our examples. We cover most of the variations in each label, showcasing the diversity of hairstyles that our method supports.

Examples	Hair Type	Gathered	Direction	Length	Layering	n	m	n_g	$n_{g_{\text{op}}}$	Opt. Time (min)
Figure 7 (a)	wavy	none	down	chin	none	28022	101	500	500	10
Figure 7 (b)	curly	behind-ear	down	shoulder	none	13991	101	500	500	3
Figure 7 (c)	straight	none	up	short/shaved	fade	59821	101	500	500	30
Figure 7 (d)	straight	none	out	ear	none	28022	101	500	500	6
Figure 7 (e)	straight	none	down	shoulder	none	13991	101	500	500	1
Figure 7 (f)	coil	none	out	very short	none	21325	101	3000	3000	7
Figure 7 (g)	straight	none	down	shoulder	none	13991	101	500	500	2
Figure 8 (a)	wavy	behind-ear	down	chin	none	166280	100	500	125	32
Figure 8 (b)	curly	none	down	chin	none	30000	100	1000	250	13
Figure 8 (c)	straight	none	side	short	taper	30000	100	1000	1000	8
Figure 8 (d)	straight	ponytail	down	chin	none	30000	100	1000	1000	8
Figure 8 (e)	straight	none	down	short	none	30000	100	1000	1000	8
Figure 8 (f)	straight	behind-ear	down	ear	textured	30000	100	500	125	4
Figure 8 (g)	wavy	none	down	shoulder	none	30000	100	500	500	4

$\sigma_{\text{scale}} = \sigma_{\text{clump}} = 10^{-1}$ and $|B_k| = \min(n/100, 200)$. In practice, this can have high variance, and so at every iteration we compute the loss 5 times for 5 different subsets for the first 3/4 of the optimization, which we increase to 50 for the last 1/4.

For the operator random number stage, we optimize for $N_{\text{rand}} = 3000$ iterations. We use the same optimizer as the operator stage, but increase the learning rate to $1 \cdot 10^{-1}$. We set $\lambda_{d_{\text{rand}}} = 2$ and $\lambda_{l_{\text{rand}}} = 1 \cdot 10^{-2}$.

7 Results

To validate our method, we test on 14 different synthetic and real-world grooms, selected to cover a variety of operators and hairstyles.

7.1 Hairstyle Diversity

We demonstrate the diversity of our examples by evaluating them against the comprehensive hairstyle taxonomy proposed by Meishvili et al. [2024]. This taxonomy provides a universal description that captures the diversity and full range of hairstyles. In Table 1, we show the taxonomic labels for our 14 examples and demonstrate that they cover the full variation in most of the labels. Our examples have full coverage in Hair Type, Direction, and Layering labels, covering all variations for each of these labels. For the Length label, we choose five most representative hair lengths excluding *bald*, as well as lengths that are longer than the *shoulder*, since they do not pose unique challenges for us. The Decoration label is omitted for the same reason. For the Gathered label, we focused on *behind the ear* and *ponytail* examples, since the other variations (*buns* and *knots*) mostly stay static so they are out of scope for the applications of our method. We omit the Strand Styling label and the associated Strand Thickness as our method currently does not support the reconstruction of *dreadlocks* or *braids*. To make these hairstyles editable, we would need specialized dreadlock and braid operators. Additionally, we only annotate our examples with local taxonomic labels, as global labels such as Bang Style and Hair Parting do not present unique challenges for reconstruction and are therefore not prioritized in our evaluation.

7.2 Evaluation on Synthetic Procedural Grooms

To validate our method, we test on 7 synthetic grooms as seen in Figure 7. Each groom was generated using our pipeline, with guide strands designed by artists in Blender. We assume only the target strands $\hat{\mathbf{x}}$ and the set of grooming operators are given. We optimize for the guides as well as operator parameters. We compare the target groom and ground truth guides to our procedural grooms and recovered guides, and also show the parameter values and their errors over optimization. Note that we do not assume the number of guides are known, and so it is expected that there are slight differences in the guide shapes and parameter values. Our method typically reconstructs accurate guides with low parameter errors. In some cases, such as the last groom in Figure 7, when the operators significantly deform the strands over the guides (high bend angle), the high amount of random variation caused by the random parameters causes the reconstruction to be less accurate.

7.3 Evaluation on Real-world Examples

To evaluate our method on real-world examples, we transform 7 subjects whose hair strands are reconstructed using Gaussian Haircut [Zakharov et al. 2024] and MonoHair [Wu et al. 2024]. We test on a variety of hairstyles from the monocular dataset [Sklyarova et al. 2023], the MonoHair dataset [Wu et al. 2024], and a custom individual. Figure 8 shows our results, where we show renders in Cycles, as well as a visualization of the strands and guides. The example in Figure 8 (a), generated using MonoHair, is a particularly challenging example for us to optimize due to the poor structural integrity of the target (unnatural inner strands, see Figure 14).

7.4 Ablations and Evaluation

Figure 9 shows qualitative ablations on guide initialization and optimization, and operator parameter and random number optimization. We also provide further detailed ablations and evaluations of our pipeline in Appendix B.

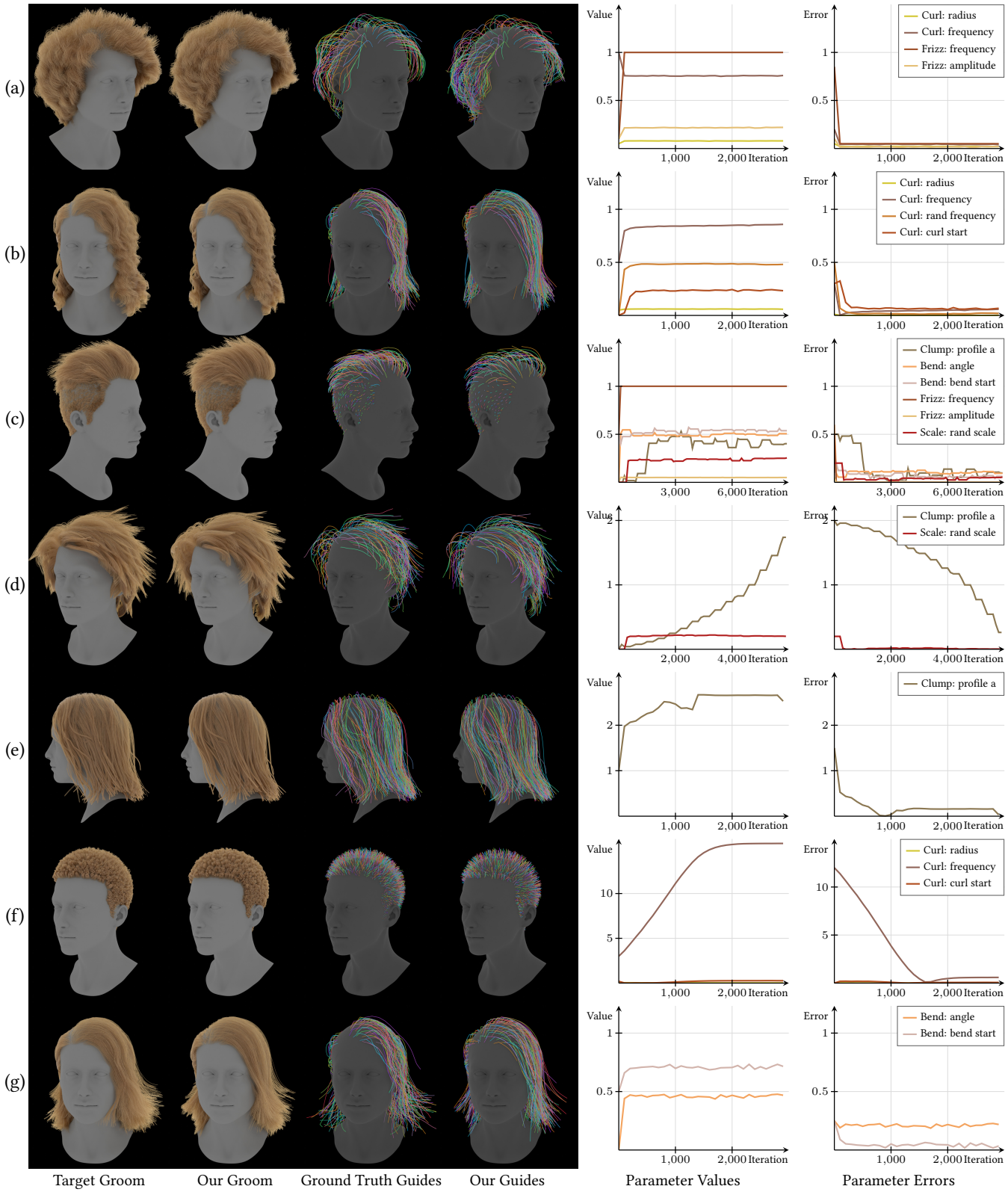


Fig. 7. Synthetic Grooms. We recover the guide strands and operator parameters of 7 synthetic grooms. We visualize the recovered procedural groom, guide strands, and show parameter values and errors over optimization iterations. Grooms adapted from Hair Styles ©Daniel Bystedt.

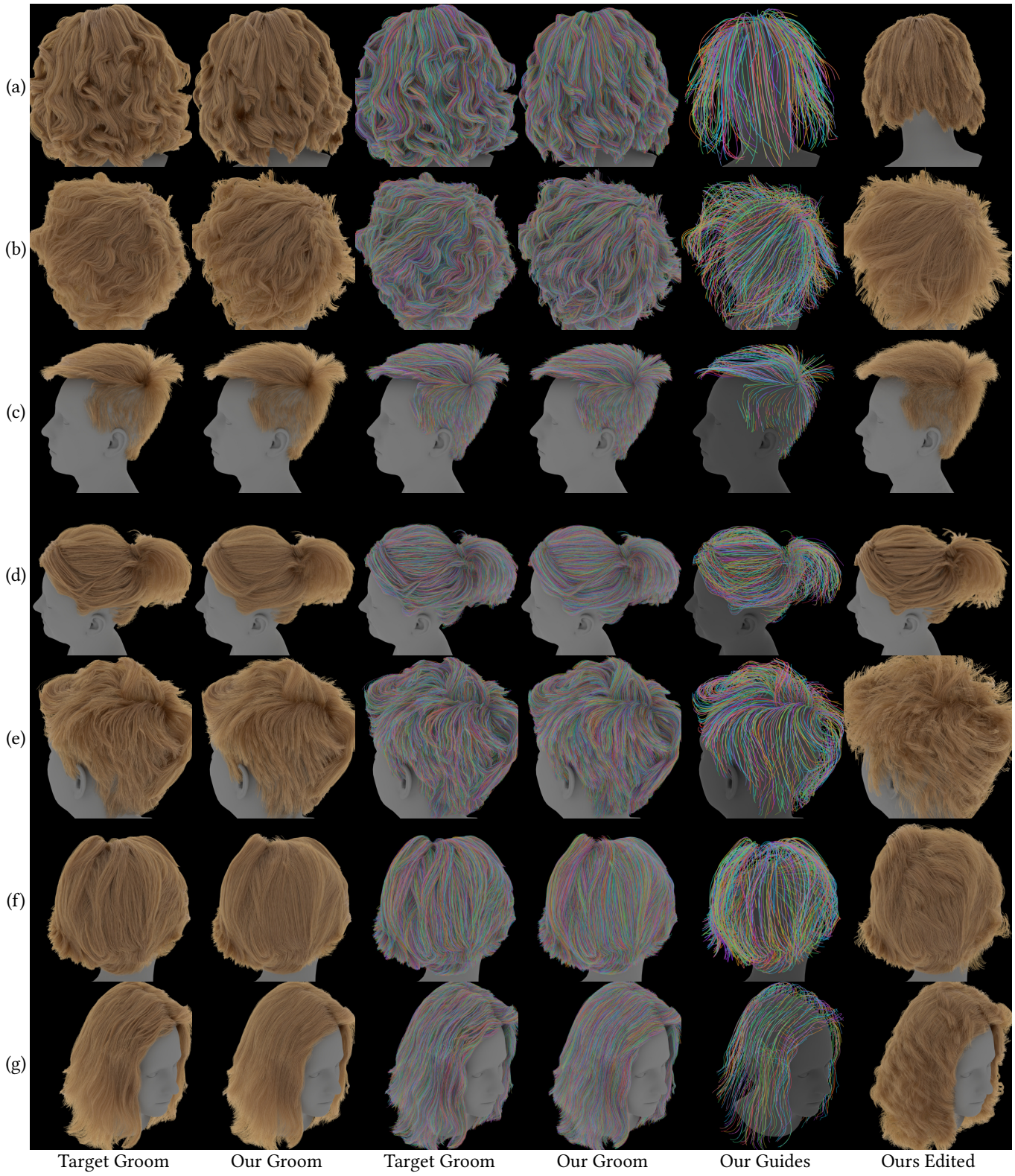


Fig. 8. Real Grooms. We recover the guide strands and operator parameters of 7 grooms from previous reconstruction work. We visualize the non-procedural target grooms and our recovered grooms and guides. We also show a simple procedural edit for each.

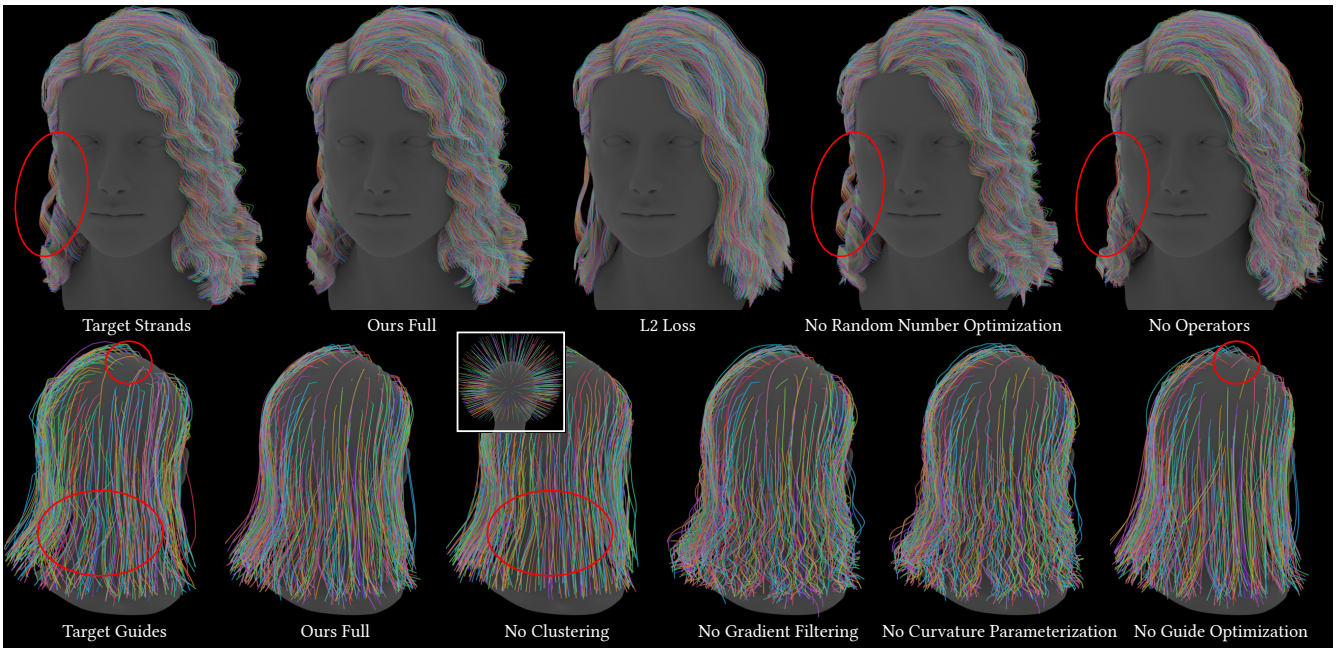


Fig. 9. Ablations. We show ablations for our operator parameter and random number optimization stages (top) and guide initialization and optimization (bottom). In the top row, we assume known guide strands. Our full method is able to recover near perfect strands. Using an L2 loss instead of our operator losses fails to recover the correct operator parameters, leading to incorrect curliness. Not applying our random number optimization leads to correct operator parameters, but incorrect local strand orientations as shown by the ellipse. If we do not have a curl operator and simply optimize guides and instancing, we can obtain a curly groom, but local strand orientations are incorrect, and the curliness cannot be edited. In the bottom row, we do not assume anything is known. If we do not initialize using clustering and instead optimize from a straight initialization shown in the inset, the result is overly smooth as shown in the ellipse. In contrast, ignoring either the gradient filtering or curvature parameterization leads to guides with baked variation. Not applying the guide optimization at all can cause issues like a guide penetrating the head, as shown in the ellipse, when the clustering initialization is not optimal. Groom adapted from Hair Styles ©Daniel Bystedt.

7.5 Editing and Simulation

The last column of Figure 8 shows edits for each of the real examples, changing multiple operator parameters. For more fine-grained edits, Figure 10 (i)-(vi) show edits to the groom from Figure 1. We edit the guides as well as change a few operator parameters to show the different types of grooms that can be achieved with our grooming pipeline. Additionally, since our hair grooms have reasonable hair even in non-visible regions, our results can easily be used in simulations. We show a frame each from 2 different simulations in Figure 10. We use only our final dense strands in the simulations; our guide strands are not used. Instead, we subsample 10k strands to use as simulation "guides", which have the effect of all operators, including curl, baked in. We then bind these simulation "guides" to the dense strands, akin to interpolation. We use a custom simulator based on XPBD [Macklin et al. 2016]. Please see our video for full clips of both editing and simulation.

8 Conclusion and Future Work

We presented a pipeline for transforming unstructured hair strands obtained from artist designed grooms and hair reconstruction methods into procedural hair grooms. Our optimization strategies enable reconstructing artist-inspired guide strands and grooming operator parameters for various complex hairstyles. By using a procedural pipeline, our transformed grooms can be easily edited and also guarantee sensible inner hair structure.

Limitations and Future Work. While our method is able to transform complex hairstyles, we do not target exact strand-level accuracy and so may miss some local features, especially in real hair that is typically more irregular and may be more difficult to express procedurally. The grooming systems of 3D modeling tools such as Blender and Houdini are able to express these more challenging hairstyles by including more complex operators and more flexible graph connectivity of the operators. Extending our grooming pipeline to match these production systems would help generalize our method to even more hairstyles. This would likely necessitate further loss function and optimization design. In addition, our method assumes the set and connectivity of grooming operators and the number of guide strands is given. Automatically determining the topology of the operators and reasonable initial parameters would reduce the amount of user-provided input and improve robustness.

Acknowledgments

This work was funded by a contract from Meta, NSF IIS grants 2105806, 2238839, and an NSERC PGS D award. We also acknowledge support from the Ronald L. Graham Chair and the UC San

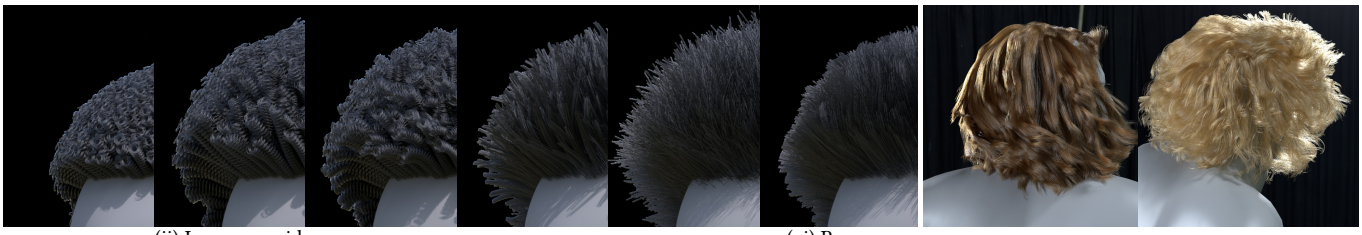


Fig. 10. Editing and simulation. To demonstrate our editing capabilities, we start from the groom in Figure 1, edit the guides (ii) and change one or two operator parameters at a time from (iii) to (vi). (b) Since our method recovers structurally sound strands, our grooms can be used in simulation. We show 2 simulations of the examples from Figure 8 (a) and (b). Full videos of both editing and simulation can be found in our video. Head model courtesy of ©Daniel Bystedt.

Diego Center for Visual Computing. We thank the anonymous reviewers for their feedback, Yash Belhe, Xuanda Yang, and Giljoo Nam for valuable discussions during the project, Xuanda Yang for editing the video, Gene Wei-Chin Lin for the simulations, Thrace Kelsick-Haseltine for sculpting the guide strands of the teaser image, Vanessa Sklyarova and the Neural Haircut authors for providing the monocular dataset, and Paris Blanco for the ponytail groom.

References

Ilya Baran and Jovan Popović. 2007. Automatic rigging and animation of 3D characters. *ACM Trans. Graph. (Proc. SIGGRAPH)* 26, 3, Article 72 (2007).

C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.* 22, 4 (dec 1996), 469–483. doi:10.1145/235815.235821

Gaurav Bhokare, Eisen Montalvo, Elie Diaz, and Cem Yuksel. 2024. Real-Time Hair Rendering with Hair Meshes. In *ACM SIGGRAPH 2024 Conference Papers* (Denver, CO, USA) *(SIGGRAPH '24)*. ACM, Article 61, 10 pages. doi:10.1145/3641519.3657521

Blender-Online-Community. 2018. Blender. http://www.blender.org

Dan Cascaval, Mira Shalah, Phillip Quinn, Rastislav Bodik, Maneesh Agrawala, and Adriana Schulz. 2022. Differentiable 3D CAD programs for bidirectional editing. In *Comput. Graph. Forum (Proc. Eurographics)*, Vol. 41. Wiley Online Library, 309–323.

Menglei Chai, Linjie Luo, Kalyan Sunkavalli, Nathan Carr, Sunil Hadap, and Kun Zhou. 2015. High-quality hair modeling from a single portrait photo. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 34, 6, Article 204 (2015), 10 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. (Proc. SIGGRAPH)* 35, 4, Article 116 (2016), 12 pages.

Menglei Chai, Lvdi Wang, Yanlin Weng, Xiaogang Jin, and Kun Zhou. 2013. Dynamic hair manipulation in images and videos. *ACM Trans. Graph. (Proc. SIGGRAPH)* 32, 4, Article 75 (2013), 8 pages.

Wesley Chang, Xuanda Yang, Yash Belhe, Ravi Ramamoorthi, and Tzu-Mao Li. 2024. Spatiotemporal Bilateral Gradient Filtering for Inverse Rendering. In *ACM SIGGRAPH Asia 2024 Conference Proceedings* (Tokyo, Japan) *(SIGGRAPH Asia '24)*. ACM, Article 70, 11 pages. doi:10.1145/3680528.3687606

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

Robert L. Cook. 1984. Shade Trees. *Comput. Graph. (Proc. SIGGRAPH)* 18, 3 (1984), 223–231.

Keenan Crane, Ulrich Pinkall, and Peter Schröder. 2013. Robust Fairing via Conformal Curvature Flow. *ACM Trans. Graph.* 32 (2013). Issue 4.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The Faiss library. (2024). arXiv:2401.08281 [cs.LG]

Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. 2018. InverseCSG: Automatic conversion of 3d models to CSG trees. 37, 6 (2018), 16.

Michael Fischer and Tobias Ritschel. 2023. Plateau-Reduced Differentiable Path Tracing. In *Computer Vision and Pattern Recognition*. 4285–4294.

Stéphane Grabli, François X Sillion, Stephen R Marschner, and Jerome E Lengyel. 2002. Image-based hair capture by inverse lighting. In *Graphics Interface*. 51–58.

Daniela Hasenbring and Henrik Karlsson. 2021. Hair Grooming with Imageworks' Fyber. In *ACM SIGGRAPH 2021 Talks* (Virtual Event, USA) *(SIGGRAPH '21)*. ACM, Article 37, 2 pages. doi:10.1145/3450623.3464668

Chengan He, Xin Sun, Zhixin Shu, Fujun Luan, Sören Pirk, Jorge Alejandro Amador Herrera, Dominik L Michels, Tuanfeng Y Wang, Meng Zhang, Holly Rushmeier, and Yi Zhou. 2024. Perm: A Parametric Representation for Multi-Style 3D Hair Modeling. *arXiv preprint arXiv:2407.19451* (2024).

Eric Heitz, Kenneth Vanhoey, Thomas Chambon, and Laurent Belcour. 2021. A Sliced Wasserstein Loss for Neural Texture Synthesis. In *Computer Vision and Pattern Recognition*.

J. Ben Hough, Manjunath Krishnapur, Yuval Peres, and Bálint Virág. 2006. Determinantal Processes and Independence. *Probability Surveys* 3, none (Jan. 2006). doi:10.1214/154957806000000078

Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. 2014. Robust hair capture using simulated examples. *ACM Trans. Graph. (Proc. SIGGRAPH)* 33, 4, Article 126 (2014), 10 pages.

Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. 2015. Single-view hair modeling using a hairstyle database. *ACM Trans. Graph. (Proc. SIGGRAPH)* 34, 4, Article 125 (2015).

Yiwei Hu, Julie Dorsey, and Holly Rushmeier. 2019. A novel framework for inverse procedural texture modeling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 38, 6 (2019), 1–14.

Wenzel Jakob, Jonathan T. Moon, and Steve Marschner. 2009. Capturing hair assemblies fiber by fiber. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 28, 5 (2009), 1–9.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and Improving the Image Quality of StyleGAN. In *Computer Vision and Pattern Recognition*.

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 2024. 3D Gaussian Splatting as Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems*. Spotlight Presentation.

Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

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.

Alex Kulesza. 2012. Determinantal Point Processes for Machine Learning. *Foundations and Trends® in Machine Learning* 5, 2–3 (2012), 123–286. doi:10.1561/2200000044

Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. 2017. Learning a model of facial shape and expression from 4D scans. *ACM Trans. Graph.* 36, 6 (Nov. 2017), 194:1–194:17. Two first authors contributed equally.

Shu Liang, Xiufeng Huang, Xianyu Meng, Kunyao Chen, Linda G. Shapiro, and Ira Kemelmacher-Shlizerman. 2018. Video to fully automatic 3D hair model. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6, Article 206 (2018), 14 pages.

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* (2024).

Linjie Luo, Hao Li, Sylvain Paris, Thibaut Weise, Mark Pauly, and Szymon Rusinkiewicz. 2012. Multi-view hair capture using orientation fields. In *Computer Vision and Pattern Recognition*. 1490–1497.

Linjie Luo, Hao Li, and Szymon Rusinkiewicz. 2013. Structure-aware hair capture. *ACM Trans. Graph. (Proc. SIGGRAPH)* 32, 4, Article 76 (2013), 12 pages.

Odile Macchi. 1975. The coincidence approach to stochastic point processes. *Advances in Applied Probability* 7, 1 (1975), 83–122.
Miles Macklin. 2022. Warp: A High-performance Python Framework for GPU Simulation and Graphics. https://github.com/nvidia/warp. NVIDIA GPU Technology Conference (GTC).
Miles Macklin, Matthias Müller, and Nuttapong Chentanez. 2016. XPBD: position-based simulation of compliant constrained dynamics. In *Proceedings of the 9th International Conference on Motion in Games* (Burlingame, California) *(MIG '16)*. ACM, 49–54. doi:10.1145/2994258.2994272
Ryota Maeda, Kenshi Takayama, and Takafumi Taketomi. 2023. Refinement of Hair Geometry by Strand Integration. *Comput. Graph. Forum (Proc. Pacific Graphics)* 42, 7 (2023).
Givi Meishvili, James Clemoes, Charlie Hewitt, Zafiirah Hosenie, Xian Xiao, Martin de La Gorce, Tibor Takacs, Tadas Baltrusaitis, Antonio Criminisi, Chyna McRae, Nina Jablonski, and Marta Wilczkowiak. 2024. Hairmony: Fairness-aware hairstyle classification. In *SIGGRAPH Asia 2024 Conference Papers* (Tokyo, Japan) *(SA '24)*. ACM, Article 116, 11 pages. doi:10.1145/3680528.3687582
Giljoo Nam, Chenglei Wu, Min H. Kim, and Yaser Sheikh. 2019. Strand-Accurate Multi-View Hair Capture. In *Computer Vision and Pattern Recognition*. 155–164.
Sofya Ogunseitan. 2022. Space Rangers with Cornrows: Methods for Modeling Braids and Curls in Pixar's Groom Pipeline. In *ACM SIGGRAPH 2022 Talks* (Vancouver, BC, Canada) *(SIGGRAPH '22)*. ACM, Article 49, 2 pages. doi:10.1145/3532836.3536277
Sylvain Paris, Hector M. Briceño, and François X. Sillion. 2004. Capture of hair geometry from multiple images. *ACM Trans. Graph.* 23, 3 (Aug. 2004), 712–719. doi:10.1145/1015706.1015784
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. (Proc. SIGGRAPH)* 27, 3 (2008).
Ken Perlin. 1985. An image synthesizer. *Comput. Graph. (Proc. SIGGRAPH)* 19, 3 (1985), 287–296.
F. Pitie, A.C. Kokaram, and R. Dahyot. 2005. N-dimensional probability density function transfer and its application to color transfer. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 2. 1434–1439 Vol. 2. doi:10.1109/ICCV.2005.166
Przemyslaw Prusinkiewicz. 1986. Graphical applications of L-systems. In *Proceedings of graphics interface*, Vol. 86. 247–253.
Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. 2012. Wasserstein Barycenter and Its Application to Texture Mixing. In *Scale Space and Variational Methods in Computer Vision*, Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein, and Michael M. Bronstein (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 435–446.
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. *European Conference on Computer Vision* (2022).
Shunsuke Saito, Liwen Hu, Chongyang Ma, Hikaru Ibayashi, Linjie Luo, and Hao Li. 2018. 3D hair synthesis using volumetric variational autoencoders. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6, Article 208 (2018), 12 pages.
Vanessa Sklyarova, Jenya Chelishev, Andreea Dogaru, Igor Medvedev, Victor Lempitsky, and Egor Zakharov. 2023. Neural Haircut: Prior-Guided Strand-Based Hair Reconstruction. In *International Conference on Computer Vision*.
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*.
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 *Computer Vision and Pattern Recognition*.
Jerry O Talton, Yu Lou, Steve Lesser, Jared Duke, Radomír Mech, and Vladlen Koltun. 2011. Metropolis procedural modeling. *ACM Trans. Graph.* 30, 2 (2011), 11.
Carlos A Vanegas, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes, and Paul Waddell. 2012. Inverse design of urban procedural models. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 31, 6 (2012), 11.
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)*. ACM, Article 56, 9 pages. doi:10.1145/1576246.1531362
Kelly Ward, Florence Bertails, Tae-Yong Kim, Stephen R Marschner, Marie-Paule Cani, and Ming C Lin. 2007. A survey on hair modeling: Styling, simulation, and rendering. *IEEE Trans. Vis. Comput. Graph.* 13, 2 (2007), 213–234.
Yasuhiko Watanabe and Yasuhito Suenaga. 1991. Drawing human hair using the Wisp model. *Vis. Comput.* 7, 2–3 (1991), 97–103.
Yichen Wei, Eyal Ofek, Long Quan, and Heung-Yeung Shum. 2005. Modeling hair from multiple views. *ACM Trans. Graph. (Proc. SIGGRAPH)* 24, 3 (2005), 816–820.
Yanlin Weng, Lvdi Wang, Xiao Li, Menglei Chai, and Kun Zhou. 2013. Hair Interpolation for Portrait Morphing. *Comput. Graph. Forum* 32, 7 (2013), 79–84. doi:10.1111/cgf.12214
Keyu Wu, Lingchen Yang, Zhiyi Kuang, Yao Feng, Xutao Han, Yuefan Shen, Hongbo Fu, Kun Zhou, and Youyi Zheng. 2024. MonoHair: High-Fidelity Hair Modeling from a Monocular Video. In *Computer Vision and Pattern Recognition*. 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 *Computer Vision and Pattern Recognition*. 1526–1535.
Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. 2020. RigNet: neural rigging for articulated characters. *ACM Trans. Graph. (Proc. SIGGRAPH)* 39, 4, Article 58 (2020), 14 pages.
Cem Yuksel, Scott Schaefer, and John Keyser. 2009. Hair meshes. *ACM Trans. Graph.* 28, 5, 1–7. doi:10.1145/1618452.1618512
Egor Zakharov, Vanessa Sklyarova, Michael J Black, Giljoo Nam, Justus Thies, and Otmar Hilliges. 2024. Human Hair Reconstruction with Strand-Aligned 3D Gaussians. In *European Conference on Computer Vision*.
Meng Zhang, Menglei Chai, Hongzhi Wu, Hao Yang, and Kun Zhou. 2017. A data-driven approach to four-view image-based hair modeling. *ACM Trans. Graph. (Proc. SIGGRAPH)* 36, 4, Article 156 (2017), 11 pages.
Meng Zhang, Pan Wu, Hongzhi Wu, Yanlin Weng, Youyi Zheng, and Kun Zhou. 2018. Modeling hair from an RGB-D camera. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)* 37, 6, Article 205 (2018), 10 pages.
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.
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 *Computer Vision and Pattern Recognition*. 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. (Proc. SIGGRAPH Asia)* 42, 6, Article 270 (Dec. 2023), 16 pages. doi:10.1145/3618309
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 Trans. Graph.* 43, 6, Article 254 (Nov. 2024), 15 pages. doi:10.1145/3687768
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 *European Conference on Computer Vision*. 235–251.

A Grooming Operator Formulations

Each operator takes in the current strands $\mathbf{x}_i$, operator guides $\mathbf{g}_{\text{op}_i}$, and outputs deformed strands $\tilde{\mathbf{x}}_i$, For brevity, we drop the strand index i from now on in this section and simply use the vertex/segment index j. Operators may draw random numbers u from either $\hat{\mathcal{N}}(0,1)$, the truncated normal distribution, bounded between $[-1,1]$, or $U(0,1)$, the uniform distribution. These u are different for each strand, with the exception of curl, in which each strand assigned to the same operator guide uses the same random number, which we denote as u^g. We list the parameters for each operator.

Below are some common operations across all operators:

$$e_j = x_{j+1} - x_j \qquad \text{strand segment}$$

$$\ell_j = \left\| e_j \right\| \qquad \text{segment length}$$

$$l = \sum_{j=1}^{m-1} \ell_j \qquad \text{strand length}$$

$$v_j = \sum_{j'=1}^{j} \frac{\ell_j}{l} \qquad \text{normalized distance along strand}$$

Bend. Parameters: Angle ϑ, bend start ζ. $R(\text{axis}, \text{angle})$ constructs an axis-angle rotation matrix.

$$
\begin{aligned}
u &\sim \hat{\mathcal{N}}(0,1) \\
e_{\text{op}_1} &= g_{\text{op}_2} - g_{\text{op}_1} && \text{guide root direction} \\
v_{\text{aux}} &= \begin{cases} (0,1,0), & \left|e^{\text{y}}_{\text{op}_1}\right| < 10^{-1} \\ (1,0,0), & \text{otherwise} \end{cases} \\
v &= e_{\text{op}_1} \times v_{\text{aux}} && \text{bend axis} \\
R_1 &= R(e_{\text{op}_1}, \pi u) \\
v' &= \frac{R_1 v}{||R_1 v||} && \text{rotate bend axis} \\
\zeta'_j &= \text{sigmoid}(10(v_j - \zeta)) && \text{start smooth indicator} \\
l' &= \sum_{j=1}^{m-1} \ell_j \zeta_j && \text{length after bend start} \\
v'_j &= \sum_{j'=1}^{j} \frac{\ell_j \zeta_j}{l'} && \text{norm. distance from bend start} \\
R_2 &= R(v', \vartheta v'_j) \\
e'_j &= R_2 e_j && \text{bend segment} \\
\tilde{x}_j &= \begin{cases} x_1, & j = 1 \\ x_1 + \sum_{j'=1}^{j} e'_j, & \text{otherwise} \end{cases} && \text{sum segments to get vertices}
\end{aligned}
$$

Curl. Parameters: Radius ρ, frequency ω, random frequency ω_{rand}, curl start ζ.

$$
\begin{aligned}
u^g_1 &\sim \hat{\mathcal{N}}(0,1) \\
u^g_2 &\sim U(0,1) \\
\omega' &= \max(3(\omega + \omega_{\text{rand}} u^g_1), 0) && \text{compute frequency} \\
\vartheta_j &= \left(\sum_{j'=1}^{j} \ell_{j'} \omega' \right) + 3.9 u^g_2 && \text{compute phase} \\
\rho'_j &= \rho(0.43 + 0.79 v_j) && \text{increase radius towards tip} \\
\beta^{\text{x}}_j &= \rho'_j \sin(2\pi\vartheta_j) \\
\beta^{\text{y}}_j &= \rho'_j \cos(2\pi\vartheta_j) && \text{curl shape} \\
\tau_j &= \begin{cases} \frac{g_{\text{op}_2} - g_{\text{op}_1}}{\left\| g_{\text{op}_2} - g_{\text{op}_1} \right\|}, & j = 1 \\ \frac{g_{\text{op}_m} - g_{\text{op}_{m-1}}}{\left\| g_{\text{op}_m} - g_{\text{op}_{m-1}} \right\|}, & j = m \\ \frac{g_{\text{op}_{j+1}} - g_{\text{op}_{j-1}}}{\left\| g_{\text{op}_{j+1}} - g_{\text{op}_{j-1}} \right\|}, & \text{otherwise} \end{cases} && \text{guide segment directions} \\
\hat{n}_j &= \begin{cases} (1,0,0), & \left|\tau^{\text{x}}_j\right| + \left|\tau^{\text{y}}_j\right| < 10^{-4} \\ \frac{(\tau^{\text{y}}_j, -\tau^{\text{x}}_j, 0)}{\left\| (\tau^{\text{y}}_j, -\tau^{\text{x}}_j, 0) \right\|}, & \text{otherwise} \end{cases} && \text{normal} \\
\hat{b}_j &= \tau_j \times \hat{n}_j && \text{binormal} \\
\zeta'_j &= \text{sigmoid}(10(v_j - \zeta)) && \text{start smooth indicator} \\
\tilde{x}_j &= \begin{cases} x_1, & j = 1 \\ x_j + \zeta'_j(\beta^{\text{x}}_j \hat{n}_j + \beta^{\text{y}}_j \hat{b}_j), & \text{otherwise} \end{cases} && \text{add curl shape}
\end{aligned}
$$

Scale. Parameters: Random scale η.

$$
\begin{aligned}
u &\sim \hat{\mathcal{N}}(0,1) \\
\eta' &= 1 + \eta u && \text{compute scale factor} \\
\tilde{x}_j &= x_1 + \eta'(x_j - x_1) && \text{scale strand}
\end{aligned}
$$

Clump. Parameters: Profile a.

$$
\begin{aligned}
a'_j &= 1 - \exp(-a v_j) && \text{0 at root, decays towards 1 at tip} \\
\tilde{x}_j &= (1 - a'_j) x_j + a'_j g_{\text{op}_j} && \text{blend strand and guide}
\end{aligned}
$$

Frizz. Parameters: Frequency ω, amplitude α. perlin implements 1D Perlin noise [Perlin 1985] and ξ^c is an arbitrary offset for c = x, y, z.

$$
\begin{aligned}
u &\sim U(0,1) \\
\chi_j &= \omega l_j v_j + 10u && \text{random offset} \\
\varepsilon^c_j &= \frac{2}{3}(\text{perlin}(\chi_j + \xi^c) + \frac{1}{3}\text{perlin}(2(\chi_j + \xi^c))) && \text{fractal noise} \\
\tilde{x}_j &= \begin{cases} x_1, & j = 1 \\ x_j + \frac{1}{2}\alpha\varepsilon_j, & \text{otherwise} \end{cases} && \text{add noise to strand}
\end{aligned}
$$

B Additional Ablations and Further Evaluation

Guide Smoothing. In Figure 11, we analyze the effect of smoothing the clustered centroids before using them as guide strands. This is controlled by the number of steps N_{smooth} used to solve the heat equation. With $N_{\text{smooth}} = 0$, we apply no smoothing, and as N_{smooth} increases, the strands approach the heat equation solution, which are completely straight strands. When no smoothing is applied to a curly groom, the curls are baked into the guides rather than the curl operator. Thus, the curl operator radius and frequency parameters are lower than desired. When we apply a high amount of smoothing, the guides become oversmoothed, and local details in the strands may be lost. Thus finding a good smoothing amount is important. We find $N_{\text{smooth}} = 80$ to be a good default for many grooms.

Instancing weights optimization. In Figure 12, we study the effect of optimizing the instancing weights. In (a), we construct a synthetic example where the target guide strands and operator parameters are known, and the instancing weights w are randomly sampled from the standard normal distribution. We then verify that optimizing the instancing weights converges to the same set of instanced strands. In (b) we show empirically that strands reconstructed from real examples are poorly represented using distance-based interpolation, leading to instabilities during guide optimization.

Removing flawed strands. In Figure 13, we compare the effect of removing flawed inner hair as discussed in Section 5.7 on optimization. Keeping flawed inner hair can bias the optimization towards undesirable parameters, such as short, straight strands.

Structural integrity comparison with MonoHair. In Figure 14, we compare the structure of the inner hair of our procedural groom with MonoHair [Wu et al. 2024]. MonoHair is the current state-of-the-art for reconstructing curly hair examples, but suffers from poor inner hair structure, as seen by the highlighted straight, short strands. On the other hand, our curl operator ensures that the strands are all

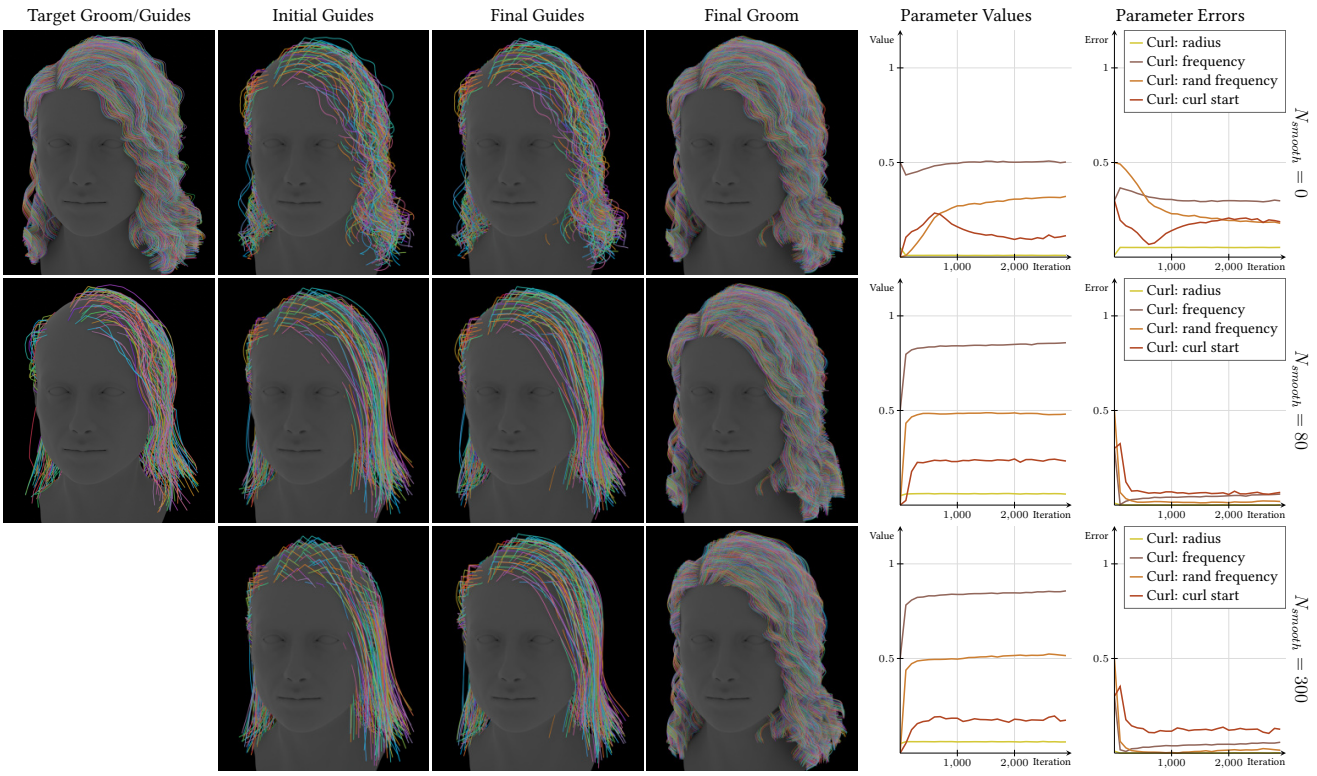


Fig. 11. Guide Initialization Smoothing. We analyze the effect of guide smoothing by running our optimization for 3 different amounts of smoothing ($N_{\text{smooth}} = 0, 80, 300$: no smoothing, normal, high). We visualize the initial guides after smoothing, the final guides and groom after the entire optimization pipeline, as well as the parameter values and Mean Absolute Error (MAE) over optimization. When we do not apply any smoothing, the curls are baked into the guides rather than the curl operator. With a high amount of smoothing, the initial guides are nearly straight. Our guide optimization fixes the oversmoothed guides to an extent, but misses some details. Setting $N_{\text{smooth}} = 80$ recovers the best guides and parameters. Groom adapted from Hair Styles ©Daniel Bystedt.

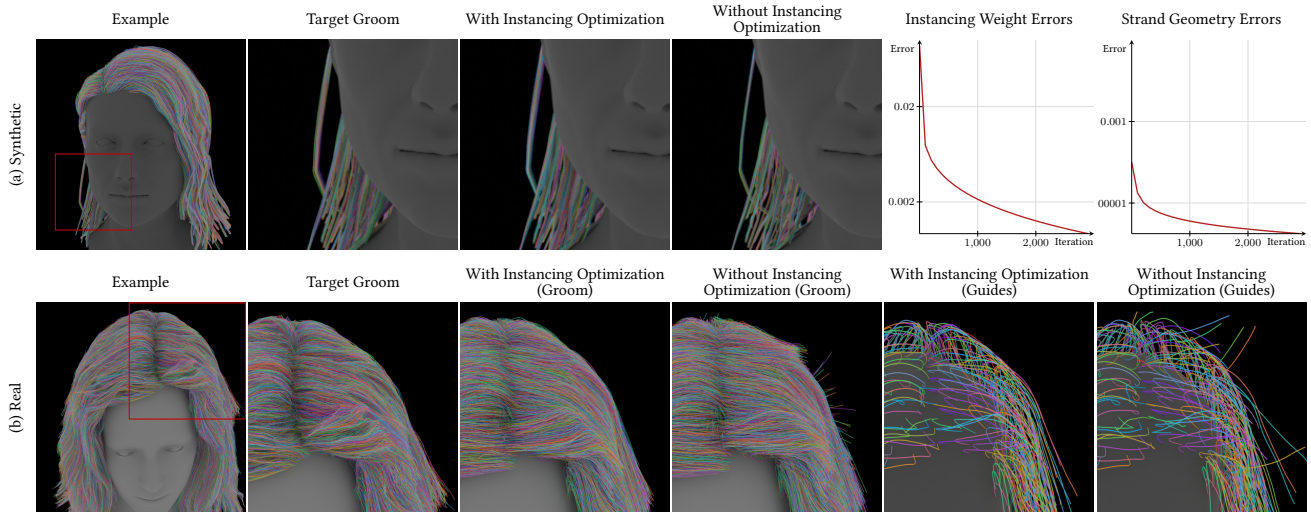


Fig. 12. Instancing Weights Optimization. We analyze the effect of optimizing the instancing weights. (a) We construct a synthetic example where all grooming pipeline parameters are known except the instancing weights. Using a simple distance-based interpolation can cause strand differences (note the bundle of hair on the left is too thin). By optimizing for the weights, we can achieve the exact reconstruction, which we validate by comparing Mean Squared Error (MSE) in instancing weights and MSE on strand vertices. (b) On real examples, optimizing the instancing weights can prevent issues during guide optimization when dense strands are poorly interpolated using a distance-based interpolation. Synthetic groom adapted from Hair Styles ©Daniel Bystedt.

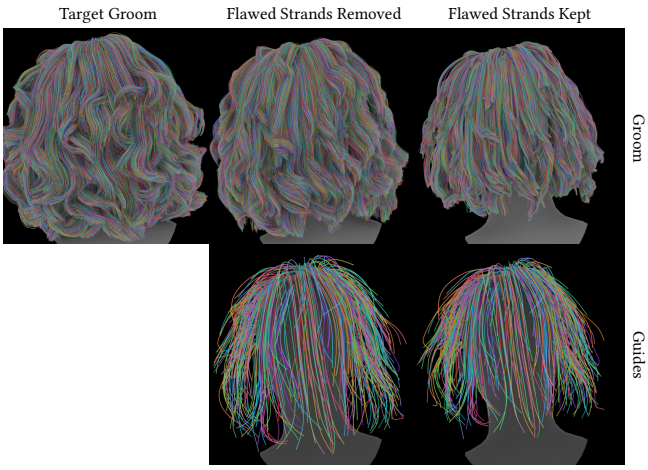


Fig. 13. Removing flawed strands. Neural hair reconstruction methods that infer inner hair using data-driven priors can have flawed strands. For example, in this groom, the inner hair in the back is short and straight, rather than curly as expected, as seen in Figure 14. Optimizing using these flawed strands causes the guide optimization to fit to the short strands, reducing the hair length of the back. Removing the flawed strands helps prevent this.

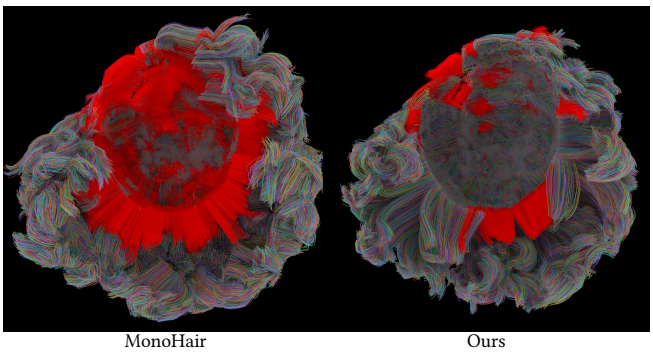


Fig. 14. Structural Integrity Comparison with MonoHair. We visualize the inner hair of a curly groom reconstructed using MonoHair [Wu et al. 2024] compared to ours. Short, straight strands, which are undesired, are highlighted in red. Our procedural operators ensure that most strands in a curly groom are curly, whereas MonoHair generates unnatural straight hair.

curly to some extent, although we lose some reconstruction quality due to the flawed target, as shown in Figure 13.

Operator losses. In Figure 15 and Figure 16, we compare our probabilistic losses with a simple vertex-wise L2 loss. We construct synthetic examples with varying amounts of known parameters: known guides or random numbers. The L2 loss can optimize some operator parameters in simple cases, but quickly fails with more unknowns.

Gradient noise. In Figure 17, we compare optimizing the operator random numbers with and without adding noise to the gradients. When operators are nonlinear, the noise can help escape local

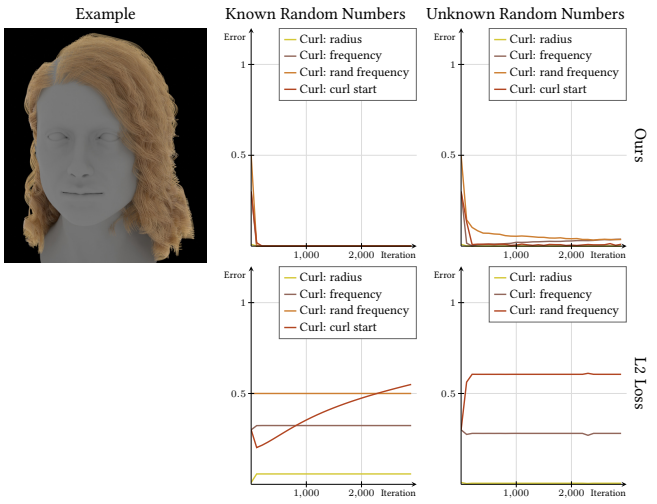


Fig. 15. Strand shape loss. We compare our loss functions in optimizing the curl operator, a strand shape operator, in a synthetic example. Here we assume the guide strands are known, and we compare MAE on the parameter values. When the random numbers are also known, there is a set of parameters that would exactly match the target strands. Even in this case, as curl is a nonlinear function in its parameters, a vertex-wise L2 loss cannot optimize to the correct parameters, whereas ours converges to the exact parameters. When the random numbers are not known, an exact match cannot be expected. Ours converges to within a reasonable tolerance, while the L2 loss still diverges. Groom adapted from Hair Styles ©Daniel Bystedt.

minima that prevent matching local strand orientations like seen in Figure 9.

Number of guides. In Figure 18, we compare optimization at increasing numbers of guide strands n_g. The number of guide strands trades off strand accuracy with editability. We typically choose $n_g = 500$ or $n_g = 1000$, which are similar numbers used in artist-designed procedural grooms.

Number of operator guides. In Figure 19, we compare optimization at increasing numbers of operator guide strands. The number of operator guide strands in a curly groom changes the number and size of the curls, as all strands assigned to an operator guide curl around that operator guide. The number of operator guides should thus be chosen based on the style of the target groom.

Guide strand generation comparison with Perm. In Figure 20, we compare our guide strand initialization and optimization with the guide strand generation from Perm [He et al. 2024] as shown in Figure 20. Their method uses a learned representation based on StyleGAN2 [Karras et al. 2020]. Thus, while their method generates reasonable smooth strands following the overall hair shape, they are unable fit accurate guide strands to specific grooms, especially challenging ones, like the curly and frizzy groom in the first row. On the other hand, our clustering and optimization is a more direct approach that can recover accurate guide strands.

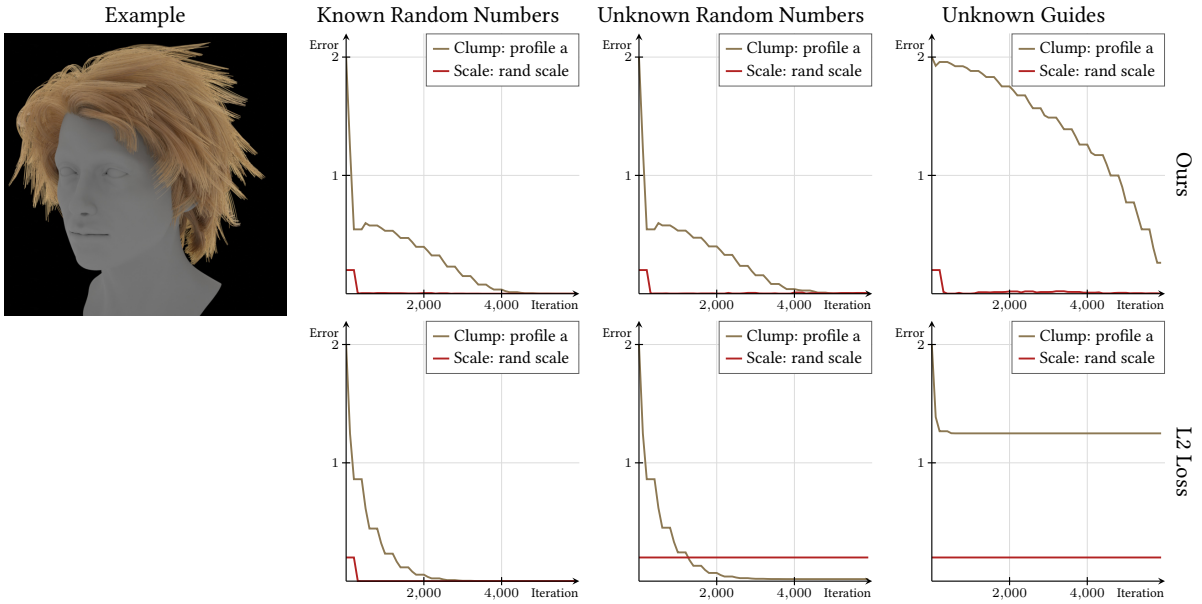


Fig. 16. Strand correlation loss. We compare our loss functions in optimizing the clump and scale operators, which are strand correlation operators, in a synthetic example. We compare MAE on the parameter values. First, we assume the guide strands are known. When the random numbers are also known, there is a set of parameters that would exactly match the target strands. As both operators are monotonic functions in their parameters, both our loss and the vertex-wise L2 loss can optimize to the correct parameters. When the random numbers are not known, an exact match cannot be expected. Both our loss and the L2 loss can optimize the clump operator, which has no random parameters, but only our loss can optimize the scale operator. When neither the guides nor the random numbers are known, our loss can optimize to parameters within a reasonable tolerance, while the L2 loss can optimize neither. Groom adapted from Hair Styles ©Daniel Bystedt.

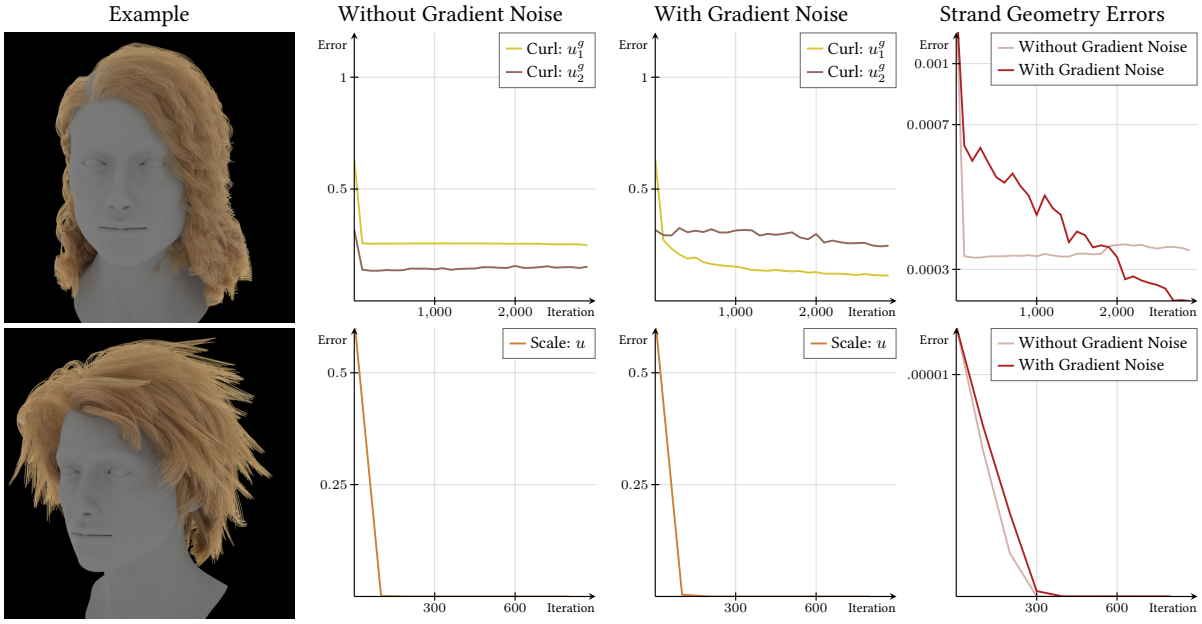


Fig. 17. Gradient Noise. We compare optimizing the operator random numbers with and without adding noise to the gradients. Here, we assume both the guide strands, as well as the operator parameters are known, only the operator random numbers are not known. We measure both the parameter MAE, as well as MSE on the strand vertex positions (shown in log scale). When optimizing for scale, which is a linear function in the random numbers, both methods can optimize to the correct random numbers. When optimizing curl, which nonlinear in its random numbers, not adding noise quickly gets stuck in a local minima. Adding noise can help escape the local minima and continue reducing the parameter and geometry errors. Groom adapted from Hair Styles ©Daniel Bystedt.

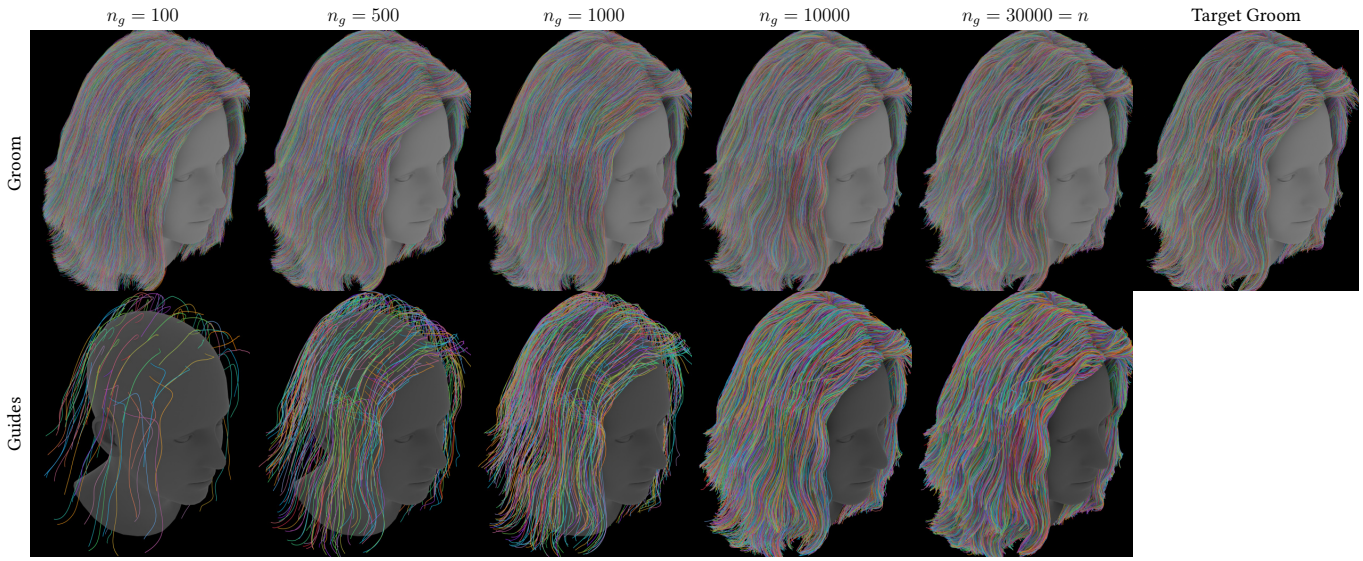


Fig. 18. Number of guides. We compare the effect of the number of guide strands on the optimization, from $n_g = 100$, to $n_g = 30000$, which is the same number of guides as the number of dense strands n. As we increase the number of guides, we can match more detail in the target groom. However, increasing the number of guides reduces the ease of editing the procedural groom.

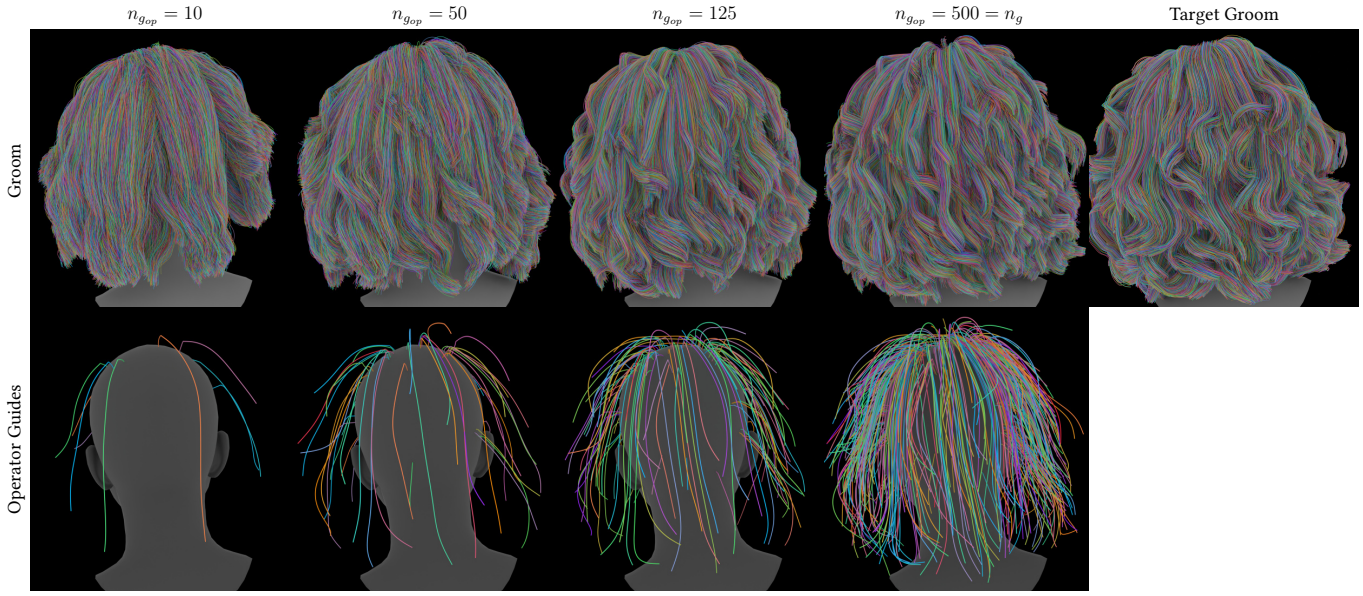


Fig. 19. Number of operator guides. We compare the effect of the number of operator guides strands on the optimization, from $n_{g_{\text{op}}} = 10$ to $n_{g_{\text{op}}} = 500$, which is the exact set of n_g guides used. Increasing the number of operator guides changes the number and width of the curls in the groom.

(a) Target Groom (b) Target Guides (c) Our Guides (d) Perm Guides

Fig. 20. Comparison with Perm. Given target grooms (a), our guide clustering and optimization (c) result in more accurate guide strands (b) compared to the learned guide strand generation of Perm [He et al. 2024] (d). Grooms adapted from Hair Styles ©Daniel Bystedt.