Received: 11 October 2017 | Revised: 16 April 2018 | Accepted: 28 May 2018

DOI: 10.1002/cav.1855

RESEARCH ARTICLE

WILEY

Game-ready 3D hair model from a small set of images

Nuttapon Vanakittistien[1] | Attawith Sudsang[1] | Nuttapong Chentanez[1,2]

[1]Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 254 Pathum Wan District, Bangkok, Thailand

[2]NVIDIA, Santa Clara,CA,USA

Correspondence
Nuttapong Chentanez, Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, 254 Pathum Wan District, Bangkok, Thailand.
Email: nuttapong26@gmail.com

Funding information
Development and Promotion of Science and Technology Talents Project

Abstract

We present a system for creating a hair model that matches a user's hairstyle from images. The model consists of guide hair strands and can be used in a real-time hair simulator. Our goal differs from most previous works that aim to create realistic high-resolution hair for off-line applications or create a mesh of the exterior of the hair volume for image manipulation. Our primary aim is for a user to be able to put his/her hairstyle into game or other real-time applications. By taking photos in eight views of the user's head using a smartphone camera and segmenting the images with some easy-to-use tools, the player will obtain his/her own hair model in NVIDIA's HairWorks, which is a hair simulator used in many games. We show a number of results demonstrating the capabilities of our system in this paper.

KEYWORDS

hair modeling, hair simulation, physical simulation, physics-based animation, reconstruction

1 | INTRODUCTION

Most characters in games have their own personalities. When players create their own game characters, they are likely to express their personal features into their avatars. In many games, players can customize the face, body, and hairstyle of the avatars. The more choices the game provides, the more varieties of characters there are. This potentially makes the game more interesting and enjoyable. In most games, players can only select from a handful of precreated hairstyles, which prevent the players from matching their character's hair to his/her own. These hairstyles are created by skillful artists who spend a lot of time crafting them. An average game player is not likely to possess the skill required to create such convincing hairstyles; therefore, there are some needs for automatic/assistive hairstyle creation for players. There are many previous works that reconstructed hairstyles from images; however, most works do not aim to reconstruct guide hair strands. They reconstruct either each individual hair in which there are large numbers or a mesh that represents the exterior volume of the hair. These representations are not ideal for real-time hair simulation and rendering for modern games. In this paper, we propose a system that utilizes ideas from several recent works on hair capture and add several components to allow average users to be able to capture his/her own hair and export the result to be used in NVIDIA's HairWorks.[1] This manuscript extends from our conference paper[2] by improving the image capture app, adding automatic pose estimation, improving the background removal process, overlapping the orientation extraction with background removal, and optimizing the running time by almost two orders of magnitude.

Our contributions include the following:

- a system for constructing a hair model starting from acquiring photographs on a smartphone to creating simulation-ready three-dimensional (3D) guide hair strands;
- a user interface for aiding the user in taking photographs;

- a simple method to improve OpenFace two-dimensional (2D) landmark extraction quality by using a video generated from a single image instead of the image itself;
- the use of orthogonal projection to avoid the need for camera calibration and utilizing the least squares fit to handle the discrepancy between views during the head pose estimation step and the 3D orientation field reconstruction step;
- a supersampling method for 2D orientation field filtering to improve the sharpness of the field;
- heuristics from trimming erroneous guide hair strands.

2 | RELATED WORKS

2.1 | Hair capture and reconstruction

A number of works have attempted to reconstruct hair from images. Paris et al.[3] took photos and tested various filters to decide on which is best for estimating a 2D orientation map. Wei et al.[4] proposed a method for combining 2D orientation maps from several views into a single 3D orientation field. Paris et al.[5] also proposed a method for diffusing a 3D orientation of known voxels to unknown voxels by utilizing a tensor representation. Jakob et al.[6] proposed a method for capturing hair strand by strand. Chai et al.[7,8] created the mesh of the exterior of the hair from a single photo of a human face with a few guiding strokes drawn by the user. Luo et al.[9] reconstructed hair from point cloud data. Hu et al.[10] proposed a hair reconstruction technique that utilizes a simulated hair strand example database. Braided hairstyle is reconstructed in the work of Hu et al.[11] In another work of Hu et al.,[12] a user's strokes were compared to a database of hairstyles, and the information was used to synthesize a new hair model. Chai et al.[13] generated a high-quality hair model for 3D printing application. Chai et al.[14] did not require user strokes to generate the hair model from a single photo; however, they still need to make a strong assumption about the appearance of the hair at the back of the head. Most recently, Zhang et al.[15] used four-sided hair images for hair modeling. To generate a rough model shape, they need a hair model database to match each side's image separately and combine them together.

2.2 | Real-time hair simulation

In recent years, real-time hair simulation techniques based on position-based dynamics (PBD)[16,17] have become popular. PBD has been used for simulating many types of object in the real world, such as cloth, soft body,[16] liquid,[18] smoke, sand,[19] and hair.[20] Han and Harada[21] proposed a real-time hair simulation using local and global shape constraints. Kim et al.[22] proposed a long-range attachment constraint to enforce inextensibility for hair simulation with PBD. Umetani et al.[23] proposed a method for simulating curly hair in PBD by using constraints on Darboux vectors. The PBD hair simulation method is implemented in NVIDIA HairWorks,[1] which has been used in many games such as Call of Duty: Ghosts, Far Cry 3, and The Witcher 3: Wild Hunt and can be used in popular game engines such as Unreal Engine 4.

3 | METHOD

The overview of our method is shown in Figure 1. The following sections describe the details of all the steps.

3.1 | Image data acquisition

3.1.1 | Capturing setup

We create a mobile application prototype for an iOS device to capture photographs of the face and the hairstyle from various views. Images of the user's head from eight views are taken with the app either by another person or by the user herself/himself with a long selfie stick. Figure 2 show examples of such images. The views are 45° apart starting from 0° (front of the player's face) to 360°. To guide the user to the correct view, the application has a template face for each view displayed. The user can then adjust the camera position to roughly match the template. Moreover, we add an indicator line above the head template. Users can align the top of the hair to the same height in every image. The eight images are saved for further processing in the next step.

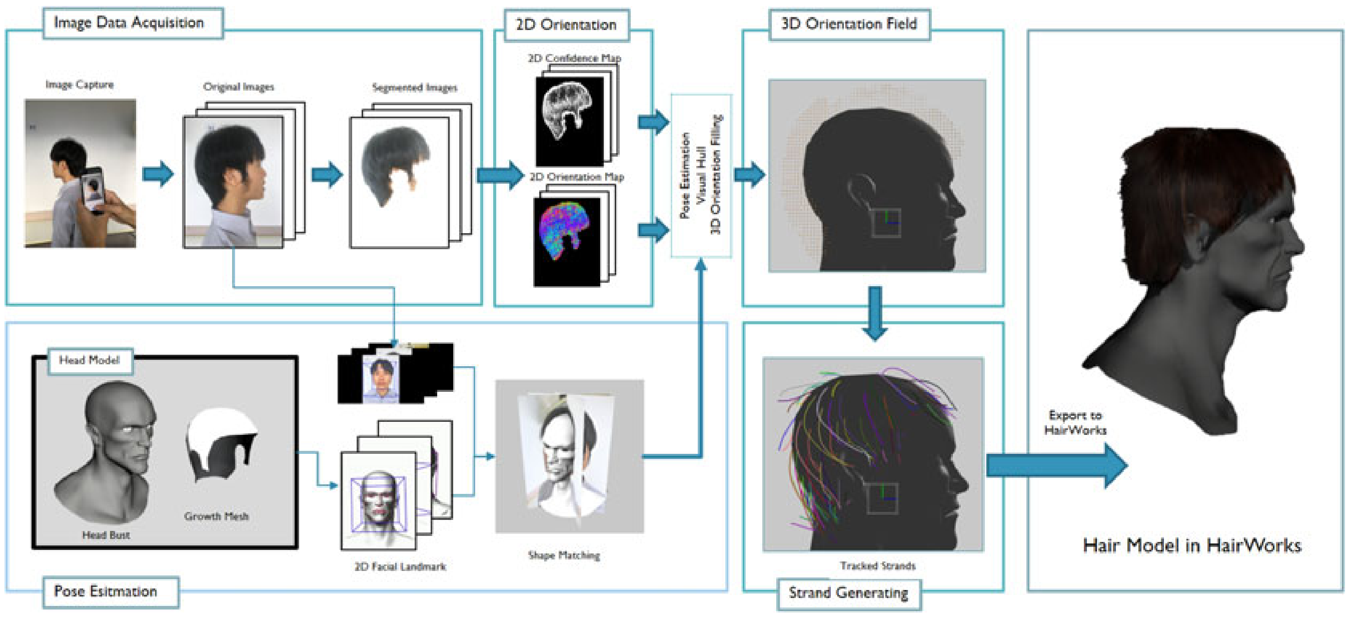

FIGURE 1 The overview of our guide hair strand capture method

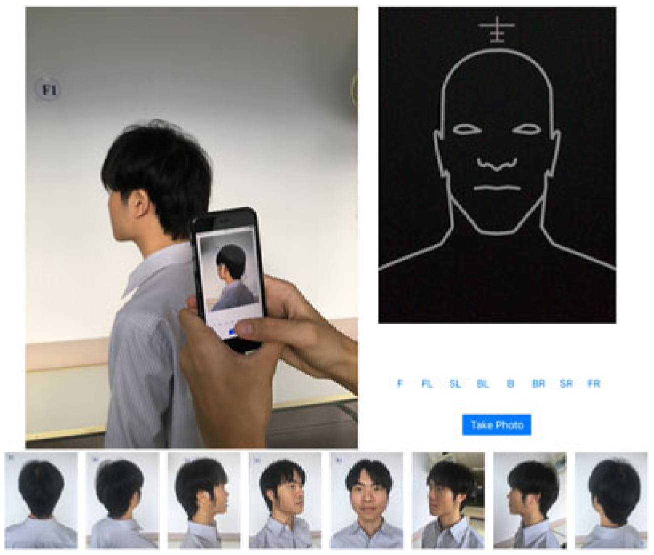

FIGURE 2 Left: process of taking a photo of a user head from eight views. Another user can hold the smartphone camera and use the capturing application or the user can use a long selfie stick. Right: our mobile application prototype interface, which is developed for an iOS device. Bottom: the images from the eight views taken with our app

3.1.2 | User image segmentation

We develop a software for constructing the hair model. After the user imports the eight images into our software, the first step is to segment the hair region out of each image. We utilize GrabCut,[24] which allows the user to draw a few simple lines or shapes to indicate the hair or nonhair regions. Our software then segments out the image to keep only the regions with hair. Note that the regions need not be a perfect segmentation as our algorithm can tolerate some errors, which make this step easy for the user. Typically, the user only needs to draw three to five strokes per image. An example of annotation is shown in Figure 3. Brightness can also optionally be adjusted in this step, simply by selecting one of the available presets. This makes the hair strands more clearly visible for the remaining processing steps.

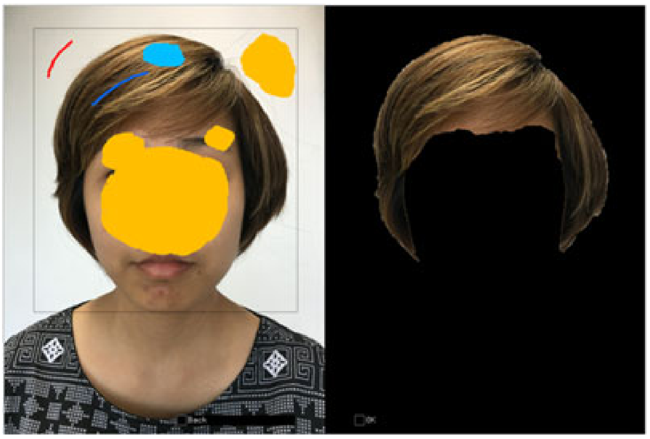

FIGURE 3 Left: an input image that was annotated with the following tools: rectangle tool, foreground tool (blue mark), background tool (red mark), filled convex shape foreground tool (cyan mark), filled convex shape background tool (yellow mark). Right: the resulting segmented image

3.1.3 | Pose estimation

After the images are segmented, we perform a pose estimation for fitting our head model to the image. We experimented with using OpenFace[25] to find a facial landmark on each of the following images' face: frontal face view, front left face view, and front right face view. However, we found that OpenFace occasionally cannot reliably detect the facial landmarks from a single image, especially on the side views. Nonetheless, we found a simple solution, which is running OpenFace on a carefully generated video instead of a single image. The video contains translation and zooming of the image. OpenFace will then adjust the landmarks frame by frame, so that the last frame contains more accurate landmarks compared to just using a single image. The video's resolution is 1920 × 1080 pixels and contains 64 frames. Our software creates the video automatically from the input image by randomly translating and zooming in on the image and uses FFmpeg[26] to encode frames into an mp4 file. This step is demonstrated in our accompanying video.

Two-dimensional facial landmarks from −45°, 0°, and 45° rotation images are then used to find the location of 3D facial landmarks. We make an assumption that the 2D facial landmarks are the result of the orthogonal projection of 3D facial landmarks. This is not strictly correct as the mobile phone camera is not orthogonal; nonetheless, this assumption allows for reconstruction without camera calibration, and it produces good-quality results. We perform the least squares fit to find the 3D location of the facial landmarks such that the root-mean-square error to the 2D facial landmarks, when orthogonally projected onto the three views, is minimum.

We also compute the location of the 3D landmarks of our template 3D head model using the same method, but using the rendered images instead of photographs as shown in Figure 4. The 3D facial landmarks of the captured image and the template head model are then used to compute the optimal transformation that fits the 3D head model to the photographs by using shape matching.[27] Let x_i^0 be the set of 3D landmark points from images, and x_i is the 3D landmark point on the template 3D head model. We find the optimal translation, t; rotation, R; and scale, s, by minimizing

$$\sum_i w_i(sR(x_i - t) - x_i^0)^2, \tag{1}$$

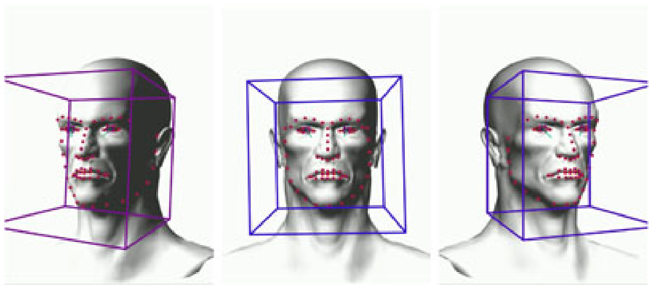

FIGURE 4 Two-dimensional (2D) facial landmark in three views from a three-dimensional (3D) head model, to find a 3D landmark for matching the 2D landmark for captured images

where w_i is the weight of each point, which we fixed the values to be 1 for all i. After the best-fit transformation is computed, the user can still make adjustments, if needed, using our software.

3.2 | Orientation field

3.2.1 | 2D orientation acquisition

We follow the previous method from the works of Paris et al.[3,5] and Chai et al.[7] in extracting a 2D orientation using a bank of Gabor filters.[28] Let the grayscale segmented image be I and the Gabor convolution kernels be

$$K_\theta(u,v) = e^{(-0.5[a^2+b^2])}\cos(2\pi u\lambda^{-1}), \tag{2}$$

where $a = \sigma_u^{-1}(u\cos(\theta) + v\sin(\theta))$ and $b = \sigma_v^{-1}(v\cos(\theta) - u\sin(\theta))$. The convolved image is then $F(x,y,\theta) = |K_\theta * I|_{(x,y)}$. In our experiments, we found that the values similar to those used in the work of Chai et al.[7] produce the best results. We use the kernel size of 9×9, $\sigma_u = 1.8$, $\sigma_v = 2.4$, and $\lambda = 4$. To make the result more robust, we add a process to perform supersampling in each pixel of the kernel by dividing each pixel into 10×10 uniform grids before filtering. After filtering, the values of the superpixels are then averaged and used. We found that this approach produces shaper directions in our examples. We filter the image using Gabor filters oriented at 32 angles spaced equally in the interval of $[0,\pi)$ and find θ that returns the maximum response value. The 2D orientation map is hence $O(x,y) = \tilde{\theta} = \arg\max_\theta(F(x,y,\theta))$. Moreover, we compute the confidence value, $W(x,y)$, for each orientation pixel[3] using

$$w(x,y) = \sum_\theta (d(\theta,\tilde{\theta}) \cdot (\Delta F(\theta,\tilde{\theta})^2)^{0.5}, \tag{3}$$

where $\Delta F(\theta,\tilde{\theta}) = F(\theta) - F(\tilde{\theta})$ and angular distance $d(\theta_1,\theta_2)$ between θ_1 and θ_2. To accentuate the dominant orientations within the 2D orientation map, we use the confidence map as an input image and repeat the above steps for three iterations, as suggested in the work of Chai et al.[7] Figure 5 shows an example of a 2D orientation map, where the hue of each pixel represents the orientation and the brightness represents the confidence. We perform the steps above on all eight images.

3.2.2 | Visual hull

We use the visual hull[29] as the rough shape of the hair volume. It can be constructed from the segmented images. Similar to the head pose estimation step, we assume that the image planes are the views taken from orthographic cameras that are from equal distance from the head and the two consecutive views are exactly 45° apart, an example of which is shown in Figure 6. For each voxel, we perform orthogonal backprojection on each of the image planes to see if the voxel lies within the hair region in that view. If a voxel lies in the hair in more than three views, we mark it as in the visual hull. As the images can be noisy, the visual hull generated may contain holes and floating islands. We remove them by performing two iterations of erosion followed by two iterations of dilation on the voxel grid. In our examples, we use a grid resolution of $90 \times 67 \times 67$ to process the visual hull. The orthography, the equal distance, and the equal space assumptions clearly

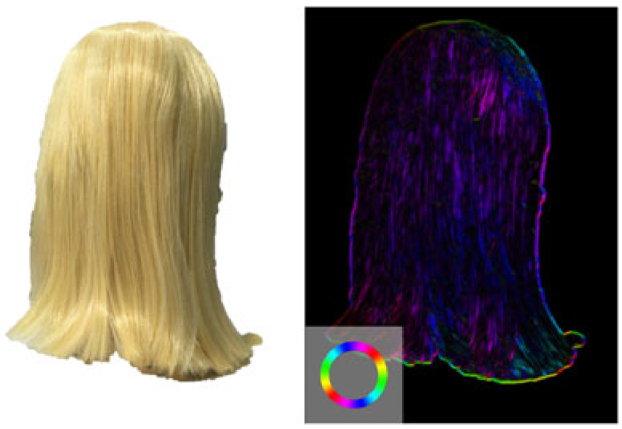

FIGURE 5 Left: segmented image from the right-back view of a wig. Right: two-dimensional orientation map after doing two iterations of iterative refinement. The direction and the confidence value are visualized as hue and brightness, respectively

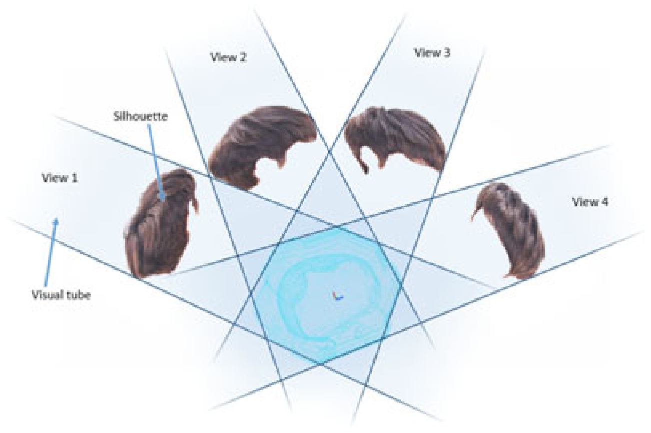

FIGURE 6 Two-dimensional orientation field from four of the eight views. We assume that the photos taken are orthogonal projections. Visual hull filled voxels are also visualized in the middle

do not hold in practice as the mobile phone camera has a relatively short focal length and the user can never position the camera exactly at the assumed poses. However, we found that the visual hull created is of sufficient quality. The remaining steps of our algorithm are also robust against these inaccuracies and can reconstruct guide hairs from various hairstyles.

3.2.3 | 3D orientation field

After obtaining the 2D orientation maps of the eight views, we next compute the 3D orientation of the outer filled voxels by following the previous method from the work of Wei et al.[4] with some modifications. A filled voxel is considered outer if it is adjacent to an empty voxel. To compute the 3D orientation of a given voxel, we first determine the visibility of the 3D position from each of the orthogonal views by checking if it is within the hair region of that view and that the ray from the center of the grid toward that voxel intersects with the image plane of that view or not. If there is only one visible view, we ignore that voxel. If there are two visible views with the image plane normals n_1 and n_2 and 2D directions (expressed as 3D vectors) v_1 and v_2, the 3D orientation of that voxel, D, can be computed using $D = N_1 \times N_2$, where $N_i = n_i \times v_i$. If there are three or more visible views, we can compute D by minimizing

$$\sum_j \sigma_j^2 (N_j \cdot D)^2, \tag{4}$$

subject to $\|D\|_2 = 1$, where σ_j is the uncertainty computed as one over the 5×5 box average of $w(x, y)$ around that pixel. The solution for D is the eigenvector corresponding to the smallest eigenvalue of $\sum_j \sigma_j^2 (N_j^T N_j)$. Note that for the purpose of strand tracking, D and $-D$ are considered the same; hence, we refer to it as orientation, as opposed to direction.

3.2.4 | Orientation filling in hidden hair volume

We now have the 3D orientations of most of the outer voxels, which correspond to the exterior of the hair. We then use the 3D pose of the head computed to translate, rotate, and scale the growth mesh, which is the 3D mesh whose vertices are the root of the guide hair strands. We next identify the voxels that intersect with the growth mesh and set the 3D orientation of each such voxel to the average normal of the triangles intersecting the voxel. We are now ready to extrapolate the orientation to other voxels. First, we convert the 3D orientation, D, at each known cell into a 3×3 tensor, DD^T. We then perform an isotropic diffusion of the tensor on all other voxels using the method from Paris et al.,[5] where we use the known voxels as Dirichlet boundary conditions. We then convert each tensor back to a 3D orientation by finding the eigenvector corresponding to the smallest eigenvalue of the tensor.

3.3 | Guide hair strand generation

3.3.1 | Strand tracking

We trace along the 3D orientation field using a method suggested by Paris et al.[5] enhanced with several modifications inspired by the works of Chai et al.[7,8] Note that we only perform tracking to generate guide hair strands, not to generate each individual hair strand. Several hundred guide hair strands are sufficient to represent most hairstyles. During rendering, these guide hair strands are used for interpolating a lot more rendered hair using a tessellation unit in the graphics processing unit (GPU). To generate a guide hair strand, we start the tracking process at a root position p_0, which is the position of a growth mesh's vertex. We refer to the 3D orientation evaluated at position p_i as dp_i. We heuristically set d_{-1} to be equal to the bent vertex normal, which is computed as the vertex normal, n, rotated by G degrees around the $n \times [0, -1, 0]^T$ axis. The user can choose G in our program where a smaller value is used for short hairstyles and a larger value should be used for long hairstyles. In our examples, values between 20° and 60° are being used. d_{-1} is an estimation of the direction that the hair strand comes out of the growth mesh. We first set the tracking state to "certain" and set the score to a maximum score; then, for every step i, starting from $i = 0$, we track each strand using the following rules.

1. If p_i is outside the visual hull, change the current state to "uncertain."
2. If $dp_{i-1} \cdot dp_i < 0$, then $dp_i = -dp_i$.
3. If $\arccos(dp_{i-1} \cdot dp_i) > \theta_{\max}$, change the current state to "uncertain."
4. If $\arccos(dp_{i-1} \cdot dp_i) \leq \theta_{\max}$, change the current state to "certain."
5. If the current status is "certain," set the score to a maximum value and set $p_{i+1} = p_i + \delta \frac{dp_i}{|dp_i|}$.
6. If the current status is "uncertain," set $p_{i+1} = p_i + \delta \frac{dp_{i-1}}{|dp_{i-1}|}$.
7. Subtract the score by 1, and if the score is 0, then break.

In our examples, we use maximum score = 4, $\delta = 0.75$ of a voxel width, and $\theta_{\max} = 50°$. The guide hair strands obtained from our tracking algorithm can still be jagged due to the inaccuracy of the 3D orientation estimation. We smooth them out using the smoothing method suggested in the work of Iben et al.[30] for five iterations, resulting in sufficiently smooth curves. We perform the tracking algorithm described for each of the growth mesh vertices to obtain all the guide hair strands.

3.3.2 | Trimming erroneous strand

Occasionally, the strand tracking algorithm described above produces erroneous results as it tracks down a strand and incorrectly tracks back up along another strand. This results in a U-shape strand. In general, a hair strand tends to curve downward due to the Earth's gravity. Although some hairstyles have an S-shape where the tip rises up, no hairstyle commonly found has a U-shape. We propose a heuristic to check if the strand should be trimmed or not. Let $\bar{p}_1, \bar{p}_2, \dots, \bar{p}_n$, be the smooth position of the vertices along a strand. Let i be the smallest index where $\bar{p}_i.y - \bar{p}_{i-1}.y < 0$. We count the number of vertices $k > i$ where $\bar{p}_k.y - \bar{p}_{k-1}.y > 0$. If the number of such vertices is greater than $\mu(n - i - 1)$, we trim the hair strand at vertex j, where j is the smallest index greater than i such that $\bar{p}_{j+1}.y - \bar{p}_j.y > 0$. The counting procedure is aimed at detecting if the hair strand unnaturally forms a U-shape. In all our examples, we use $\mu = 0.3$.

3.4 | Exporting to HairWorks

NVIDIA HairWorks simulates and renders hair in real time with a GPU. As input, HairWorks needs a growth mesh and guide hair strands. The root of each guide hair strand is a vertex of the growth mesh. Rendering hair strands are seeded randomly within triangles of the growth mesh. Each rendering hair strand is interpolated from the three guide hair strands whose roots correspond to the vertices of the triangle. The rendering hair strands are created dynamically during runtime using a GPU tessellation shader.

HairWorks requires that all the guide strands have the same number of vertices, m. Therefore, we resample each tracked strand curve into $m - 1$ equal-length edges. We ignore the effect of gravity on the rest pose of our guide hair strands, as typically done in games. We write the guide hair strands into an .apx file and create an .fbx file for the growth mesh model. As shown in Figure 7, they are imported to the HairWorks Viewer Tool. HairWorks has many parameters to be adjusted both for simulation and rendering, such as animation speed (time step), constraints controller, external force (wind and gravity), waviness, density, and so on, for which we use the values similar to the example scenes provided with HairWorks.

3.5 | Speed optimization

We optimize our method to gain speed over the experiment we did for Vanakittistien et al.[2] In that work, our processing time was about 30–40 min on our PC laptop (CPU Intel Core i7-4700HQ and GPU Nvidia GTX 860m). With our new

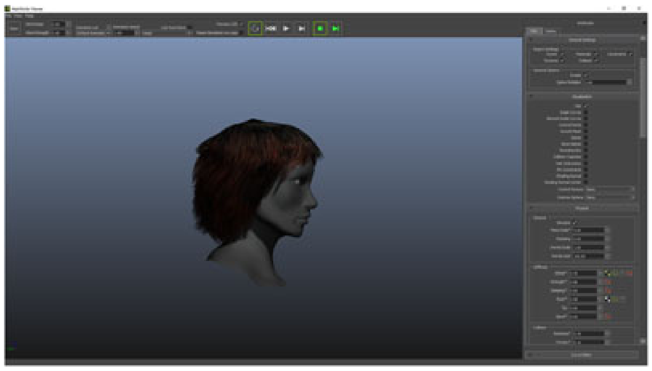

FIGURE 7 HairWorks Viewer Interface allows the user to adjust various simulation and rendering parameters. We import the guide hair strands generated with our method to this program

optimization presented in this manuscript, the processing time from the image editing process to finishing exporting to the HairWorks file is about 30 s after the user finishes the last segmentation. To summarize, we have proposed the following optimizations.

1. Use the background threads for 2D orientation extraction to overlap this time with the user segmentation of the next image and 3D pose estimation.
2. Remove memory allocation in loops and improve cache coherency for the 2D orientation and confidence map extraction step, which reduces the time from 10 min to 10 s per image.
3. Use bounding volume hierarchy to accelerate ray casting when checking if a grid point is in the growth mesh or not.
4. Remove memory allocation in loops and improve cache coherency for the 3D orientation field construction step, which reduces the time from 8 to 15 min to under 20 s.

4 | RESULT

We collected the data from a number of volunteers. All data presented in this paper are either taken with our smartphone (iPhone 6s in our experiment) or captured from still frames of a YouTube video. The results from our algorithm were simulated and rendered in HairWorks in real time. Visually, they closely match their real-world counterparts. We also tested our system with images of a wig on a wicker head, as shown in Figure 8. More results can be found in our accompanying

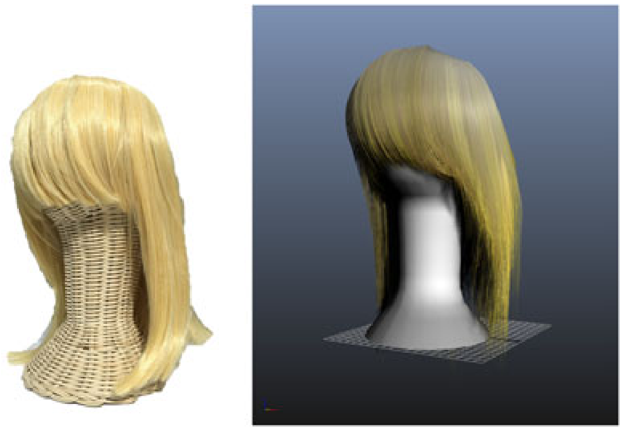

FIGURE 8 Left: the wicker head with a wig. Right: our result, simulated and rendered in real time in the HairWorks Viewer from a similar viewpoint

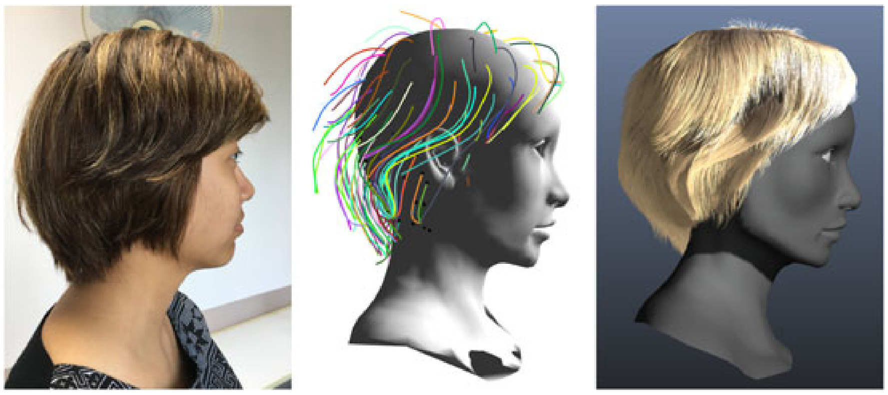

FIGURE 9 Left: input image of a medium-length hairstyle. Center: reconstructed guide hair strands. Right: our result, simulated and rendered in real time in the HairWorks Viewer from a similar viewpoint

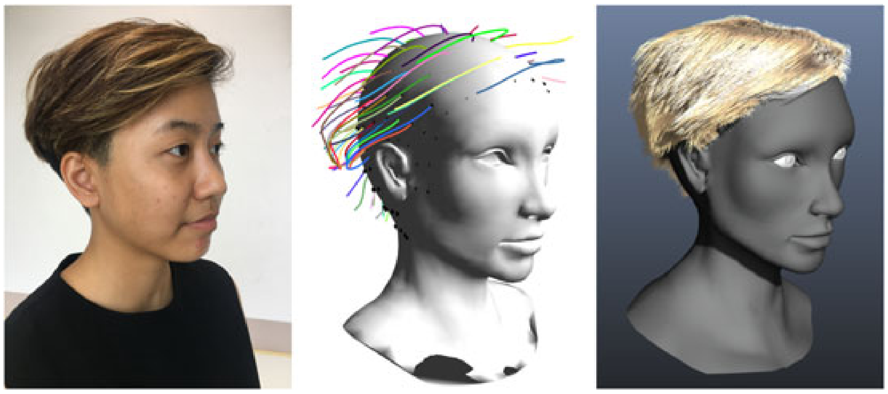

FIGURE 10 Input image of a short hairstyle, reconstructed guide hair strands, and our result

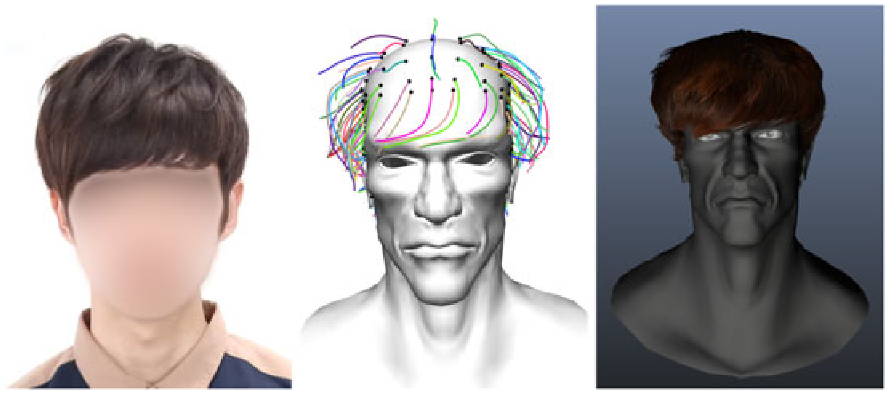

FIGURE 11 Input image of a hairstyle from still frames of a YouTube video, reconstructed guide hair strands, and our result

video. The final hair strands from all examples were simulated and rendered in real time. We have provided our implementation of the algorithm presented in this paper along with the sample images at https://isl2-dev.cp.eng.chula.ac.th/~dae/bank/gr3dhm/project_page/index.html.

Figures 9 and 10 show images of a woman's hairstyles. The first one is a medium-length hairstyle, and the other image is a short hairstyle. Figure 11 shows a hairstyle reconstructed from still frames of a YouTube video of a man recording his hairstyle in 360°. We selected eight frames from the video that roughly matched the eight views required and used them as input to our system.

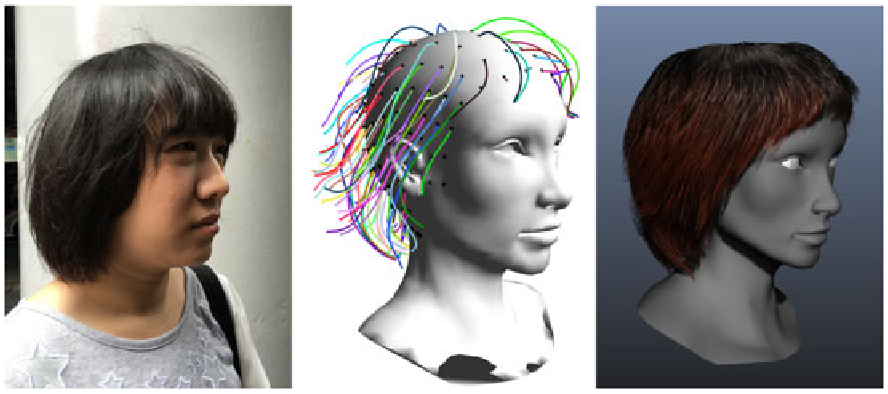

FIGURE 12 Input images of a long hairstyle, reconstructed guide hair strands, and our result

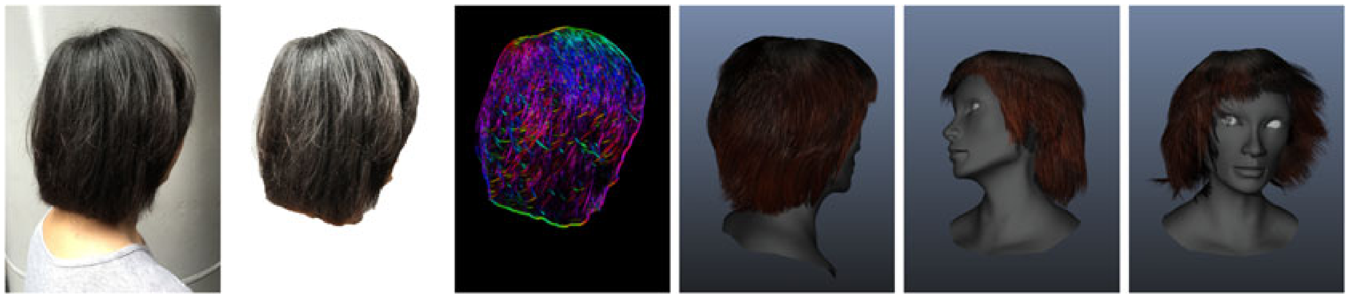

FIGURE 13 We segment out the hair region within an image and construct a two-dimensional orientation field. Eight such fields are combined to generate a three-dimensional orientation field. Guide hair strands are then tracked, processed, and imported to HairWorks as the guide hairs for real-time simulation and rendering

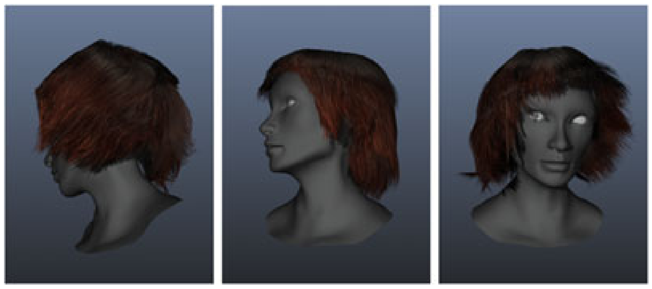

FIGURE 14 Snapshots of the animation of hair whose guide strands are captured with our method

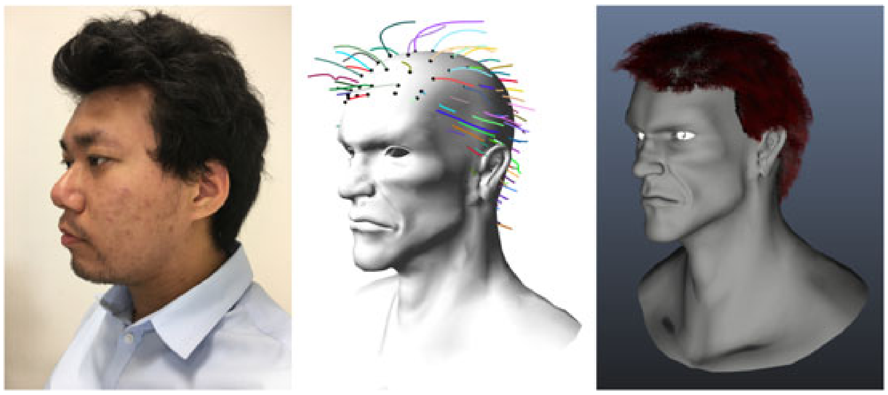

FIGURE 15 Input images of a man's hairstyle with curly short hair, reconstructed guide hair strands, and our result

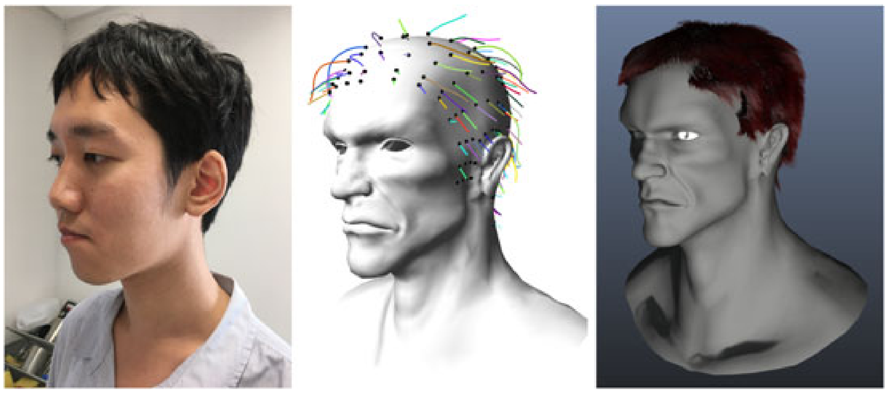
FIGURE 16 Input images of a man's short hairstyle, reconstructed guide hair strands, and our result

TABLE 1 Parameters of reconstruction and strand details

Model	Figure 11	Figure 12	Figure 13	Figure 14	Figure 15	Figure 16
Resolution	540 × 720	540 × 720	722 × 720	540 × 720	540 × 720	540 × 720
Guided strands	161	123	183	163	166	176
Strand particles	4,889	1,913	3,599	4,673	1,814	1,809
Exported strands	176	176	209	176	209	209
Exported particles	1,408	1,408	1,672	1,408	1,672	1,672
Strand curves	28,364	20,889	28,860	28,979	23,478	27,468
Root angle (degrees)	45	37	50	49	20	45

TABLE 2 Processing time of each step in seconds

	Model	Figure 11	Figure 12	Figure 13	Figure 14	Figure 15	Figure 16
2D orientation field processing step	Front	9.9	7.1	12.6	11.2	8.2	6.8
	Front Left	12.6	8.0	14.2	13.7	10.9	9.2
	Side Left	15.5	11.6	15.1	17.5	13.6	11.8
	Back Left	18.0	13.3	16.6	17.4	12.5	12.6
	Back	19.2	13.1	16.7	17.7	13.6	12.5
	Back Right	17.4	13.5	17.5	17.2	14.3	13.3
	Side Right	16.7	12.0	15.6	15.8	12.0	11.2
	Front Right	11.9	8.8	14.6	12.0	8.4	7.8
3D orientation field creation step		13.0	6.6	25.5	15.2	10.3	5.2
Tracking step		10.1	5.9	12.3	12.7	5.8	4.4

Figure 12 shows a long hairstyle, the hairstyle's result of which can be seen in Figures 7, 13 and 14. Figure 15 shows a man's hairstyle with curly short hair. Figure 16 shows a man's short hairstyle. Reconstructed guide hair strands are also included in these Figures, for comparison with the final results. In most examples, the user only needed to make small adjustments to the 3D pose found by our automatic pose estimation. However, in few of the examples, the 2D facial landmarks were not accurate and, hence, required the user to make moderate adjustments to the pose. The rendering parameters like hair color, curliness, etc., can be chosen by the user in HairWorks to obtain the desired looks.

Table 1 shows the resolution used, the number of strands and particles in different computational steps, and the bent angle from the root normal used. The timing of each computation step in seconds is shown in Table 2.

Limitations: We can create guide hair strands suitable for real-time simulation and rendering from various hairstyles. Unfortunately, some complex hairstyles, such as a fizzy hairstyle and a highly curly hairstyle, might not be tracked accurately using our method. Handling such styles is an interesting area for future work.

5 | CONCLUSION

With our system, users can put their hairstyles into games with ease. A user can acquire the hairstyle data by using a smartphone camera and our mobile app to take photos of eight views around his/her head. The images should then be segmented and used as input into our algorithm, which will generate 2D orientation fields, pose estimation, visual hull, and 3D orientation field and finally track the guide hair strands. The final guide hair strands are then processed and exported to HairWorks. Our future work includes utilizing a fully automatic background removal to replace the manual process and improving the tracking algorithm to be able to handle more complex hairstyles.

ACKNOWLEDGEMENTS

We would like to thank the volunteers for helping us collect the data and the anonymous reviewers for their helpful comments and suggestions. We also thank the Development and Promotion of Science and Technology Talents Project for supporting this research.

ORCID

Nuttapon Vanakittistien http://orcid.org/0000-0003-3510-7257

REFERENCES

1. NVIDIA. Nvidia HairWorks; 2015.
2. Vanakittistien N, Sudsang A, Chentanez N. 3D hair model from small set of images. Proceedings of the 9th International Conference on Motion in Games (MIG'16); 2016; Burlingame, CA. New York, NY: Association for Computing Machinery; 2016. pp. 85–90.
3. Paris S, Briceño HM, Sillion FX. Capture of hair geometry from multiple images. ACM Trans Graph. 2004;23(3):712–719.
4. Wei Y, Ofek Y, Quan L, Shum H-Y. Modeling hair from multiple views. ACM Trans Graph. 2005;24(3):816–820.
5. Paris S, Chang W, Kozhushnyan OI, et al. Hair photobooth: geometric and photometric acquisition of real hairstyles. ACM Trans Graph. 2008;27(3).
6. Jakob W, Moon JT, Marschner S. Capturing hair assemblies fiber by fiber. ACM Trans Graph. 2009;28(5).
7. Chai M, Wang L, Weng Y, Yu Y, Guo B, Zhou K. Single-view hair modeling for portrait manipulation. ACM Trans Graph. 2012;31(4).
8. Chai M, Wang L, Weng Y, Jin X, Zhou K. Dynamic hair manipulation in images and videos. ACM Trans Graph. 2013;32(4).
9. Luo L, Li H, Rusinkiewicz S. Structure-aware hair capture. ACM Trans Graph. 2013;32(4).
10. Hu L, Ma C, Luo L, Li H. Robust hair capture using simulated examples. ACM Trans Graph. 2014;33(4).
11. Hu L, Ma C, Luo L, Wei L-Y, Li H. Capturing braided hairstyles. ACM Trans Graph. 2014;33(6).
12. Hu L, Ma C, Luo L, Li H. Single-view hair modeling using a hairstyle database. ACM Trans Graph. 2015;34(4).
13. Chai M, Luo L, Sunkavalli K, Carr N, Hadap S, Zhou K. High-quality hair modeling from a single portrait photo. ACM Trans Graph. 2015;34(6):1–10.
14. Chai M, Shao T, Wu H, Weng Y, Zhou K. AutoHair: fully automatic hair modeling from a single image. ACM Trans Graph. 2016;35(4).
15. Zhang M, Chai M, Wu H, Yang H, Zhou K. A data-driven approach to four-view image-based hair modeling. ACM Trans Graph. 2017;36(4):1–11.
16. Müller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics. J Vis Commun Image Represent. 2007;18(2):109–118.
17. Bender J, Müller M, Otaduy MA, Teschner M. Position-based methods for the simulation of solid objects in computer graphics. Paper presented at: STAR Proceedings of Eurographics; 2013; Girona, Spain. Aire-la-Ville, Switzerland: The Eurographics Association; 2013.
18. Miles M, Müller M. Position based fluids. ACM Trans Graph. 2013;32(4).
19. Macklin M, Müller M, Chentanez N, Kim T-Y. Unified particle physics for real-time applications. ACM Trans Graph. 2014;33(4):1–12.
20. Müller M, Kim TY, Chentanez N. Fast simulation of inextensible hair and fur. Paper presented at: Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS); 2012; Darmstadt, Germany. Aire-la-Ville, Switzerland: The Eurographics Association; 2012.
21. Han D, Harada T. Real-time hair simulation with efficient hair style preservation. Paper presented at: Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS); 2012; Darmstadt, Germany. Aire-la-Ville, Switzerland: The Eurographics Association; 2012.
22. Kim T-Y, Chentanez N, Müller-Fischer M. Long range attachments—a method to simulate inextensible clothing in computer games. Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'12); 2012; Lausanne, Switzerland. Aire-la-Ville, Switzerland: The Eurographics Association; 2012. pp. 305–310.
23. Umetani N, Schmidt R, Stam J. Position-based elastic rods. Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation; 2014; Copenhagen, Denmark. Aire-la-Ville, Switzerland: The Eurographics Association; 2014. pp. 21–30.
24. Rother C, Kolmogorov V, Blake A. "GrabCut": interactive foreground extraction using iterated graph cuts. ACM Trans Graph. 2014;23(3):309–314.

25. Amos B, Bartosz L, Satyanarayanan M. OpenFace: a general-purpose face recognition library with mobile applications. Pittsburgh, PA: Carnegie Mellon School of Computer Science; 2016. Technical Report CMU-CS-16-118.
26. FFmpeg. https://www.ffmpeg.org
27. Müller M, Heidelberger B, Teschner M, Gross M. Meshless deformations based on shape matching. Proceedings of ACM SIGGRAPH 2005 Papers (SIGGRAPH'05); 2005; Los Angeles, CA. New York, NY: Association for Computing Machinery; 2005. pp. 471–478.
28. Jain AK, Farrokhnia F. Unsupervised texture segmentation using Gabor filters. Pattern Recognit. 1991;24(12):1167–1186.
29. Laurentini A. The visual hull concept for silhouette-based image understanding. IEEE Trans Pattern Anal Mach Intell. 1994;16(2):150–162.
30. Iben H, Meyer M, Petrovic L, Soares O, Anderson J, Witkin A. Artistic simulation of curly hair. Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation; 2013; Anaheim, CA. New York, NY: Association for Computing Machinery; 2013. pp. 63–71.

AUTHOR BIOGRAPHIES

Nuttapon Vanakittistien was born in Bangkok, Thailand, in 1992. He received B.Eng degree in computer engineering from Chulalongkorn University, Bangkok, Thailand, in 2014. He is currently master student in computer engineering, Chulalongkorn University, Bangkok, Thailand. His research interests include computer graphics and simulation.

Attawith Sudsang received the B.E. degree in computer engineering from Chulalongkorn University, Bangkok, Thailand, in 1991 and the M.S. and Ph.D. degrees in computer science from the University of Illinois at Urbana-Champaign in 1994 and 1999, respectively. He is currently an Assistant Professor with the Department of Computer Engineering, Chulalongkorn University. He was a Research Associate with the Department of Computer Science, Rice University, Houston, TX, from 1999 to 2000. His research interests include distributed manipulation, part fixturing, grasping, and multirobot systems.

Nuttapong Chentanez received the PhD degree in computer science from the University of California, Berkeley. His dissertation was about interactive simulation of surgical needle insertion and steering. Currently, he is a research consultant for the physics research group of NVIDIA and teaches at the Department of Computer Engineering, Chulalongkorn University. His main research interest includes liquid and deformable body simulation both for real-time and offline applications.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of the article.

How to cite this article: Vanakittistien N, Sudsang A, Chentanez N. Game-ready 3D hair model from a small set of images. *Comput Anim Virtual Worlds*. 2019;30:e1855. https://doi.org/10.1002/cav.1855