Strand-accurate Multi-view Hair Capture

Giljoo Nam*[1] Chenglei Wu[2] Min H. Kim[1] Yaser Sheikh[2]
[1]KAIST [2]Facebook Reality Labs, Pittsburgh

Abstract

Hair is one of the most challenging objects to reconstruct due to its micro-scale structure and a large number of repeated strands with heavy occlusions. In this paper, we present the first method to capture high-fidelity hair geometry with strand-level accuracy. Our method takes three stages to achieve this. In the first stage, a new multi-view stereo method with a slanted support line is proposed to solve the hair correspondences between different views. In detail, we contribute a novel cost function consisting of both photo-consistency term and geometric term that reconstructs each hair pixel as a 3D line. By merging all the depth maps, a point cloud, as well as local line directions for each point, is obtained. Thus, in the second stage, we feature a novel strand reconstruction method with the mean-shift to convert the noisy point data to a set of strands. Lastly, we grow the hair strands with multi-view geometric constraints to elongate the short strands and recover the missing strands, thus significantly increasing the reconstruction completeness. We evaluate our method on both synthetic data and real captured data, showing that our method can reconstruct hair strands with sub-millimeter accuracy.

1. Introductoin

Hair is an important way to define a person's look, an indispensable part of virtual human, and a key component for many VR and AR applications. While there has been great progress in capturing high-quality face [3], body [35] or even teeth [37], not much attention has been paid to capturing hair geometry. Owing to the microscale geometry of hair strands, the large number of strands, and the heavy occlusions and high similarity between strands, hair is probably one of the most challenging objects to capture for computer vision methods, especially if our goal is for strand-accurate reconstructions. Due to these challenges, directly applying the traditional multi-view stereo (MVS) methods [10, 33] on the multi-view image data cannot achieve satisfactory results, as MVS methods, to achieve a robust correspondence matching, usually assume the local patch to be a plane, which is obviously invalid for hair. Therefore, they are not able to reconstruct the fine-grained strand structures, like flying strands against the background in Figure 1. Thus, to recover the strand structures, many hair capture techniques [15, 26] run the second step of strand fitting to the reconstructed point cloud from MVS. Although the previous methods properly capture the overall shape of hair wisps, the reconstructed geometries often lack small-scale details, e.g., flying strands.

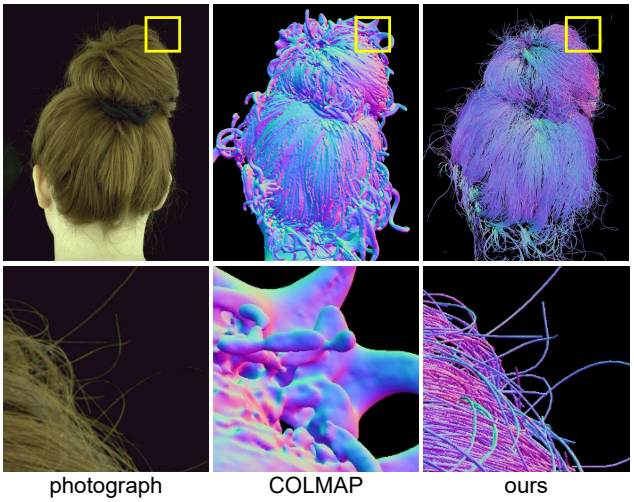

Figure 1. (Left) One of the photographs from multi-view capture. (Middle) Final geometry by traditional MVS (COLMAP [33]). (Right) Final geometry by our method. Our method can produce high-fidelity hair geometry with strand-level accuracy.

To achieve strand-accurate capture, in this work we rethink the right way for multi-view hair capture, and propose a new pipeline to reconstruct hair metrically. As a first step, we reformulate the traditional multi-view stereo algorithm with the strand-line assumption, specifically targeting microscale thin geometry of hair. We propose to use a local line support for effectively matching strand features across views. Note that although hair is a 3D curve, it can be seen as a connected strand of short line segments. To help for matching, we design our novel cost function to not only rely on the color correlation but also a new geometry-consistency term enforcing the projected 2D line from predicted 3D line to be consistent with 2D orientations detected in the images. This cost function is effi-

*Work done during an internship at Facebook Reality Labs, Pittsburgh.

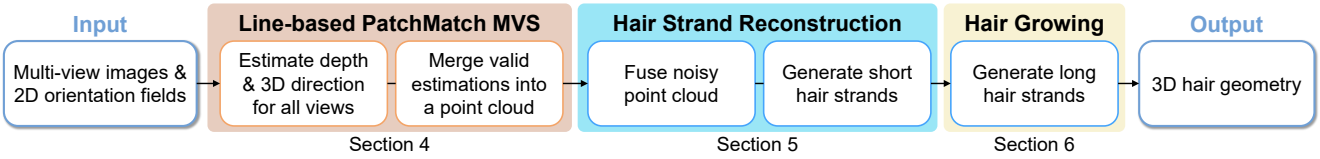

Figure 2. An overview of our complete hair geometry reconstruction pipeline.

ciently minimized using the randomized optimization strategy as PatchMatch [28] to estimate depth maps, which are then merged to a point cloud. At this point, a counterpart of meshing step [19] in MVS is necessary to fuse noisy samples to a unique representation, i.e., strands. Therefore, we contribute a new strand reconstruction algorithm from point cloud using the mean-shift method [9]. The reconstructed strands may be short segments, and some challenging strands may still be missing. We thus take a third step, which propagates the current strands to its adjacent pixels, i.e., growing the strands, by optimizing the same multi-view geometric constraint as that in our MVS.

We evaluate our method on both synthetic and real captured data, demonstrating that our method can achieve submillimeter accuracy quantitatively on the synthetic data and pixel-accurate on the real data when projecting the reconstructed strands to a novel view.

2. Related Work

Multi-view Stereo Multi-view stereo is a well-known method to reconstruct 3D geometry from a set of images captured from different views. One of the major challenges in MVS is to estimate correspondences across views, which is generally attacked by optimizing a photo-consistency function measuring the color similarity of the slanted local patches from different viewpoints. The local patch is generally assumed to be a 3D plane. To optimize this highly nonlinear energy, PatchMatch-based methods [10, 28, 33] have become popular and achieved many successes in MVS benchmarks. However, applying these methods on hair would instantly invalidate this assumption and thus cannot achieve strand-accuracy.

Hair Capture from Dense Multi-view Images A series of pioneered work in hair capture from multi-view images estimate a dense 2D/3D vector field from input images and combine it with other 3D surface constraints, e.g., from structure light or visual hull [30, 31, 36]. To improve the hair capture quality, different sensor modalities have also been investigated, e.g., using depth-of-focus techniques [18], an RGB-D sensor [17] or thermal imaging [14].

Recently, with the success of MVS techniques, the epipolar constraints are explicitly investigated for reconstructing hair. Luo et al. [25, 27] presented capture methods based on the orientation fields detected from captured images to better reconstruct the geometric details of hair. Luo et al. [26] and Hu et al. [15] developed a progressive method to steadily fit hair structures, i.e., ribbons, wisps, and strands, to the point cloud data. As the point cloud is reconstructed from a traditional MVS, these methods still suffer from the plane assumption. Free of the plane assumption, Beeler et al. [4] develop a coupled reconstruction method to capture sparse facial hair by 2D/3D hair growing. However, due to the complex occlusions between hair strands, this method cannot be applied to the dense hair.

Hair Capture from Limited Views Another direction in hair capture is with data-driven methods, especially with limited input views or even with a single view. With a single input image and a few user strokes, Hu et al. [16] retrieved closest hair examples from the database and combined them to match the input hairstyle. Chai et al. have proposed a series of works on single-view hair modeling and manipulation [5, 6, 7, 8], pushing towards high-quality hair modeling without any user interactions. The quality of captured hair can be improved by having four views of input images [40]. A recent trend is to employ deep learning to automatically capture the hair strands from a single image [41]. Although these approaches achieved plausible results, the estimated hair geometry is not metrically accurate.

3D Line/Curve Reconstruction The work in reconstructing 3D line or curve primitives is also related to our method. Structure from motion with line primitives has been proposed by Bartoli et al. [1]. Stereo matching with line segments has been investigated to reconstruct the line structures of architectures [2, 11, 23, 24, 29, 34]. Line-based representation has also been applied to visual SLAM [39, 42]. These methods, however, cannot be applied to hair due to the high curvature and dense occlusion of hair strands.

3. Overview

Figure 2 shows an overview of our method. Our method is three-fold: line-based PatchMatch multi-view stereo (LPMVS) (Section 4), strand reconstruction from point cloud (Section 5) and multi-view hair growing (Section 6). LPMVS takes captured images and 2D orientation fields as input and yields a 3D point cloud $\{P\}$, of which each point represents a line segment. We represent a line segment P with its 3D position and 3D direction $\{P_{pos}, P_{dir}\}$. The strand reconstruction takes the point cloud $\{P\}$ as input and produces strand segments $\{S\}$. We then grow these strand segments into long strands that best match the captured images and 2D orientation fields. We describe each stage next.

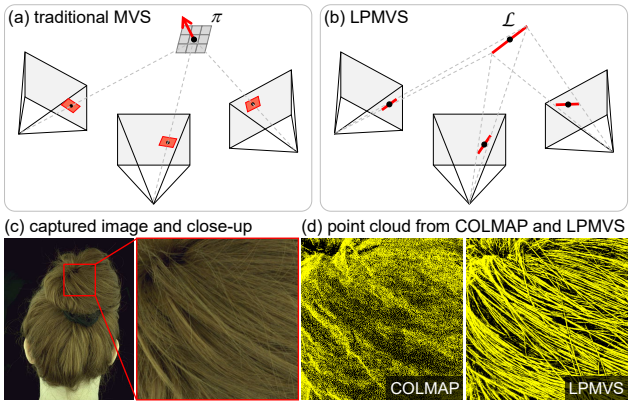

Figure 3. (a) Traditional MVS with a plane assumption. (b) LPMVS with a line assumption. (c) One of the captured images. (d) Point cloud from COLMAP [33] and our LPMVS.

4. Line-based PatchMatch MVS

Previous MVS assumes a 3D local plane is projected to a pixel on a reference view and tries to find the position and the normal of that plane using intensity/color correlation between the reference view and neighboring views (Figure 3a). Analogous to this, our LPMVS assumes a 3D line segment is projected to a pixel and tries to find the position and the 3D direction of the line (Figure 3b).

The overall pipeline of our LPMVS is shown in Algorithm 1. It largely follows that of the PatchMatch MVS [10, 33]. Different from traditional MVS, however, the output of LPMVS is a 3D line map. Each pixel in the 3D line map represents the 3D position (i.e., depth) and the 3D direction of a line. Our contribution in LPMVS is a new cost function which is specifically designed for hair geometry.

Figures 3c and 3d demonstrate the difference of reconstructed point clouds respectively from traditional MVS and our proposed LPMVS. While the output point cloud of MVS [33] exhibits a planar surface shape that follows the plane assumption of MVS, our LPMVS can recover the fine-grained strands of hair geometry, which lays the foundation for strand-accurate hair reconstruction.

4.1. Per-view 3D Line Estimation

Input data The inputs to our LPMVS are calibrated images from multiple views with estimated 2D orientation fields for each image. Here we only use grayscale images and do not use color information. However, incorporating color information should be straightforward. We first generate the 2D orientation fields following previous work [30]. The Gabor filter is used for the convolution kernel and the resolution of kernel rotation is set to one degree, and thus we get 180 responses for each pixel. The variance of the responses are computed following Paris et al. [30]. We use the inverse of squared variance ($conf = 1/var^2$) as a confidence measure of the 2D orientation fields.

Algorithm 1 Line-based PatchMatch MVS

Input: multi-view images and 2D orientation fields
Output: 3D line maps
1: **for** each view **do**
2: set reference and neighboring views
3: randomly initialize a 3D line map
4: **for** iteration $i = 1$ to N_{iter} **do**
5: update the 3D line map via spatial propagation
6: refine the 3D line map via random perturbation
7: **end for**
8: **end for**

Cost Function for LPMVS Given a 2D pixel position p in a reference view, LPMVS tries to find a corresponding 3D line $\mathcal{L}_p$ using its N_{nei} number of neighboring views. A 3D line $\mathcal{L}_p$ is parameterized by the depth value at p and its 3D direction; thus it has three degree-of-freedom. The cost function $m\left(p, \mathcal{L}_p\right)$ defines the loss when the pixel p corresponds to the 3D line $\mathcal{L}_p$. In detail, it consists of two terms, geometric cost $m_g\left(p, \mathcal{L}_p\right)$ and intensity cost $m_c\left(p, \mathcal{L}_p\right)$:

$$m\left(p, \mathcal{L}_p\right) = (1-\alpha) m_g\left(p, \mathcal{L}_p\right) + \alpha m_c\left(p, \mathcal{L}_p\right). \quad (1)$$

These two terms are weighted by α ($\alpha = 0.1$).

2D Sample Points To compute these two energy terms, we use 2D sample points on the reference view and its neighboring views. Figure 4 illustrates the process of generating $S_{p,i}$, which is a set of 2D samples in i-th view (we always refer 0-th view as the reference view). We first project $\mathcal{L}_p$ to the reference image and get the corresponding 2D line $l_{p,0}$. For 3D to 2D line projection, we use Plücker line coordinates (Hartley and Zisserman [13], p.198). We then sample κ number of points uniformly along $l_{p,0}$, centered at the p with radius r_κ, obtaining κ 2D samples for the reference view $S_{p,0}$. Notice that r_κ defines the distance between the farthest sample and p ($\kappa = 41$ and $r_\kappa = 10$ pixels).

Once we obtain $S_{p,0}$, we shoot rays from the reference view's origin towards each sample in $S_{p,0}$. We find intersection points with the 3D line $\mathcal{L}_p$ and re-project the points into i-th view and get corresponding 2D samples $S_{p,i}$.

Geometric Cost The geometric cost $m_g\left(p, \mathcal{L}_p\right)$ defines the difference of the direction of the 3D line $\mathcal{L}_p$ and its corresponding 2D orientations detected from the input images:

$$m_g\left(p, \mathcal{L}_p\right) = \sum_{i=0}^{N_{nei}} \gamma_i g_i\left(S_{p,i}, l_{p,i}\right) \Big/ \sum_{i=0}^{N_{nei}} \gamma_i, \quad (2)$$

where γ_i is the weight for i-th view. $g_i\left(S_{p,i}, l_{p,i}\right)$ defines angular difference between the detected orientation on the 2D samples $S_{p,i}$ and the direction of 2D line $l_{p,i}$:

$$g_i\left(S_{p,i}, l_{p,i}\right) = \sum_{s \in S_{p,i}} c_s \cdot diff\left(\theta_s, l_{p,i}\right) \Big/ \sum_{s \in S_{p,i}} c_s, \quad (3)$$

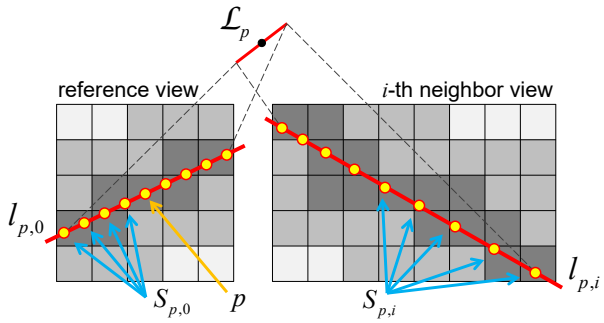

Figure 4. Sampling along the 2D lines in multiple views.

where c_s is the confidence value of the 2D orientation field at the sampled position s, θ_s is the 2D orientation value and $diff\ (\theta, l)$ returns the angle difference between a 2D orientation θ and the 2D direction of a line l. We use γ to weight more on the reference view and equally for the neighboring views, as otherwise, the matching may ignore the reference view cost ($\gamma_0 = N_{nei}, \gamma_{i\neq 0} = 1$).

Intensity Cost Intensity cost is defined as follows:

$$m_c\left(p, \mathcal{L}_p\right) = \frac{1}{N_{nei}} \sum_{i=1}^{N_{nei}} c_i\left(S_{p,i}, S_{p,0}\right), \tag{4}$$

where $c_i\left(S_{p,i}, S_{p,0}\right)$ is 1D normalized cross correlation (NCC) between the intensity values of the two sample sets.

Random Initialization To start LPMVS, we randomly initialize the 3D line map of the reference view. For each pixel p, we generate $\mathcal{L}_p$ by assigning it a random depth value and a random 3D unit vector for the line direction.

Spatial Propagation We use the Red-Black pattern to propagate good estimations to their neighbors following Galliani et al. [10]. Instead of propagating plane parameters, LPMVS propagates 3D line parameters. However, replacing the line parameters is not as straightforward as replacing plane parameters since 3D lines do not intersect at a unique point in general. To solve this, we first shoot a ray from the camera center of the reference view through the reference pixel. We then find a 3D point on the ray that has the minimum distance to the 3D line of the neighboring pixel. A new line $\mathcal{L}$ is defined by that 3D point and the line direction from the neighboring pixel. In this way, we can guarantee the line hypothesis to test is always projecting to the reference pixel. With that, we check if the new line parameters reduce current cost so that we replace the line parameters with the new ones. We repeat this process N_{iter} times ($N_{iter} = 8$).

Line Refinement After each spatial Red-Black propagation, we refine the 3D line map by providing random perturbations w.r.t depth and 3D direction. We follow Galliani et al. [10] for the refinement process. If the new parameters reduce the cost, we replace the old parameters with them.

4.2. 3D Line Filtering

We follow the traditional MVS pipeline [10]; we first compute 3D line maps from all views and merge these line maps into a point cloud. To check the consistency of the estimated lines, we compare 3D position and 3D line direction. For each pixel on the reference view, we project corresponding 3D line position into neighboring views and get 3D lines from the neighboring views. The estimation is consistent with j-th neighbor view if the following criteria are met:

$$\begin{aligned} &\left\|\text{pos}\left(\mathcal{L}_p\right) - \text{pos}\left(\mathcal{L}_{p,j}\right)\right\|_2 < \tau_p, \\ &\text{angle}\left(\text{dir}\left(\mathcal{L}_p\right), \text{dir}\left(\mathcal{L}_{p,j}\right)\right) < \tau_d. \end{aligned} \tag{5}$$

We use $\tau_p = 1\,\text{mm}$ and $\tau_d = 10°$. We keep the reconstructed point from the reference pixel if it is consistent with at least two neighboring views. By running this filtering for each viewpoint, we eventually obtain a point cloud with each point P having its position P_{pos} and direction P_{dir}.

5. Strand Reconstruction from Point Cloud

Similar to previous MVS, after obtaining the point cloud $\{P\}$, we need to fuse the noisy samples to a unique representation, which are strands in our case. A single hair strand is defined as a sequence of connected 3D points $S = \{P_0, \ldots, P_S\}$. Our hair strand reconstruction algorithm is purely based on the captured data, i.e., the point cloud. It has two stages, point cloud fusion and hair strand generation, which we describe in detail in the followings.

5.1. 3D Line Fusion

There are several technical challenges in generating hair strands from the point cloud. First, the point cloud is noisy and has outliers. These come from imperfect calibration, repeated strand patterns, specular reflection, occlusion, etc. Second, hair strands have complex geometric topology, as many strands are clustered and close to each other and are often intertwined. To deal with this, we first perform 3D fusion on the point cloud to shape the point cloud into thin hair strands while being robust to noise and outliers.

Existing 3D point cloud fusion methods, such as moving least-squares (MLS) [22] which is used in previous hair capture work [26, 15], are designed for surface geometry. Since MLS removes high-frequency noises by fitting point cloud into a 3D surface, it would destroy our line structure if applied to our point cloud. Although Lee [21] has extended it for 3D curve reconstruction, it still cannot handle closely located parallel curves or crossing curves which are common in hair geometry. Therefore, to fuse our point cloud while keeping the strand structure, we propose a novel 3D line fusion algorithm based on the mean-shift that effectively generates thin 3D curves from our point cloud. Algorithm 2 describes our mean-shift based line fusion method. This process is performed for each point independently.

Algorithm 2 3D Line Fusion with Mean-Shift

Input: unclean point cloud $\{P\}$, $P = \{P_{pos}, P_{dir}\}$
Output: fused point cloud $\{Q\}$, $Q = \{Q_{pos}, Q_{dir}\}$

for $P \in \{P\}$ **do**
  $Q^{prev} \leftarrow P$
  repeat
    $Q^{next} \leftarrow \text{LocalMean}(Q^{prev})$ ▷ mean-shift
    $d \leftarrow \left\| Q^{next}_{pos} - Q^{prev}_{pos} \right\|_2$
    $Q^{prev} \leftarrow Q^{next}$
  until $d > \tau_s$
  $\{Q\} \leftarrow \{Q\} \cup Q^{next}$
end for

Let P be a 3D point with its position and direction, i.e., $P = \{P_{pos}, P_{dir}\}$. We move the point P to its local mean position and update the direction to its local mean direction. We repeat this moving process (or shifting) until the distance of the movement is less than τ_s ($\tau_s = 0.002\,\text{mm}$).

Finding the mean of multiple 3D lines in the Euclidean space, however, does not have a general solution. Inspired by the 4D compact representation of 3D lines [32], we propose an effective solution (Figure 5). From the point P, we create a plane Π_P with the plane normal P_{dir}. We then treat all the neighboring points of P as 3D lines and find the intersection points $\{X\}$ with plane Π_P. We use a kd-tree for searching neighbors within a radius $r_{nei} = 2.0\,\text{mm}$. The average line $P^* = \{P^*_{pos}, P^*_{dir}\}$ is calculated as follows:

$$P^*_{pos} = \sum_{i=0}^{M} w_i X_{i,pos} \Big/ \sum_{i=0}^{M} w_i, \quad P^*_{dir} = \sum_{i=0}^{M} w_i X_{i,dir} \Big/ \sum_{i=0}^{M} w_i, \tag{6}$$

where $X_0 = P$ and $X_i = \{X_{i,pos}, X_{i,dir}\}$ is the intersection point of the i-th neighboring point and Π_P We compute the bilateral weight w_i to consider both position and direction:

$$w_i = \exp\left(-\frac{\left\| X_{0,pos} - X_{i,pos} \right\|_2^2}{2\sigma_p^2} - \frac{\left(\cos^{-1}(X_{0,dir} \cdot X_{i,dir}) \right)^2}{2\sigma_d^2} \right). \tag{7}$$

We set $\sigma_p = 0.1\,\text{mm}$ and $\sigma_d = \pi/6$. Our mean-shift algorithm is highly efficient for clustering thin 3D curves while being robust to outliers and crossing strands. In addition, the parameters have a clear link to the physical properties of hair strands, i.e., strand thickness σ_p and curve angle σ_d. Figure 5b shows the real example of before and after applying our fusion process to a point cloud.

5.2. Strand Generation

From the fused point cloud $\{P\}$, similar to the previous work [26], we use the forward Euler method for generating hair strands $\{S\}$. A strand segment S is defined as a sequence of 3D points that belong to the same hair strand $S = \{P_0, \ldots, P_S\}$. First, we select a random seed point P^{seed} in the point cloud $\{P\}$ and set it as current strand: $S_c \leftarrow \{P^{seed}\}$. From P^{seed}, we move towards its line direction with step size s. From the moved position, we search neighboring points within a radius τ_r. We discard points with large angle difference τ_a and average the positions and the directions of remaining neighboring points. The averaged position and direction defines a new current point P^{cur}, and we add the point to the current strand: $S_c \leftarrow S_c \cup \{P^{cur}\}$. We repeat this procedure until there is no neighboring point near the moved position. We perform this process for both directions of the initial seed point. Once we complete one iteration of the forward Euler method step, i.e., having generated one strand segment, we remove points from $\{P\}$ that are within τ_r to that strand. We then store current strand to the output strand set: $\{S\} \leftarrow \{S\} \cup \{S_c\}$. We repeat the forward Euler method until no remaining points exist in $\{P\}$. We set $s = \tau_r = 0.1\,\text{mm}$ and $\tau_a = 30°$.

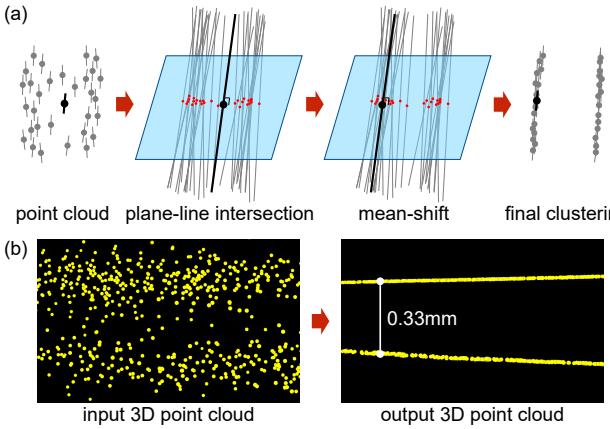

Figure 5. (a) Mean-shift based 3D hair fusion algorithm. (b) Point clouds before and after applying the fusion. The point cloud is a part of that used in Figure 3d.

6. Multi-view Hair Growing

The output of the strand reconstruction is a set of short strand segments, where the average length is usually less than $10\,\text{mm}$. Now we want to grow the short segments to be long strands and reconstruct the hair on the outer surface, which is missed in the LPMVS. Different from previous approaches [26, 15], our method revisits the multi-view constraint to makes sure the hair reconstructed from growing is still metrically meaningful.

The growing algorithm is performed on each strand segment S and on each tip P^{tip} of the segment. We start by projecting the segment onto each view and find 2D growing direction. Let θ_l be the 2D direction of the projected segment at the tip. We sample multiple 2D directions making a 2D cone centered at θ_l with 5° of opening angle and 1° of angular resolution. For each 2D direction, we make a 3×10 window. We compute the score for each directional window by averaging angle difference between the 2D pixel orien-

tation θ_o and the segment direction θ_l. When averaging, we ignore the pixels with small confidence measure for the orientation value. We also ignore the pixels with large angle difference from θ_l ($> 5^\circ$), as we prefer the growing direction not affected by crossing strands. Then we take the 2D direction with the lowest score as the 2D growing direction. We do not take the 2D growing direction if the number of scored pixels is less than 10.

Let N' be the number of views with valid 2D growing direction. Finding 3D growing direction is equivalent to finding the intersection of N' 2D planes, each of which is defined by the camera center of each view and its 2D growing direction. In practice, however, the planes do not intersect to form a unique line due to occlusions. Therefore we formulate this as a minimization problem with a multi-view geometric line constraint. Let $\mathbf{H}$ be a $N' \times 3$ matrix, where i-th row represents the plane normal from i-th view, and $\mathbf{g}$ be the 3D growing direction as a unit vector:

$$\underset{\mathbf{g}}{\text{minimize}} \, \|\mathbf{Hg}\|_2^2 \qquad s.t. \;\; \|\mathbf{g}\|_2 = 1. \tag{8}$$

To solve this equation, we can perform SVD on $\mathbf{H} = \mathbf{U\Sigma V}^\intercal$, and select the column vector in $\mathbf{V}$ that corresponds to the smallest Eigen value in $\mathbf{\Sigma}$. This is however prone to outliers. We thus apply iteratively re-weighted least squares for robust optimization. In detail, we first solve Eq. (8) and get residual $\mathbf{r} = \mathbf{Hg}$. We then re-weight each row in $\mathbf{A}$ with $1/\mathbf{r}^2$ and solve Eq. (8). Two iterations are enough for efficient outlier rejection. Notice that by solving Eq. (8), we guarantee that the estimate 3D growing direction is minimizing the same multi-view geometric cost in Eq. (1).

After we find the 3D growing direction $\mathbf{g}$, we elongate the strand by adding a new 3D point to the segment: $P^{new} = P^{tip} + s_g \cdot \mathbf{g}$, $S \leftarrow S \cup \{P^{new}\}$, where s_g is a growing step size. We set $s_g = 0.1\,\mathrm{mm}$ which roughly corresponds to a pixel width on captured images. We terminate growing hair strands if one of the following criteria is met: : a) Less than N_p ($N_p = 8$) images return 2D growing direction; b) The new 3D point falls onto background area; c) 3D growing direction rapidly changes from previous growing ($> 45^\circ$). The background area is determined by simple intensity thresholding on captured images.

7. Results

7.1. Quantitative Evaluation on Synthetic Data

As there is no way to scan the ground-truth hair geometry, we evaluate our method on the synthetic data, which is rendered using Maya and V-Ray on the hair geometry from Yuksel et al. [38]. We create two data sets for evaluation, one with 280 images of curly hair and the other with 100 images of straight hair. See Figure 6 for example images. We run our method on both data sets to reconstruct the hair strands. The results are visualized in Figure 6.

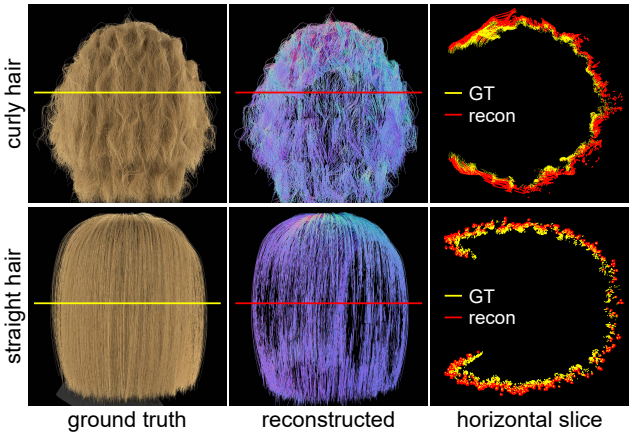

Figure 6. Evaluation using synthetic datasets. (Left) Rendered images using a ground-truth hair geometry. (Middle) Reconstructed hair geometry. (Right) A horizontal slice of the ground-truth (yellow) and the reconstructed geometry (red).

	τ_p (mm)	τ_d (deg.)	Precision (%)	Recall (%)	F-score
curly	0.50	5.00	46.02	14.54	22.10
	1.00	10.00	74.31	25.32	37.77
	2.00	20.00	94.91	43.71	59.86
straight	0.50	5.00	72.94	23.37	35.39
	1.00	10.00	92.94	31.44	46.98
	2.00	20.00	99.20	45.46	62.35

Table 1. Precision and recall of synthetic dataset curly hair and straight hair with various threshold values. About 75% of reconstructed points of curly hair and 90% of points of straight hair suffice sub-millimeter accuracy with less than 10° error in tangential directions.

Note that the large portion of hair strands of the ground truth is in the inner part, which is heavily occluded from captured images. For a fair evaluation, we assess our method by comparing against only the outer stands of the ground truth. We thus obtain new reference strands by removing the inside strands in the ground truth that are far ($>$10 mm) from the outer surface. The horizontal slices in Figure 6 shows the ground truth strands and the reconstructed ones. Qualitatively, we are able to reconstruct accurately the outer surface of the ground truth.

With our reconstruction, we conduct quantitative analysis on reconstructed point positions and directions to measure precision (a.k.a. accuracy) and recall (a.k.a. completeness) similar to that in the traditional MVS methods [20]. Table 1 shows the quantitative evaluation results, where τ_p and τ_d are thresholds for estimated position and direction of strand points. We validate points if and only if they satisfy both τ_p and τ_d. F-score is defined as harmonic mean of precision and recall as in the MVS benchmark [20]. The results demonstrate high accuracy on both curly and straight hair dataset using our method. For the curly hair, $\sim$75% of reconstructed points have errors less than $1.0\,\mathrm{mm}$ and 10°. For the straight hair, more than 90% of points show sub-millimeter accuracy.

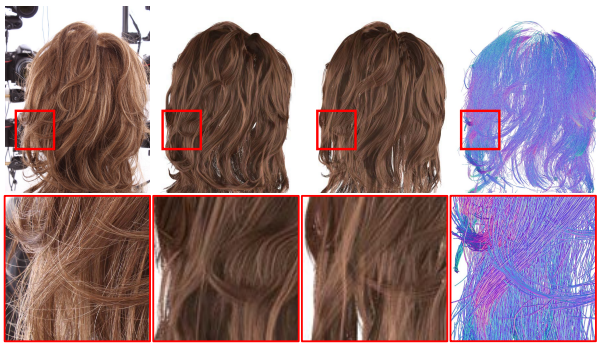

Figure 7. Comparison between our method and previous work (Luo et al. [26] and Hu et al. [15]). The output geometries from previous work lose fine details of hair strands. In contrast, our method recovers the original strands *as is*.

7.2. Comparison with Previous Work

Figure 7 shows a comparison between our method and previous methods [26, 15] on their dataset, which takes 46 views of images as input. While the previous work can produce the overall shape of the hairstyle by synthesizing ribbons and wisps, the output geometry lacks fine details of hair strands, like flying hair or random hair. In contrast, our method captures hair strands *as is* from input images and thus can recover details of the original strands (Figure 7 bottom). Note that due to the low coverage, some of the occluded part is not reconstructed by our method, as we do not perform plausible hair strand synthesis if they are not multi-view constrainable, while previous methods do.

Comparison with Traditional MVS We compare our method with one of the best previous MVS methods, i.e., COLMAP [33] (Figure 1). Our method captures much more fine-grained strand structures than COLMAP, thanks to our line-based processing pipeline.

7.3. Evaluation on Real Data

Capture System To demonstrate our method on the real captured dataset, we employ a multi-camera system with 70 machine vision cameras with 4096×2668 resolution and running at 30 fps. The cameras are located on a spherical structure covering the half of the sphere. The distances between a subject and the cameras are about one meter so that a single hair strand is captured with 1 – 2 pixels on images. All cameras are calibrated both intrinsically and extrinsically with a calibration target [12]. 300 LEDs are evenly distributed on the sphere to provide close-uniform lighting to reduce the specularity.

Leave-one-out Evaluation To evaluate our reconstruction from an unseen viewpoint, we intentionally leave one camera view out in the reconstruction, i.e., we use 69 camera views for reconstructing the hair geometry. We then project it to the unseen view for evaluation. The results shown in Figure 8 demonstrate that our re-projection is pixel accurate.

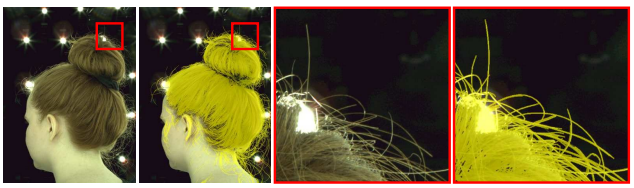

Figure 8. Reconstructed hair geometry from our method is projected to a novel view photograph which is not included in the reconstruction, demonstrating the pixel-accuracy of our method.

Qualitative Evaluation on Real Data To evaluate our method, we have captured 6 real datasets to cover various hairstyles including short/long, curly/straight, dark/bright hair. Out of these, four are the real hair from actors and two are wigs. Figure 9 shows the captured images and reconstructed hair geometry. The close-up figures clearly demonstrate the accuracy of our method, achieving strand-level agreement between the captured images and reconstructed hair. Notice that in Figure 9 we have converted hair strands $\{S\}$ into cylinder mesh just for visualization.

Dynamic Hair Our method runs per frame and thus can be readily applied to a video sequence. Notice that we do not enforce temporal coherency, and reconstruct hair frame by frame. The reconstructed dynamic hair sequence demonstrates that our reconstructed strands are temporally consistent, further confirming the accuracy of our method. Please refer to the supplemental video for the results.

Running Time Our method is implemented in C++ and CUDA. We run experiments on a machine with a Intel(R) Xeon(R) CPU and 8 NVIDIA Tesla V100 GPUs. It takes ∼40 seconds to estimate a 3D line map for a single view, and we process multiview images in parallel using multiple GPUs. Hair fusion takes ∼100 seconds to process 10 – 20 million points. Hair strand generation takes less than one minute, and hair growing takes ∼20 seconds. The total processing time for a single frame with 70 views is ∼15 minutes.

8. Conclusion

In this paper, we propose the first method to achieve strand-accurate hair reconstruction from a multi-view setup. Our contributions include a line-based PatchMatch MVS, a line-based strand fusion and reconstruction algorithm from a point cloud, and a multi-view hair growing method. There are several directions we want to look into as future work. Comparing with many hair capture works, we do not connect our strands to the scalp, which may be necessary for hair simulation applications. We may improve the reconstruction performance on the occluded part by incorporating the view optimization into our framework. Exploring the temporal information in a dynamic sequence may improve the reconstruction coverage, and getting a temporally coherent hair reconstruction or hair tracking may look challenging but exciting for future directions.

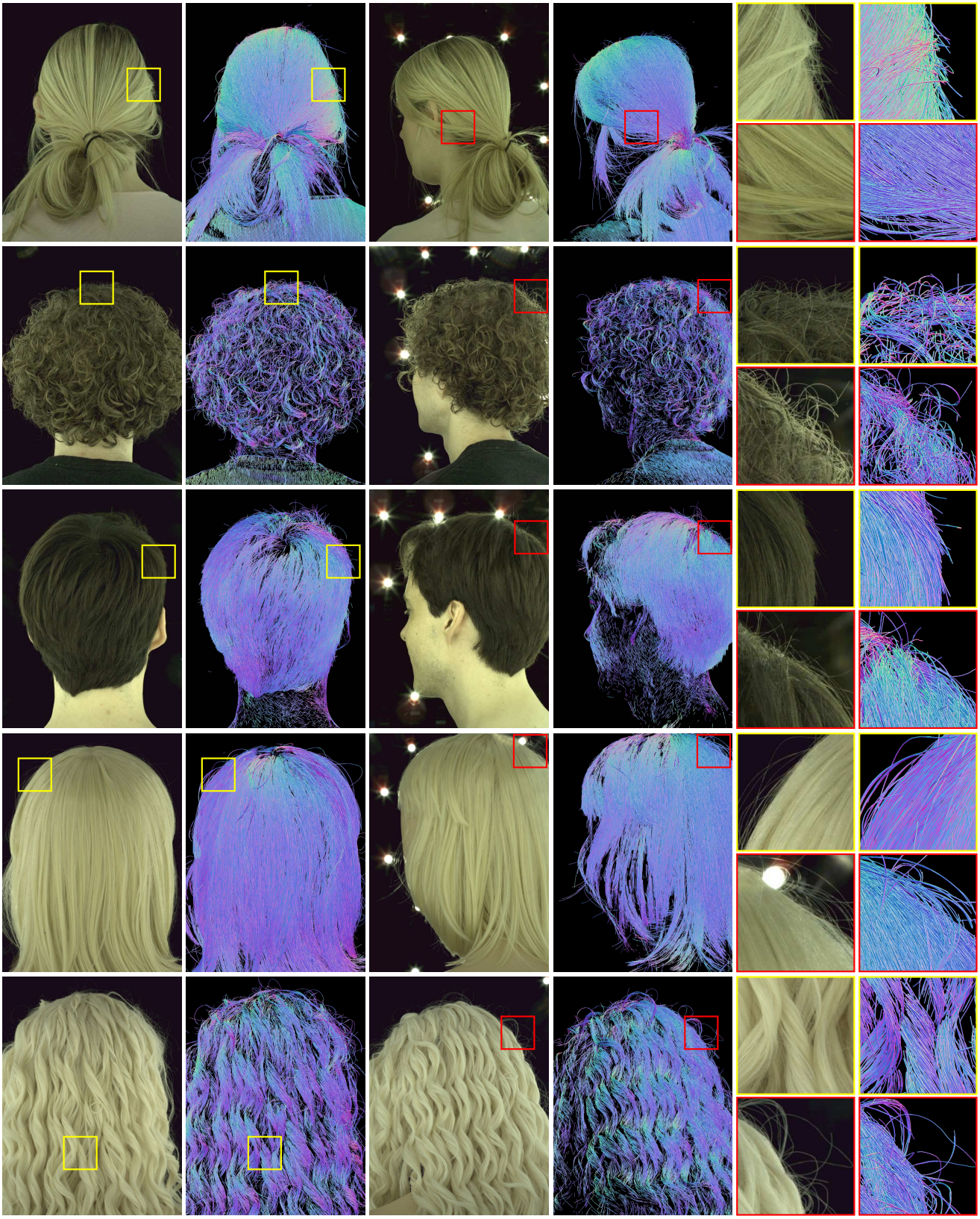

Figure 9. Results from real datasets. The top three rows show the real hair of actors and the bottom two rows show wigs. We capture various hair styles including short/long, curly/straight, dark/bright hair in high accuracy.

References

[1] Adrien Bartoli and Peter Sturm. Structure-from-motion using lines: Representation, triangulation, and bundle adjustment. *Computer vision and image understanding*, 100(3):416–441, 2005. 2

[2] Herbert Bay, Vittorio Ferraris, and Luc Van Gool. Wide-baseline stereo matching with line segments. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 329–336. IEEE, 2005. 2

[3] Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. High-quality single-shot capture of facial geometry. *ACM Trans. Graph.*, 29(4):40:1–40:9, July 2010. 1

[4] Thabo Beeler, Bernd Bickel, Gioacchino Noris, Paul Beardsley, Steve Marschner, Robert W Sumner, and Markus Gross. Coupled 3d reconstruction of sparse facial hair and skin. *ACM Transactions on Graphics (ToG)*, 31(4):117, 2012. 2

[5] Menglei Chai, Linjie Luo, Kalyan Sunkavalli, Nathan Carr, Sunil Hadap, and Kun Zhou. High-quality hair modeling from a single portrait photo. *ACM Transactions on Graphics (TOG)*, 34(6):204, 2015. 2

[6] Menglei Chai, Tianjia Shao, Hongzhi Wu, Yanlin Weng, and Kun Zhou. Autohair: fully automatic hair modeling from a single image. *ACM Transactions on Graphics*, 35(4), 2016. 2

[7] Menglei Chai, Lvdi Wang, Yanlin Weng, Xiaogang Jin, and Kun Zhou. Dynamic hair manipulation in images and videos. *ACM Transactions on Graphics (TOG)*, 32(4):75, 2013. 2

[8] Menglei Chai, Lvdi Wang, Yanlin Weng, Yizhou Yu, Baining Guo, and Kun Zhou. Single-view hair modeling for portrait manipulation. *ACM Transactions on Graphics (TOG)*, 31(4):116, 2012. 2

[9] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, May 2002. 2

[10] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 873–881, 2015. 1, 2, 3, 4

[11] Ruben Gomez-Ojeda and Javier Gonzalez-Jimenez. Robust stereo visual odometry through a probabilistic combination of points and line segments. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2521–2526. IEEE, 2016. 2

[12] H. Ha, M. Perdoch, H. Alismail, I. S. Kweon, and Y. Sheikh. Deltille grids for geometric camera calibration. In *2017 IEEE International Conference on Computer Vision (ICCV)*, volume 00, pages 5354–5362, Oct. 2018. 7

[13] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 3

[14] Tomas Lay Herrera, Arno Zinke, and Andreas Weber. Lighting hair from the inside: A thermal approach to hair reconstruction. *ACM Transactions on Graphics (TOG)*, 31(6):146, 2012. 2

[15] Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. Robust hair capture using simulated examples. *ACM Transactions on Graphics (TOG)*, 33(4):126, 2014. 1, 2, 4, 5, 7

[16] Liwen Hu, Chongyang Ma, Linjie Luo, and Hao Li. Single-view hair modeling using a hairstyle database. *ACM Transactions on Graphics (TOG)*, 34(4):125, 2015. 2

[17] Liwen Hu, Chongyang Ma, Linjie Luo, Li-Yi Wei, and Hao Li. Capturing braided hairstyles. *ACM Transactions on Graphics (TOG)*, 33(6):225, 2014. 2

[18] Wenzel Jakob, Jonathan T Moon, and Steve Marschner. Capturing hair assemblies fiber by fiber. *ACM Transactions on Graphics (TOG)*, 28(5):164, 2009. 2

[19] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*, SGP '06, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. 2

[20] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):78, 2017. 6

[21] In-Kwon Lee. Curve reconstruction from unorganized points. *Computer aided geometric design*, 17(2):161–177, 2000. 4

[22] David Levin. The approximation power of moving least-squares. *Mathematics of Computation of the American Mathematical Society*, 67(224):1517–1531, 1998. 4

[23] Shiwei Li, Yao Yao, Tian Fang, and Long Quan. Reconstructing thin structures of manifold surfaces by integrating spatial curves. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2

[24] Lingjie Liu, Duygu Ceylan, Cheng Lin, Wenping Wang, and Niloy J Mitra. Image-based reconstruction of wire art. *ACM Transactions on Graphics (TOG)*, 36(4):63, 2017. 2

[25] Linjie Luo, Hao Li, Sylvain Paris, Thibaut Weise, Mark Pauly, and Szymon Rusinkiewicz. Multi-view hair capture using orientation fields. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1490–1497. IEEE, 2012. 2

[26] Linjie Luo, Hao Li, and Szymon Rusinkiewicz. Structure-aware hair capture. *ACM Transactions on Graphics (TOG)*, 32(4):76, 2013. 1, 2, 4, 5, 7

[27] Linjie Luo, Cha Zhang, Zhengyou Zhang, and Szymon Rusinkiewicz. Wide-baseline hair capture using strand-based refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 265–272, 2013. 2

[28] Christoph Rhemann Michael Bleyer and Carsten Rother. Patchmatch stereo - stereo matching with slanted support windows. In *Proc. BMVC*, pages 14.1–14.11, 2011. 2

[29] AO Ok, Jan Dirk Wegner, Christian Heipke, Franz Rottensteiner, Uwe Sörgel, and V Toprak. Accurate matching and reconstruction of line features from ultra high resolution stereo aerial images. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences:[ISPRS Hannover Workshop 2011: High-Resolution Earth Imaging For Geospatial Information] 38-4 (2011), Nr.*

W19, volume 38, pages 215–220. Göttingen: Copernicus GmbH, 2011. 2

[30] Sylvain Paris, Hector M Briceño, and François X Sillion. Capture of hair geometry from multiple images. *ACM transactions on graphics (TOG)*, 23(3):712–719, 2004. 2, 3

[31] Sylvain Paris, Will Chang, Oleg I Kozhushnyan, Wojciech Jarosz, Wojciech Matusik, Matthias Zwicker, and Frédo Durand. Hair photobooth: geometric and photometric acquisition of real hairstyles. In *ACM Transactions on Graphics (TOG)*, volume 27, page 30. ACM, 2008. 2

[32] Kenneth S Roberts. A new representation for a line. In *Computer Vision and Pattern Recognition, 1988. Proceedings CVPR'88., Computer Society Conference on*, pages 635–640. IEEE, 1988. 5

[33] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision*, pages 501–518. Springer, 2016. 1, 2, 3, 7

[34] Anil Usumezbas, Ricardo Fabbri, and Benjamin B Kimia. From multiview image curves to 3d drawings. In *European Conference on Computer Vision (ECCV)*, pages 70–87. Springer, 2016. 2

[35] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. Dynamic shape capture using multi-view photometric stereo. *ACM Trans. Graph.*, 28(5):174:1–174:11, Dec. 2009. 1

[36] Yichen Wei, Eyal Ofek, Long Quan, and Heung-Yeung Shum. Modeling hair from multiple views. In *ACM Transactions on Graphics (ToG)*, volume 24, pages 816–820. ACM, 2005. 2

[37] Chenglei Wu, Derek Bradley, Pablo Garrido, Michael Zollhöfer, Christian Theobalt, Markus Gross, and Thabo Beeler. Model-based teeth reconstruction. *ACM Trans. Graph.*, 35(6):220:1–220:13, Nov. 2016. 1

[38] Cem Yuksel, Scott Schaefer, and John Keyser. Hair meshes. *ACM Transactions on Graphics (TOG)*, 28(5):166, 2009. 6

[39] Guoxuan Zhang, Jin Han Lee, Jongwoo Lim, and Il Hong Suh. Building a 3-d line-based map using stereo slam. *IEEE Transactions on Robotics*, 31(6):1364–1377, 2015. 2

[40] Meng Zhang, Menglei Chai, Hongzhi Wu, Hao Yang, and Kun Zhou. A datadriven approach to four-view image-based hair modeling. *ACM Trans. Graph*, 36(4):156, 2017. 2

[41] Yi Zhou, Liwen Hu, Jun Xing, Weikai Chen, Han-Wei Kung, Xin Tong, and Hao Li. Hairnet: Single-view hair reconstruction using convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 235–251, 2018. 2

[42] Xingxing Zuo, Xiaojia Xie, Yong Liu, and Guoquan Huang. Robust visual SLAM with point and line features. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, pages 1775–1782, 2017. 2