DOI: 10.1111/cgf.14970
Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 7

Refinement of Hair Geometry by Strand Integration

Ryota Maeda[1†], Kenshi Takayama[2], Takafumi Taketomi[2]

[1]University of Hyogo, Japan [2]CyberAgent, Japan

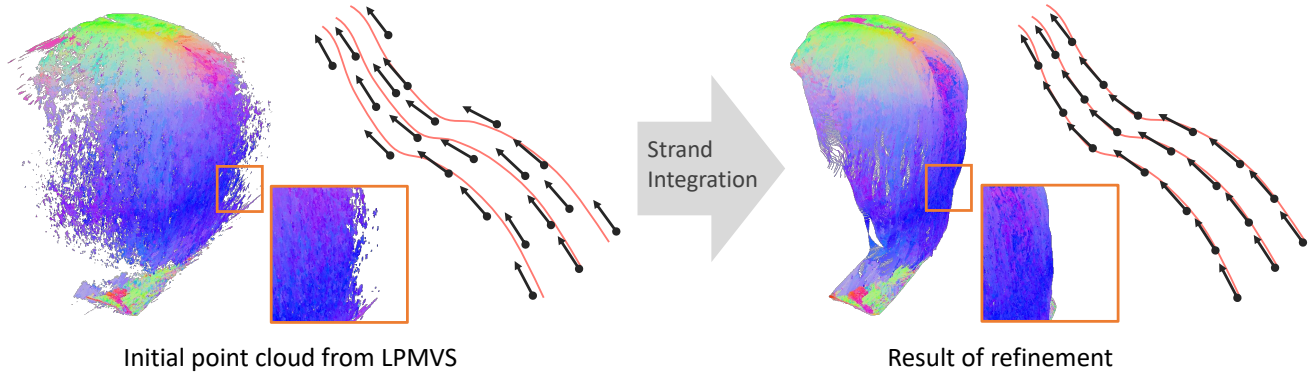


Figure 1: ***Example result of shape refinement.*** *Our strand integration technique refines the inaccurate hair strand by integrating the gradient along the strand. Left: Initial hair geometry from LPMVS [NWKS19]. Each point has 3D direction information illustrated as black arrows (while the red lines are for the purpose of illustration only and are not generated). Right: Refined hair geometry using our method. Our method improves the accuracy of the hair geometry without any additional data.*

Abstract

Reconstructing 3D hair is challenging due to its complex micro-scale geometry, and is of essential importance for the efficient creation of high-fidelity virtual humans. Existing hair capture methods based on multi-view stereo tend to generate results that are noisy and inaccurate. In this study, we propose a refinement method for hair geometry by incorporating the gradient of strands into the computation of their position. We formulate a gradient integration strategy for hair strands. We evaluate the performance of our method using a synthetic multi-view dataset containing four hairstyles, and show that our refinement produces more accurate hair geometry. Furthermore, we tested our method with a real image input. Our method produces a plausible result. Our source code is publicly available at `https://github.com/elerac/strand_integration`.

CCS Concepts

• ***Computing methodologies*** → ***Reconstruction; Shape modeling;***

1. Introduction

Hair is an important aspect of our appearance, and it has a significant impact on how we look and feel. Modeling realistic and natural-looking hair is essential to creating virtual humans for many applications such as VR, AR, and online commerce. However, hair modeling is a laborious and time-consuming task that requires technical expertise and artistic skill, especially when trying to achieve a specific hairstyle.

Automatic hair generation using deep neural networks has

† Work done during an internship at CyberAgent AI Lab.

widely been investigated in graphics and vision [RSW*22, KCF*22]. 3D hair reconstruction from a single portrait can in particular drastically improve a hair modeling workflow [WYY*22, YZM*23]. These approaches are, however, difficult to generalize due to diverse variations in hairstyles.

On the other hand, directly capturing actual hair strands using a multi-camera system has also been explored. Although special equipment is needed, various hairstyles can be reconstructed. To reconstruct hair strands from multiple-view images, the multi-view stereo (MVS) algorithm is specialized by assuming 3D local lines instead of using 3D local planes [PCK*08, HMLL14, NWKS19, SNA*21]. In these methods, reconstructed points are associated with direction information. In particular, Nam et al. proposed an algorithm for accurately reconstructing hair strands named line-based PatchMatch MVS (LPMVS) [NWKS19]. While their method achieves remarkable hair reconstruction results, the intermediate raw output of reconstructed strands still exhibits significant amount of noise. In the original paper, they address this issue by eliminating the inconsistent 3D lines and selecting only a few reliable ones. Consequently, this approach reduces the overall density of the 3D lines, resulting in sparse reconstruction results.

In the 3D vision research field, normal integration is a classical technique for surface reconstruction that determines shape from a set of normal vectors [HB86]. This technique is beneficial for reconstructing the detailed shape, whereas the MVS technique is suited to the overall shape. The basic idea is that the gradient equals the differentiation of shape. As a result, we can reconstruct the object shape by integrating normals. Although numerous normal integration approaches have been proposed [QDA18], all of them still need to be modified to be applied to hair strands because existing approaches assume smooth surfaces.

In this study, we propose a refinement method for hair strands by integrating gradients with a line assumption, as illustrated in Fig. 1. We start with a noisy 3D line map (depth and direction) obtained through the LPMVS technique and refine it for each view. The refined lines are merged to reconstruct the final hair geometry. Unlike the previous approach, our refinement process can restore the noisy 3D lines instead of eliminating them. It can preserve 3D lines and reconstruct dense and less noisy hair shapes. To accomplish this, inspired by the normal integration method, we leverage the gradient for correcting the noisy lines. Compared to the traditional normal integration, our method extracts the gradient from the direction of the 3D line map rather than the normal vector of a plane. Furthermore, in our method, the directions of integration are defined by the 2D hair directions estimated on the input images. Additionally, as an option to further improve the quality, our method can integrate normal vectors estimated by the photometric stereo technique [Woo80]. Because our method adapts the concept of normal integration to the task of strand reconstruction, we call our method **Strand Integration**.

2. Related Work

There are several approaches for reconstructing the 3D shape of hair from a captured image input. 3D hair reconstruction from a single portrait is actively studied because it can drastically reduce the modeling cost. As 3D reconstruction from a single portrait is an ill-posed problem, these methods estimate the 3D hair shape using prior knowledge. For example, information from the user's sketch is used to determine the flow of the hair [CWW*12,CLS*15, HMLL15]. Neural networks are also used to estimate the 3D shape of the hair [CSW*16, ZHX*18, SHM*18, WYY*22, YZM*23]. This type of data-driven approach can predict the entire 3D shape of the hair. Although the single-view and data-driven approach is beneficial for easier deployment, it struggles in reconstructing the invisible area, and tends to fail to reconstruct irregular hairstyles that are not well represented in the training dataset.

The multi-view camera system can capture accurate hair shapes without relying on prior knowledge. Pioneering research by Paris et al. introduced a method to reconstruct a hair shape from multi-view images [PBS04]. Their approach involves analyzing the orientation of hair strands to recover the 3D orientation or 3D lines. Building upon their research, subsequent work has steadily improved the quality of hair reconstruction [WOQS05, PCK*08, LLP*12, LZZR13,LLR13,HMLL14]. In addition, some research uses a special capture approach or setup for hair reconstruction such as focal stack photography [JMM09], thermal camera [HZW12], and RGB-D camera [ZWW*18]. These unique setups can acquire additional physical cues to help reconstruct hair. Moreover, some studies have employed a learning-based approach with multi-view images [ZCW*17, KCF*22].

Among these multi-view hair capture approaches, Nam et al. proposed a successful hair reconstruction method called LPMVS [NWKS19]. This method expands upon a traditional MVS for hair strands by considering hair strands as 3D lines rather than 3D planes. Our method uses LPMVS to generate the initial input of rough hair shape and subsequently improves its quality through refinement. Some of their subsequent works use LPMVS. For example, Sun et al. combined photometric images as the feature of matching strands [SNA*21]. In our approach, we also use photometric images, but we employ them to extract normal vectors and integrate them into refinement. Recent advanced neural approaches represent hair features in a latent space, enabling the rendering of realistic hair images [RSW*22, WNS*22]. These methods incorporate the output of LPMVS as a geometric guide during optimization, and can benefit from the improved accuracy of prior hair shapes offered by our method. In this sense, our work is complementary to these neural methods and has a potential to contribute to the overall quality improvement of hair reconstruction.

3. Strand Integration

Our strand integration algorithm is inspired by normal integration. Normal integration, also known as shape from gradients, is a well-known method in computer vision that reconstructs a 3D surface from its normal vectors. This method was originally introduced by Horn and Brooks in 1986 [HB86]. Since then, numerous improved methods have been proposed over several decades [QDA18]. Despite the advancements, it remains a challenging task. Our work adapts this idea to hair strands by leveraging the 3D direction as a gradient.

In this section, we describe the details of our proposed method;

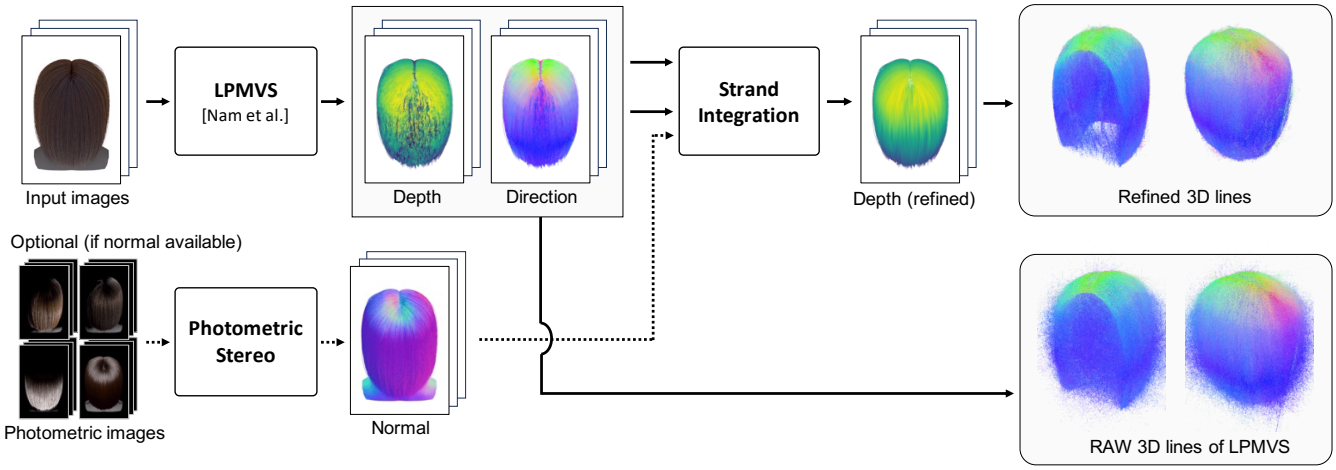


Figure 2: ***Overall pipeline of our hair reconstruction.*** *The input images are captured under a multi-view camera condition. As a preprocessing, we reconstruct 3D lines (depth and direction) using LPMVS [NWKS19]. These 3D lines become the input to our method, Strand Integration, which refines the depth map for each view. While our method can work without any extra input, we can also optionally accept a normal map as an additional input. We can obtain the hair geometry as a set of 3D lines by accumulating.*

see Fig. 2 for an overview. The input of our method is a 3D line map which is obtained as the output of LPMVS [NWKS19]. This 3D line map comprises the depth map (3D points) and their 3D directions. The basic concept of our strand integration is to harness the position and direction information for improving geometrical coherence. If we move along a 3D line following its direction, we will likely encounter another 3D line that represents another piece of the same strand. By following successive 3D lines, we can determine the shape of an entire strand of hair. We formulate this relationship similar to normal integration but for 3D lines.

We first introduce the notation of the geometrical parameters of the hair strand (Fig. 3). When a calibrated camera captures the hair, its strand is projected to a point on the image plane $\mathbf{u} = [u,v]^\top \in \mathbb{R}^2$. We define the area where at least one strand is projected as Ω, and let $|\Omega|$ denote its number of pixels. The prior hair strands obtained by LPMVS are represented as a pair of depth map $z_{\text{prior}}(\mathbf{u}) \in \mathbb{R}$ and the 3D direction map $\mathbf{d}_{\text{prior}}(\mathbf{u}) = [d_{x,\text{prior}}, d_{y,\text{prior}}, d_{z,\text{prior}}](\mathbf{u})^\top \in \mathcal{S}^2 \subset \mathbb{R}^3$ in the camera coordinate system where $\mathbf{u} \in \Omega$. We denote the angle of the strand on the image plane as $\theta(\mathbf{u}) = -\arctan(d_{y,\text{prior}}/d_{x,\text{prior}}) \in [-90^\circ, 90^\circ)$.

3.1. Notation of Strand

Resolving Directional Ambiguity. Note that there exists an ambiguity regarding the directionality of a 3D line: the 3D direction vector $\mathbf{d}_{\text{prior}}$ and its flipped version $-\mathbf{d}_{\text{prior}}$ can represent the same 3D line. We eliminate this ambiguity by flipping the direction vector as needed such that the X component of the direction $d_{x,\text{prior}}$ is always positive.

3.2. Deriving 3D Direction from Depth

In the following, we formulate the derivation of the 3D strand direction $\mathbf{d}$ from the to-be-optimized depth z. First, we take the directional derivative of the depth as

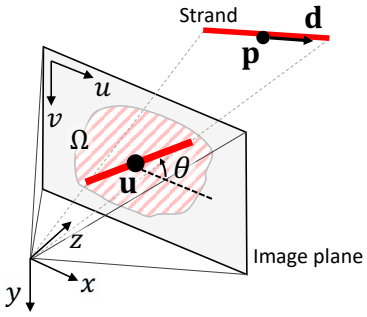


Figure 3: ***Notation of strand.*** *The 3D line as a strand is projected to the image plane.*

$$\nabla_\theta z = [\cos\theta, -\sin\theta]^\top \cdot \nabla z \tag{1}$$

where the gradient of the depth is computed by

$$\nabla z = \frac{1}{\delta(u,v)} \begin{bmatrix} z(u+1,v) - z(u,v) \\ z(u,v+1) - z(u,v) \end{bmatrix}. \tag{2}$$

Here, δ represents the length of one pixel seen at depth $z(u,v)$ obtained as

$$\delta(u,v) = \frac{z(u,v)}{f} \tag{3}$$

where f is the camera's focal length.

With Equation (1), we can derive the 3D strand direction as

$$\mathbf{d} = \langle [\cos\theta, -\sin\theta, \nabla_\theta z]^\top \rangle \tag{4}$$

where $\langle\cdot\rangle$ denotes the normalization operation. We are particularly

interested in its Z component

$$d_z = \frac{\nabla_\theta z}{\sqrt{1+(\nabla_\theta z)^2}} \tag{5}$$

which is used in our loss function introduced in the next subsection.

3.3. Loss Function

Our strand integration finds the depth map which minimizes a loss function consisting of the following terms.

Direction Loss. The direction loss $\mathcal{L}_\mathrm{d}$ evaluates the difference between the prior direction $\mathbf{d}_\mathrm{prior}$ and the direction $\mathbf{d}$ computed from the unknown depth using Equation (4). Because we are optimizing the depth, only the Z component needs to be compared:

$$\mathcal{L}_\mathrm{d}(z) = \frac{1}{|\Omega|}\sum_\Omega \left(d_z - d_{z,\mathrm{prior}}\right)^2. \tag{6}$$

Depth Loss. The depth loss $\mathcal{L}_\mathrm{z}$ is a regularization term which compares the unknown depth z to the prior depth z_prior:

$$\mathcal{L}_\mathrm{z}(z) = \frac{1}{|\Omega|}\sum_\Omega c(z - z_\mathrm{prior})^2. \tag{7}$$

Here, we introduce a per-pixel weight c; the higher the weight, the more consistent the 3D line with respect to neighboring views. We defer the details on how to compute this weight to Subsection 3.5. With this weighting scheme, we can effectively incorporate the multiview constraint into the reconstruction process.

Normal Loss. This loss is optional but has the potential to improve reconstruction quality. If we obtain the normal vector $\mathbf{n}_\mathrm{prior}$ of hair as prior knowledge (e.g., by using photometric stereo or polarization imaging), we can incorporate it as an additional constraint. A strand's normal vector should be perpendicular to its 3D direction; i.e., the dot product of these vectors should be small. Accordingly, we define the normal loss $\mathcal{L}_\mathrm{n}$ as

$$\mathcal{L}_\mathrm{n}(z) = \frac{1}{|\Omega|}\sum_\Omega (\mathbf{n}_\mathrm{prior}\cdot\mathbf{d})^2. \tag{8}$$

Total Loss. We combine these loss functions into our total loss function $\mathcal{L}_\mathrm{total}$ defined as

$$\mathcal{L}_\mathrm{total}(z) = \mathcal{L}_\mathrm{z}(z) + \lambda_\mathrm{d}\mathcal{L}_\mathrm{d}(z) + \lambda_\mathrm{n}\mathcal{L}_\mathrm{n}(z) \tag{9}$$

where λ_d and λ_n are the weight parameters controlling the importance of the individual terms. We optimize the depth z such that the total loss is minimized by using a gradient descent method.

3.4. Robust Discretization

For making our algorithm more robust to noisy 3D lines, we find it useful to use both the forward and backward differencing schemes when computing the derivative. We denote the forward differencing operator introduced in Equation (2) by $\boldsymbol{\nabla}^+ z$, and likewise introduce the backward differencing operator as

$$\boldsymbol{\nabla}^- z = \frac{1}{\delta(u,v)}\begin{bmatrix} z(u,v) - z(u-1,v) \\ z(u,v) - z(u,v-1)\end{bmatrix}. \tag{10}$$

Using these two operators, we obtain two versions of the estimated strand direction, $\mathbf{d}^+$ and $\mathbf{d}^-$ following the same Equation (4). We then modify the direction loss and the normal loss as

$$\mathcal{L}_\mathrm{d}(z) = \frac{1}{|\Omega|}\sum_\Omega \frac{\left(d_z^+ - d_{z,\mathrm{prior}}\right)^2 + \left(d_z^- - d_{z,\mathrm{prior}}\right)^2}{2} \tag{11}$$

$$\mathcal{L}_\mathrm{n}(z) = \frac{1}{|\Omega|}\sum_\Omega \frac{(\mathbf{n}_\mathrm{prior}\cdot\mathbf{d}^+)^2 + (\mathbf{n}_\mathrm{prior}\cdot\mathbf{d}^-)^2}{2}. \tag{12}$$

3.5. 3D Line Consistency Map

In this subsection, we give details on how to compute the consistency weight map c used in the depth loss (Equation (7)). Our method is inspired by the consistency checking method used in the filtering step of LPMVS [NWKS19] (Subsection 4.2): we project the 3D line on the reference view to its neighboring views, and check for the consistency in terms of both position and direction.

Specifically, for each pixel on the reference view, we turn the prior depth z_prior into a 3D point $\mathbf{p}_\mathrm{prior}$ in the world coordinate system using the camera information, and then project it to each of the neighboring views. On the neighboring view i, the projected location has its prior depth $z_{i,\mathrm{prior}}$ which can then be turned into a world-coordinate 3D point $\mathbf{p}_{i,\mathrm{prior}}$. We compute the squared distance between these two points, and take the weighted average of this value over the set of neighboring views:

$$r = \frac{\sum_{i=1}^{N_\mathrm{nei}} w_i ||\mathbf{p}_\mathrm{prior} - \mathbf{p}_{i,\mathrm{prior}}||^2}{\sum_{i=1}^{N_\mathrm{nei}} w_i}. \tag{13}$$

Here, the weight w_i is derived by considering the consistency of the directions of the corresponding 3D lines. Specifically, let $\mathbf{d}_\mathrm{prior}$ and $\mathbf{d}_{i,\mathrm{prior}}$ denote the 3D direction vectors associated with the 3D lines on the reference view and the neighbor view i, respectively, expressed in the world coordinate system. We define the weight as

$$w_i = 90^\circ - \angle(\mathbf{d}_\mathrm{prior}, \mathbf{d}_{i,\mathrm{prior}}), \tag{14}$$

where $\angle(\cdot,\cdot)$ measures the angle between given two 3D lines. This definition allows us to ignore outlier or occluded lines in the neighboring views when computing the weighted average of the squared distances.

We plug this weighted average of the squared distances into the Gaussian kernel to derive the consistency weight:

$$c = \exp\left(-\frac{r}{2\sigma^2}\right), \tag{15}$$

where σ (set to 25mm) is the radius of the Gaussian kernel which controls the tolerance of depth inconsistency.

Note that we use this consistency map for the depth loss only, because we found that the direction map estimated by LPMVS is much more reliable than the depth map for most pixels.

4. Experiment

We evaluated our method using four synthetic hair geometry data released by Cem Yuksel [YSK09] (referred to as *Straight*, *Curly*, *Wavy* and *WavyThin*) as well as an actual multi-view capture of a real human subject (referred to as *Real*), as reported below.

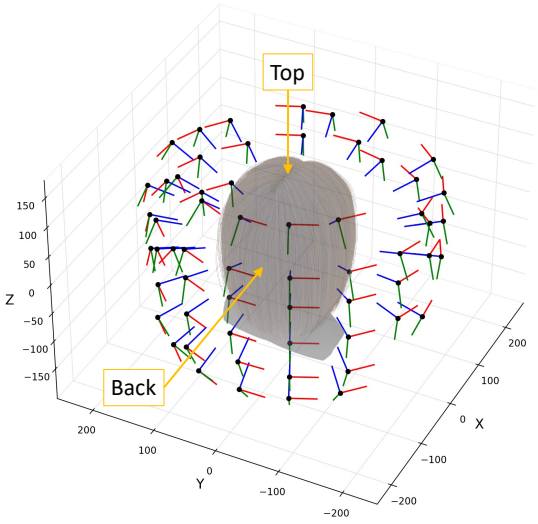


Figure 4: ***Camera poses of our synthetic data.*** *We position the cameras denser on the back side of the head.*

4.1. Implementation Details

Since there exists no publicly-available implementation of LPMVS [NWKS19], we created a CPU-based reimplementation of it in C++ with parallelization via OpenMP. We implemented our strand integration algorithm using PyTorch, and we used Adam [KB15] to minimize the loss function in Equation (9) and used the exponential learning rate decay scheme such that the learning rate is scaled down by 0.01 in the last iteration. The initial learning rate was set to 1.0 for all the synthetic cases, while it was set to 0.01 for the *Real* case. We adjusted the total number of iterations depending on the geometric complexity: 30k for the *Straight* case, 40k for the *Wavy* and *WavyThin* cases, and 50k for the *Curly* and *Real* cases. When running on a MacBook Pro with the Apple M1 Max chip and using PyTorch's GPU support, the typical per-view running time of our strand integration algorithm for the *Straight* case was about 20m.

4.2. Evaluation with Synthetic Data

Data Preparation. We used pbrt-v4 to render the hair geometry data into images of resolution 2730×4096. As shown in Fig. 4, we placed 60 cameras around the head model with more (less) cameras placed on the back (front) side, simulating an actual multi-view capture setup. We used 30 directional light sources evenly distributed in all directions, and rendered images in the one-light-at-a-time (OLAT) fashion. We fed this data to the classical photometric stereo technique [Woo80], assuming the Lambertian reflectance, to obtain normal maps needed by our normal loss (Equation (8)). We generated fully lit images by summing all the OLAT images, and used them as input to LPMVS [NWKS19]. Using the "gbuffer" functionality of pbrt-v4, we generated ground-truth depth maps as well as the mask images representing the hair region Ω.

Table 1: ***Aggregated error across all the views.*** *Our method consistently achieves lower errors than LPMVS.*

	Straight		Curly		Wavy		Wavythin	
	MAE ↓	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓	RMSE ↓	MAE ↓	RMSE ↓
LPMVS	34.58	54.20	37.60	59.55	31.62	53.06	32.64	52.61
Ours	**12.79**	**18.56**	**17.08**	**26.09**	**11.43**	**18.68**	**15.11**	**23.50**

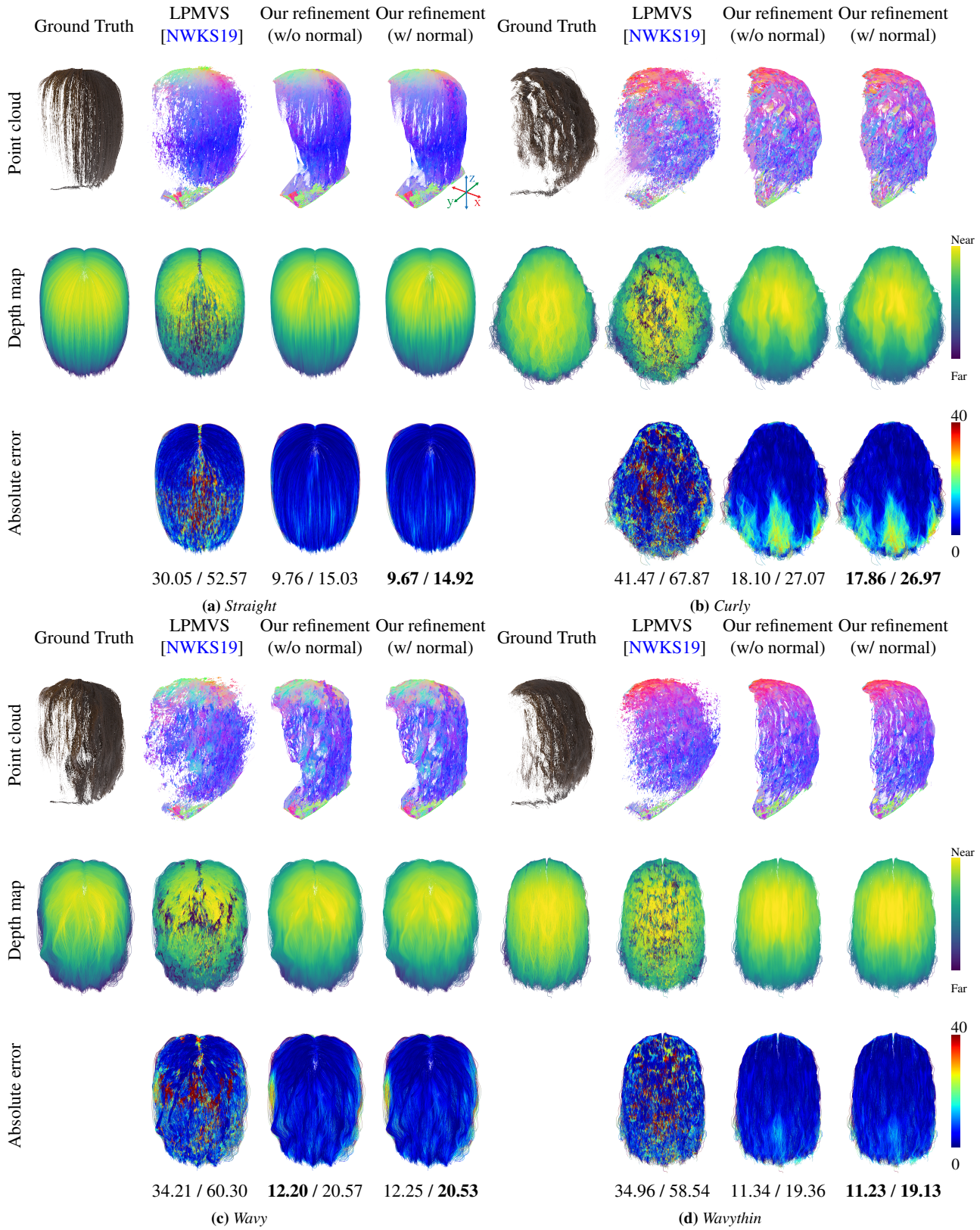
Error Analysis on Depth Maps. Using the ground-truth depth maps, we analyzed the errors of the depth maps generated by LPMVS [NWKS19], our method without normal loss ($\lambda_d = 72, \lambda_n = 0$), and our method with normal loss ($\lambda_d = 72, \lambda_n = 36$). Note that we determined these weight values relative to the depth loss which is assumed to be in the unit of mm^2. We report the mean-absolute-error (MAE) and root-mean-squared-error (RMSE) of the depth maps (both in mm) in Fig. 5 for a single view from the back, and in Table 1 for the aggregated values over all the views. We can see that our method consistently achieved lower error compared to LPMVS for all the cases, and the use of normal loss proved effective for all but the *Wavy* case where there was no significant difference. Fig. 6 shows how the error varies depending on the direction loss weight λ_d (the normal loss weight λ_n was fixed to 0), indicating that our algorithm can consistently achieve higher accuracy than LPMVS without relying on hyperparameter tuning.

Merged Hair Geometry. After obtaining depth maps of all the views, we merge the point clouds of all the views into one point cloud. Fig. 7 shows two versions of merged point clouds: those that use the filtering scheme of LPMVS [NWKS19] (Subsection 4.2) and those that do not. For the filtering, we set the positional threshold τ_p and the directional threshold τ_d to 7.3mm and 10°, respectively, and kept points that satisfy the multi-view consistency criteria in at least 2 out of 6 neighboring views. Note that our method's merged point clouds come from the depth maps generated without using the normal loss. We can clearly see that the unfiltered results of LPMVS contain a significant amount of outlier points, while the filtered results of our method retain significantly more points thanks to the more accurate depth maps obtained by our strand integration.

An exception to the above assessment is observed with the Curly hairstyle where the number of remaining points after filtering is higher for the result of LPMVS than ours (37M vs. 34M). A plausible explanation for this is that the number of remaining points after filtering is likely not the most useful measure of success, especially when it comes to complex and heavily occluded hairstyles such as Curly. To show this, we ran the same filtering on the *ground truth* point clouds and confirmed that for the Curly hairstyle, 49.6% of points were removed, whereas only 9.8% of points were removed for the Straight hairstyle. This significant reduction of points is because a ground truth 3D point seen from one viewpoint may often be invisible from its neighboring viewpoints due to occlusion. As such, we deem the reported numbers for the Curly hairstyle to be less meaningful. This insight also suggests a better way of filtering the point cloud, a subject for future work.

4.3. Evaluation with Real Data

Capture Setup. We photographed a real female person with long hairs in a multi-view capture setup with 60 cameras arranged in a



(a) *Straight* **(b)** *Curly* **(c)** *Wavy* **(d)** *Wavythin*

Figure 5: ***Error analysis on synthetic data.*** *Top: Ground truth point cloud and estimated 3D line maps. The color represents the 3D direction (The xyz components of* $\mathbf{d}_{\text{prior}}$ *corresponds to RGB, respectively.). Middle: Depth maps. Bottom: Absolute error maps of the depth map. The numbers represent MAE / RMSE (in mm).*

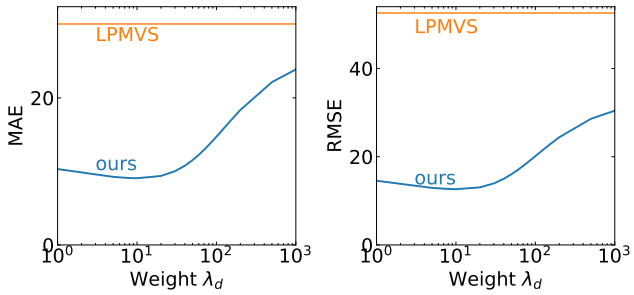


Figure 6: ***Impact of weight parameter on the accuracy.*** *We plot errors resulting from varying* λ_d *on the* Straight *case (Fig 5(a)). Our strand integration produces more accurate depth maps than LPMVS over a wide range of values. Note that this result does not use the normal loss.*

way almost identical to the one used in our experiment reported in the previous section, where the image resolution was 5315×8001. Note that our capture setup was not in the OLAT fashion, so we did not use the normal loss in our strand integration. We used the facer toolkit [ZYZ*22] for generating masks of the hair regions. Since we found that the 3D hair directions estimated by LPMVS tended to be less reliable compared to the synthetic cases, we set $\lambda_d = 5$ so that the depth maps would not get drifted too much by the noisy directional prior.

Result. Similar to our experiment reported in the previous section, Fig. 8 shows both the unfiltered and filtered point clouds resulting from LPMVS [NWKS19] and our method. As expected, our method leaves much more points after filtering compared to LPMVS thanks to the higher accuracy in our optimized depth maps.

5. Conclusion

In this paper, we proposed a refinement method for hair geometry by integrating the gradient of strands with line assumption. We conducted an evaluation using both synthetically rendered images and real photographs. The result verified that our method improves reconstruction quality for different hairstyles. We believe that this result has the potential to benefit various downstream tasks such as (semi-)automatic modeling of long hair strands as well as training of neural networks that infer hair geometry.

Limitations and Future Work. Our method assumes that hair is continuous and coherent everywhere, implying that the adjacent pixels belong to the same hair strand. This assumption can, however, break when the hair is strongly curled or scattered (e.g., the *Curly* case). Nevertheless, our method is less sensitive to discontinuity than the typical normal integration methods thanks to our depth loss (Eq. 7) that partially mitigates this issue by fixing the depth values that are deemed reliable. Our method can also struggle with certain hairstyles (even without curls) where multiple groups of hair strands with different directions are layered on top of each other. Such sudden changes in the strand direction will cause our integration method to generate inaccurate hair shape. This limitation is not unique to our method, as standard normal integration methods also encounter similar challenges. One possible way to overcome this issue may be to adapt the one-sided partial derivatives [CSS*22] to our strand integration task.

While our input is multi-view, our refinement algorithm is inherently single-view, leveraging the multi-view information only indirectly through the use of the consistency weight term in the depth loss (Equation (7)). It may be worthwhile to explore other more direct ways of introducing the multi-view constraints into the refinement process. In particular, integrating our refinement idea into the LPMVS algorithm itself could be an interesting avenue for future work.

To use the resulting oriented point cloud for downstream applications such as rendering and simulation, one would need to convert it into a set of long 3D strands by using the strand-growing algorithm described in the original LPMVS paper or by utilizing it as a prior for neural rendering [RSW*22, WNS*22]. Either way, our refined result will improve the overall quality of these tasks.

References

[CLS*15] CHAI M., LUO L., SUNKAVALLI K., CARR N., HADAP S., ZHOU K.: High-quality hair modeling from a single portrait photo. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 1–10. 2

[CSS*22] CAO X., SANTO H., SHI B., OKURA F., MATSUSHITA Y.: Bilateral normal integration. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part I* (2022), Springer, pp. 552–567. 7

[CSW*16] CHAI M., SHAO T., WU H., WENG Y., ZHOU K.: Autohair: Fully automatic hair modeling from a single image. *ACM Transactions on Graphics 35*, 4 (2016). 2

[CWW*12] CHAI M., WANG L., WENG Y., YU Y., GUO B., ZHOU K.: Single-view hair modeling for portrait manipulation. *ACM Transactions on Graphics (TOG) 31*, 4 (2012), 1–8. 2

[HB86] HORN B. K., BROOKS M. J.: The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing 33*, 2 (1986), 174–208. 2

[HMLL14] HU L., MA C., LUO L., LI H.: Robust hair capture using simulated examples. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 1–10. 2

[HMLL15] HU L., MA C., LUO L., LI H.: Single-view hair modeling using a hairstyle database. *ACM Transactions on Graphics (ToG) 34*, 4 (2015), 1–9. 2

[HZW12] HERRERA T. L., ZINKE A., WEBER A.: Lighting hair from the inside: A thermal approach to hair reconstruction. *ACM Transactions on Graphics (TOG) 31*, 6 (2012), 1–9. 2

[JMM09] JAKOB W., MOON J. T., MARSCHNER S.: Capturing hair assemblies fiber by fiber. In *ACM SIGGRAPH Asia 2009 papers*. 2009, pp. 1–9. 2

[KB15] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. In *Proceedings of International Conference on Learning Representations (ICLR)* (2015). 5

[KCF*22] KUANG Z., CHEN Y., FU H., ZHOU K., ZHENG Y.: Deepmvshair: Deep hair modeling from sparse views. In *SIGGRAPH Asia 2022 Conference Papers* (2022), pp. 1–8. 2

[LLP*12] LUO L., LI H., PARIS S., WEISE T., PAULY M., RUSINKIEWICZ S.: Multi-view hair capture using orientation fields. In *2012 IEEE Conference on Computer Vision and Pattern Recognition* (2012), IEEE, pp. 1490–1497. 2

[LLR13] LUO L., LI H., RUSINKIEWICZ S.: Structure-aware hair capture. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–12. 2

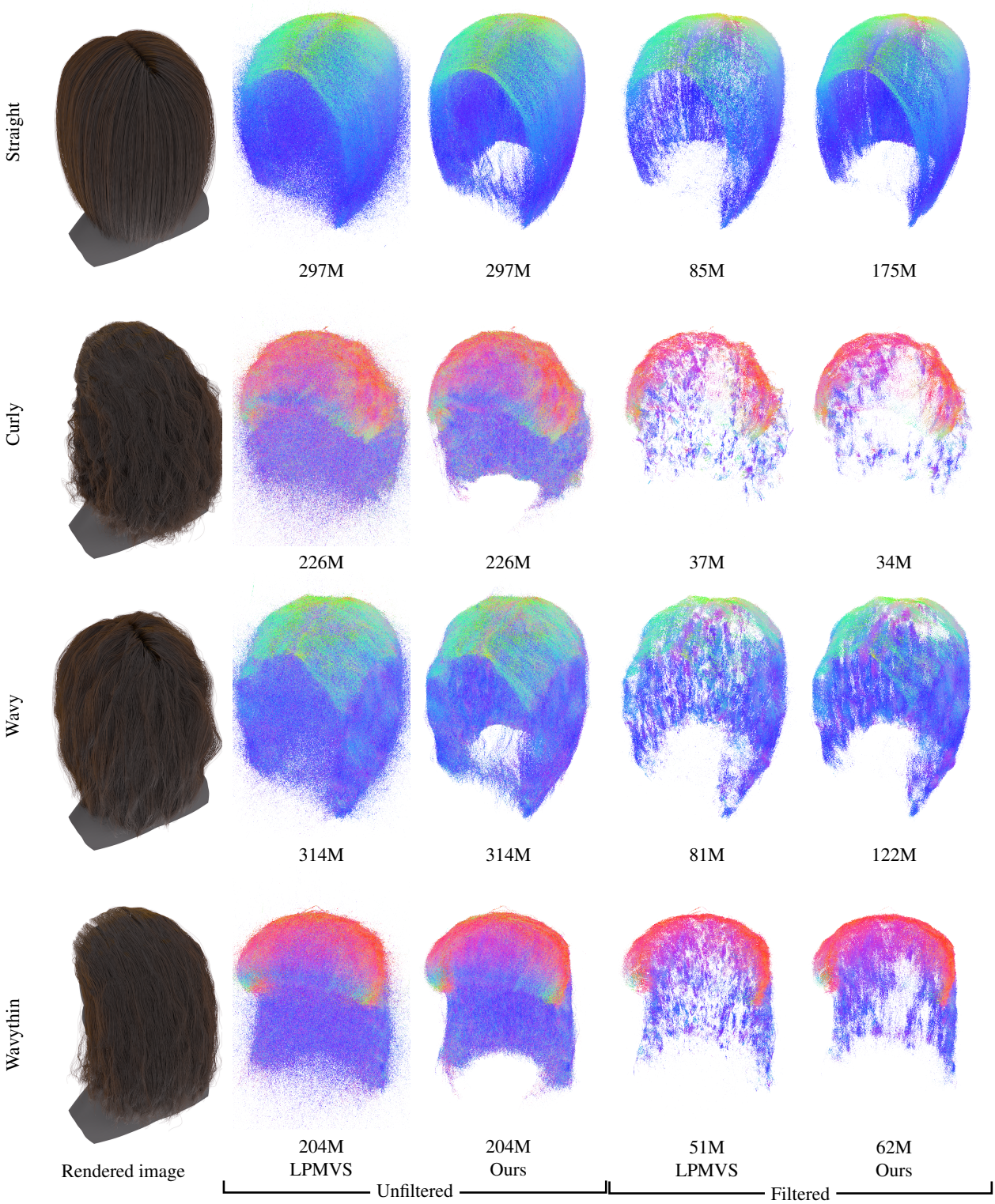


Figure 7: ***Final merged hair shapes.*** *We show both unfiltered and filtered 3D lines. Note that viewpoints of rendered images as a reference are different from the rendered point clouds. The value expressed in millions (M) below is the total count of 3D lines. A higher value indicates denser 3D lines. Before applying the filtering, the raw 3D lines obtained from LPMVS contain a significant number of noisy lines. In contrast, our refinement can reduce noisy lines while preserving the overall line count.*

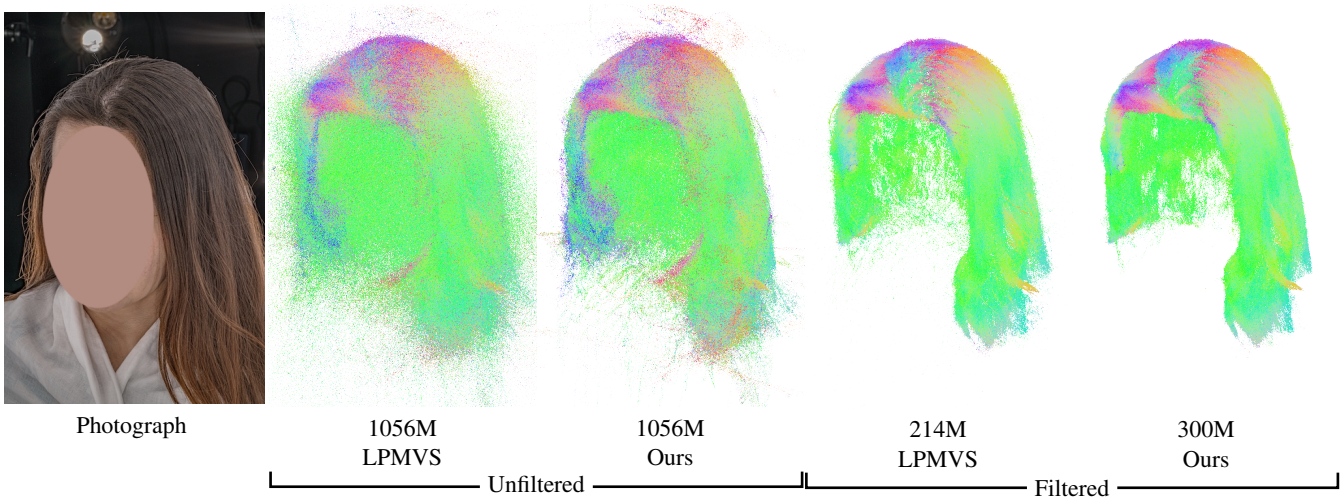


Figure 8: ***Reconstruction result of real hair.*** *The value shown below is the total count of 3D lines in million (M). Also in a real capture scenario, our refinement method achieves less noise and denser 3D lines.*

[LZZR13] LUO L., ZHANG C., ZHANG Z., RUSINKIEWICZ S.: Wide-baseline hair capture using strand-based refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 265–272. 2

[NWKS19] NAM G., WU C., KIM M. H., SHEIKH Y.: Strand-accurate multi-view hair capture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 155–164. 1, 2, 3, 4, 5, 6, 7

[PBS04] PARIS S., BRICENO H. M., SILLION F. X.: Capture of hair geometry from multiple images. *ACM transactions on graphics (TOG) 23*, 3 (2004), 712–719. 2

[PCK*08] PARIS S., CHANG W., KOZHUSHNYAN O. I., JAROSZ W., MATUSIK W., ZWICKER M., DURAND F.: Hair photobooth: geometric and photometric acquisition of real hairstyles. *ACM Trans. Graph. 27*, 3 (2008), 30. 2

[QDA18] QUÉAU Y., DUROU J.-D., AUJOL J.-F.: Normal integration: a survey. *Journal of Mathematical Imaging and Vision 60* (2018), 576–593. 2

[RSW*22] ROSU R. A., SAITO S., WANG Z., WU C., BEHNKE S., NAM G.: Neural strands: Learning hair geometry and appearance from multi-view images. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII* (2022), Springer, pp. 73–89. 2, 7

[SHM*18] SAITO S., HU L., MA C., IBAYASHI H., LUO L., LI H.: 3d hair synthesis using volumetric variational autoencoders. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–12. 2

[SNA*21] SUN T., NAM G., ALIAGA C., HERY C., RAMAMOORTHI R.: Human hair inverse rendering using multi-view photometric data. 2

[WNS*22] WANG Z., NAM G., STUYCK T., LOMBARDI S., ZOLLHÖFER M., HODGINS J., LASSNER C.: Hvh: Learning a hybrid neural volumetric representation for dynamic hair performance capture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 6143–6154. 2, 7

[Woo80] WOODHAM R. J.: Photometric method for determining surface orientation from multiple images. *Optical engineering 19*, 1 (1980), 139–144. 2, 5

[WOQS05] WEI Y., OFEK E., QUAN L., SHUM H.-Y.: Modeling hair from multiple views. In *ACM SIGGRAPH 2005 Papers*. 2005, pp. 816–820. 2

[WYY*22] WU K., YE Y., YANG L., FU H., ZHOU K., ZHENG Y.: Neuralhdhair: Automatic high-fidelity hair modeling from a single image using implicit neural representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 1526–1535. 2

[YSK09] YUKSEL C., SCHAEFER S., KEYSER J.: Hair meshes. *ACM Transactions on Graphics (TOG) 28*, 5 (2009), 1–7. 4

[YZM*23] YUJIAN Z., ZIRONG J., MORAN L., HAIBIN H., CHONGYANG M., SHUGUANG C., XIAOGUANG H.: Hairstep: Transfer synthetic to real using strand and depth maps for single-view 3d hair modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023). 2

[ZCW*17] ZHANG M., CHAI M., WU H., YANG H., ZHOU K.: A data-driven approach to four-view image-based hair modeling. *ACM Trans. Graph. 36*, 4 (2017), 156–1. 2

[ZHX*18] ZHOU Y., HU L., XING J., CHEN W., KUNG H.-W., TONG X., LI H.: Hairnet: Single-view hair reconstruction using convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 235–251. 2

[ZWW*18] ZHANG M., WU P., WU H., WENG Y., ZHENG Y., ZHOU K.: Modeling hair from an rgb-d camera. *ACM Transactions on Graphics (TOG) 37*, 6 (2018), 1–10. 2

[ZYZ*22] ZHENG Y., YANG H., ZHANG T., BAO J., CHEN D., HUANG Y., YUAN L., CHEN D., ZENG M., WEN F.: General facial representation learning in a visual-linguistic manner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 18697–18709. 7