2.5D Cartoon Hair Modeling and Manipulation

Chih-Kuo Yeh, Pradeep Kumar Jayaraman, Xiaopei Liu, Chi-Wing Fu, *Member, IEEE*, and Tong-Yee Lee, *Senior Member, IEEE*

Abstract—This paper addresses a challenging single-view modeling and animation problem with cartoon images. Our goal is to model the hairs in a given cartoon image with consistent layering and occlusion, so that we can produce various visual effects from just a single image. We propose a novel 2.5D modeling approach to deal with this problem. Given an input image, we first segment the hairs of the cartoon character into regions of hair strands. Then, we apply our novel layering metric, which is derived from the Gestalt psychology, to automatically optimize the depth ordering among the hair strands. After that, we employ our hair completion method to fill the occluded part of each hair strand, and create a 2.5D model of the cartoon hair. By using this model, we can produce various visual effects, e.g., we develop a simplified fluid simulation model to produce wind blowing animations with the 2.5D hairs. To further demonstrate the applicability and versatility of our method, we compare our results with real cartoon hair animations, and also apply our model to produce a wide variety of hair manipulation effects, including hair editing and hair braiding.

Index Terms—Cartoon, still image animation, 2.5D modeling, layering, single-view modeling

1 Introduction

Single-view modeling and animation is a challenging but valuable computer graphics problem. From just a single image, it aims to reconstruct the image contents for producing appealing visual effects with very little user input. Since there is very limited information in a single image, the problem is highly challenging, but at the same time, it is also highly valuable because it requires very little data preparation effort, and yet can deliver visually-pleasing results.

Several pioneering works have been proposed to deal with this research problem on different kinds of input images. For example, Lin et al. [1] generated animated videos from a small collection of high resolution stills by temporal ordering and a second-order Markov chain model. Xu et al. [2] created believable animations of animals by analyzing the motion of animal groups in a single image, while Okabe et al. [3] animated fluids in images by extracting their motion patterns from video examples. More recently, Chai et al. [4] animated hairs in a real photo by estimating hair direction field and reconstructing the hair volume.

Inspired by the above works, this work takes cartoon/manga images as input instead of real world photos, and aims to analyze the image contents for animating and editing the cartoon hair in an input image. This work is driven by the fact that hair animation is very common in cartoon anime. If we are able to produce a 2.5D hair model from a cartoon image, we can effectively produce a wide variety of intriguing visual effects such as hair animation and editing. For example, we can present a cartoon character with hair blown by the wind, and manipulate his/her hair with proper layering.

However, such a modeling problem is technically very challenging: since the given cartoon image is just 2D, while animating/editing the hairs requires at least a 2.5D understanding of the image contents. In particular, we need to layer the hair strands and simulate their movement to create visually-plausible animations, while attempting to retain the artist's expressive style in the original drawing. This is very different from traditional methods, where artists manually draw the keyframes.

In this work, we take a 2.5D approach since 2.5D methods are often employed in cartoon modeling and animation. For example, Rivers et al. [5] and Yeh et al. [6] showed the simplicity and effectiveness of taking 2.5D approaches to rapidly model and manipulate cartoon characters. Moreover, the general public is used to the 2D visual style of cartoons, and 2.5D approaches can help retain the artist's drawing style. Furthermore, 2.5D hair models are sufficient for our problem since we take 2D cartoon images as inputs, and by reconstructing a 2.5D hair model, various cartoon hair animation and manipulation effects can be achieved without tediously reconstructing a 3D hair model.

In summary, we propose a novel three-fold solution to handle the problem: (i and ii) *new layering* and *completion methods* to construct a 2.5D hair model from a still cartoon image; and (iii) a *deformation strategy* to animate 2.5D hairs based on fluid simulations. Our layering method relies on a simple and effective metric derived from the Gestalt psychology to automatically identify and optimize the layering among the hair strands. Moreover, hair layers from 2D segmentations could be incomplete due to occlusion; our layer completion method can automatically resolve this issue, and avoids holes in the hair animation. Then, we generate skeletons of hair strands, and devise a simplified 2D fluid simulation model to produce wind-blown hair animations. To evaluate the quality of our results, we conduct a comparative study to examine hair animations produced by our methods and conventional cartoon anime. Lastly, to show

- *C.K. Yeh and T.Y. Lee are with the Department of Computer Science and Information Engineering, National Cheng-Kung University, Taiwan, Republic of China. E-mail: simpson.ycg@gmail.com; tonylee@mail.ncku.edu.tw.*
- *P.K. Jayaraman, X. Liu and C.W. Fu are with the School of Computer Engineering, Nanyang Technological University, Singapore. E-mail: pradeep.pyro@gmail.com; liuxp@ntu.edu.sg; cwfu@ntu.edu.sg.*

Manuscript received 21 Apr. 2014; revised 18 Aug. 2014; accepted 16 Sept. 2014. Date of publication 24 Sept. 2014; date of current version 30 Jan. 2015. Recommended for acceptance by J. Keyser.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2014.2360406

the applicability of our method, we demonstrate a wide variety of hair animation and manipulation effects with our 2.5D hair models.

2 Related Work

Note that there has been a large volume of research work on hair modeling; due to space limit in the paper, we cannot provide a complete overview of them.

Realistic hair modeling. There are three major approaches to model realistic hair: geometry-based, physically-based, and image-based. Geometry-based methods model the hair by using curves and surfaces. The trigonal prism wisp model [7], [8] clusters adjacent hair strands as continuous trigonal prisms linked by B-splines. Kim and Neumann employed a thin shell volume [9] with parametric surfaces and particles to model the hair surface and strands; they further proposed a multi-resolution hair model [10] using a hierarchy of generalized cylinders and polylines to enable global and local editing. Yuksel et al. [11] later developed an interactive hair modeling and styling tool to generate hair strands within simple polygonal meshes.

Physically-based methods attempt to simulate hair based on its physical properties. Hadap and Magnenat-Thalmann considered fluid flow around an obstacle as analogue of human hair [12]; they modeled hair-hair and hair-body interactions while considering the stiffness and inertial dynamics of individual strands [13]. Choe et al. [14] solved for the static deformation shapes of different hair styles while considering various physical factors. Selle et al. [15] modeled the physical properties of a single hair by a mass-spring model.

Image-based techniques aim to reconstruct the hair structure from single or multiple images. Paris et al. captured intricate hair geometry by using a fixed camera and varying light sources [16], as well as by a dome-like setup with multiple cameras, controllable LEDs, and video projectors [17]. Herrera et al. [18] employed simpler hardware setup and captured images with a hand-held thermal camera to generate accurate hair strands from the temperature orientation. Chai et al. [4] estimated the 3D hair direction field of human hair from single images and short video sequences, and reconstructed the hair volume for interactive editing. Luo et al. [19] employed a collection of images to reconstruct a 3D hair model by performing clustering, connection, and direction analysis to grow long hair strands.

Since we work with cartoon-style hair, we do not adopt these methods. Instead, we propose a novel single-view hair modeling method based on 2.5D modeling.

Cartoon hair modeling. Unlike realistic hair modeling, cartoon hair modeling aims to support computer-assisted cartoon production by non-photorealistic rendering of hair. Sakai et al. [20] modeled cartoon hair by using the implicit surface generated from 3D hair skeletons. Sugisaki et al. [21] applied physical simulations on a 3D mesh, and matched user-drawn keyframes to a manually-created motion database for hair motion interpolation. In contrast, Shin et al. [22] proposed a particle-based method that considered each hair strand as a sparse-hair model with multiple particles. While these works focused on accurate modeling of cartoon hair from scratch, we tackle a different problem: given the immense availability of 2D digital cartoon images, e.g., manga and anime, we aim to construct a 2.5D model of hair strands to support visually-plausible hair animation and editing. Most of the existing works are 3D methods, whereas we adopt a novel and simpler approach based on 2.5D modeling of 2D images. To the best of our knowledge, this is the first work we aware of in modeling image-based cartoon hair from 2.5D perspective, allowing us to retain the original art style while achieving rich animation effects.

2.5D cartoon modeling. 2.5D methods create complex scenes by layering 2D graphics at different depths. McCann and Pollard [23] generalized the layering concept: 2D graphics can partially occlude one another, or even with themselves by local layering. Rivers et al. [5] combined 2D vector drawings in different views to create a 2.5D model that can be rotated arbitrarily. Wiley [24] presented a 2.5D drawing system, which can handle self-intersections of layers by labeling the intersecting boundary curves. Zhang et al. [25] extracted temporally-consistent layer information from cartoon videos by propagating manually-marked layer labels among frames. Liu et al. [26] produced stereoscopic visual effect for 2D cartoon videos by employing T-junction cues to extract layer information combined with inpainting to fill the occluded pixels. Yeh et al. [6] enriched 2.5D modeling with assorted effects by considering 2D graphics with both front and back sides.

This work is similar in theme to [25], [26], but reconstructs 2.5D hair models from single-view 2D cartoon images to support cartoon hair animation and manipulation. Compared with [25], [26], which consider sequence of video frames as input, our input is only a single image. Our method can automatically infer the hair layering order from junctions and region overlaps among segmented hair layers, and complete the occluded hair strands. This cannot be achieved by existing methods, and is technically very challenging because our input does not have any depth information. Yet, from the experiments, our results are stable and consistent with human judgment, allowing us to deliver various hair animation and manipulation effects.

3 Overview

Fig. 1 overviews the major steps in our 2.5D modeling approach with a running example.

Segmentation. For the segmentation step, we first construct a 2D mesh as a coarser representation of the input image because it is computationally more efficient. To do so, we detect contours by the curve extraction method in [27] (see Fig. 2 (left)) as it produces fewer and nicer connected contours when compared to conventional edge detection methods. First, we find all endpoints and junctions from these curves and keep them fixed. Then, we smooth the curves by Gaussian filtering, and iteratively simplify them using the Douglas-Peucker algorithm [28]. Finally, constrained Delaunay triangulation [29] is used to partition the image (see Fig. 2 (middle)).

Next, we build a graph $G = (V, E)$, where the nodes in V and edges E correspond to faces and edges of the mesh, respectively. We adopt a mesh-based graph-cut segmentation method [30] to segment out each hair strand with

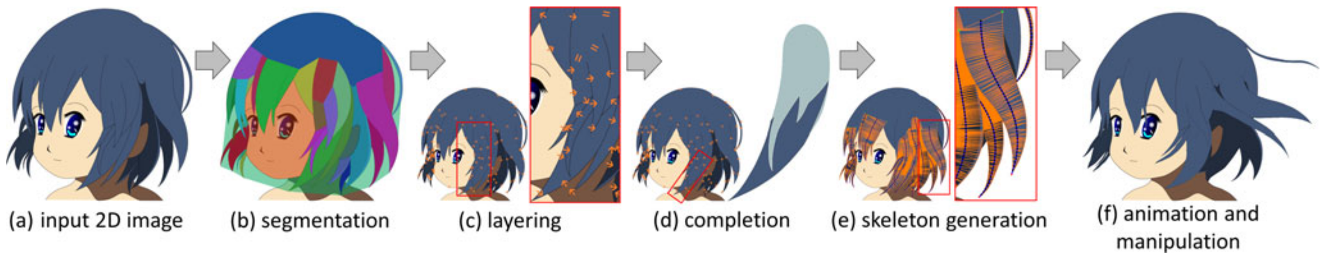


Fig. 1. Overview of our 2.5D approach. Note that in subfigure (c) above, the orange arrows show the depth ordering, i.e., pointing from occludees (lower layer) to occluders (upper layer), whereas the = signs denote same depth ordering; in subfigure (d) the light blue color represents the completed hair region.

markups (see Fig. 2 (right)). The weights of the edges $e \in E$ are set as $w(e_{ij}) = \varphi_{ij}\|e_{ij}\|$, where $(i,j) \in V$; $\|\cdot\|$ is the norm operator for computing the edge length; φ_{ij} is used to penalize the cost of cutting through edges corresponding to homogeneous regions in the image, and is set to be 0 for extracted curves, and 30 otherwise. Moreover, we employ terminal weights in the graph-cut process, and model the foreground and background by a Gaussian mixture model; our background model B is $P(C,D|B) = 1 - P(C,D|F)$, where $P(C,D|F)$ is the joint probability that measures color similarity C and point distance D to the foreground stroke F. By this, we can reduce the amount of background strokes needed for the segmentation. Note that our method can work with gradients/shading if the hair strands have clear boundaries, but still, other curve extraction or segmentation methods, e.g., [31], may also be used.

Hair modeling. After segmenting out each hair strand (see Fig. 1a and 1b), we automatically estimate the layering of hair strands from the segmented regions (see Fig. 1c) based on an analytical method we derived from psychological and experimental observations. Then, we devise a new geometric completion method based on the active contour model to fill the occluded parts in each hair strand (see Fig. 1d). Lastly, we generate the skeletons (see Fig. 1e) to enable hair animations.

Hair animation and manipulation. We may use keyframe-based methods to animate the hair skeletons, but they lack visual realism and are tedious. Hence, we devise a simplified 2D elastic model and perform a fluid simulation to animate the hair mesh, (see Fig. 1f). Other than animation, we can also perform various hair manipulation effects, e.g., hairstyle editing (see Section 7).

4 Cartoon Hair Layering

To ensure proper occlusion among hair strands when we animate and manipulate them, we next determine their depth ordering (henceforth called layering). Previous methods mainly focus on general layering problems, requiring either tedious manual specification [32] or extensive datasets for machine learning [33]. A recent method by Palou and Salembier [34] also employed T-junctions to resolve the layering, but their results show only a few number of segmented regions, and the method deals with basic T-junction cases only. In contrast, our layering method is fully automatic; it does not require extensive datasets for learning, and can handle rather complex hair layering cases with arbitrarily shaped junctions.

4.1 Junctions and Cusp Points

Junctions are corner points shared among three or more image regions. See the junctions shown in Fig. 3. They are effective visual cues [35] for determining the layering. However, not all junctions are meaningful for layering. If a junction is located on a contour edge without depth occlusion, see the zoom-in images in Fig. 3, we call it a *cusp point*, and will not use it when determining depth ordering between related image regions. Hence, before we compute the layering later in our pipeline, we first have to identify cusp points among the junctions.

The procedure to identify cusp points among junctions is as follow: If a boundary line around a hair strand is found to be on a contour line in the input cartoon image (by summing up the local Laplacian along it), we regard it as an *occluding boundary*, see again Fig. 3. Otherwise, it is *non-occluding*, and we label its end-point junctions as cusp points. Note that in this paper, we use the equal sign (=), arrow ($\rightarrow$), and bidirectional arrow ($\leftrightarrow$) to denote three possible layering cases between neighboring regions, see again Fig. 1c, say $\mathcal{A}$ and $\mathcal{B}$: $\mathcal{A} = \mathcal{B}$ means the same depth; $\mathcal{A} \rightarrow \mathcal{B}$ means $\mathcal{B}$ above $\mathcal{A}$; and $\mathcal{A} \leftrightarrow \mathcal{B}$ means an ambiguous order.

4.2 The Junction Metric Φ_J

To employ junctions for layering, we first look at the *amodal completion law* in the Gestalt psychology [35]:

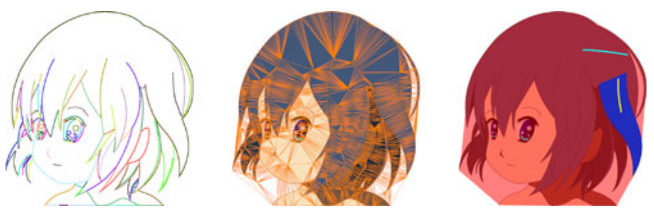


Fig. 2. Left: edge extraction; middle: constrained Delaunay triangulation; right: markups for segmenting hair strands.

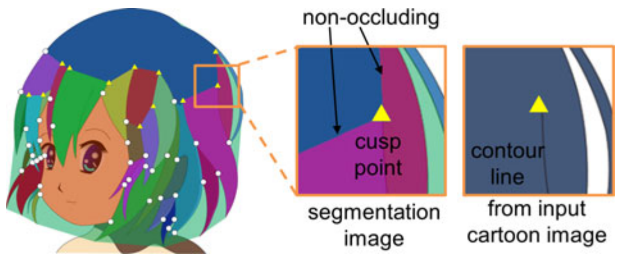


Fig. 3. Junctions (both yellow triangle and white square) and cusp points (yellow triangle).

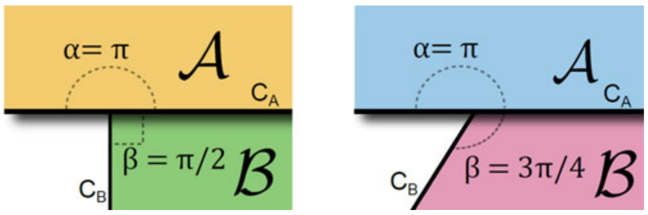


Fig. 4. Estimating the layering from junction angles: Left: an ideal case, and Right: a general case.

"When a curve, say C_B, stops on another curve, say C_A, thereby producing a T-junction, our perception tends to interpret C_B as the boundary of an object being occluded by the object associated with C_A."

Fig. 4 illustrates this law. As a notation, we name the regions associated with C_A and C_B as $\mathcal{A}$ and $\mathcal{B}$, respectively, and denote the angles subtended by $\mathcal{A}$ and $\mathcal{B}$ at the junction as α and β, respectively. In the ideal case, $\alpha = \pi$ and $\beta = \pi/2$; and we tend to interpret $\mathcal{B}$ as the *occludee* and $\mathcal{A}$ the *occluder*.

In the general case, C_A and C_B are arbitrary curves, see Fig. 4 (right), so they may not intersect at a right angle to give a layering perception [35], see Property 1:

Property 1. *If α is close to π and β is close to $\pi/2$, we regard $\mathcal{A}$ as on top of $\mathcal{B}$, i.e., $\mathcal{B} \rightarrow \mathcal{A}$.*

Moreover, based on observation with cartoon hairs, see Fig. 5, we develop three additional properties for layering. To begin, we first define the concept of *angles domain* for junction angles α and β. Since they are located at a common 2D junction, their sum should not exceed 2π. Moreover, without loss of generality, we assume α to be a larger angle, i.e., $\alpha \geq \beta$. Hence, the domain of α and β is basically the lower quarter of $[0, 2\pi] \times [0, 2\pi]$, which is the green area in Fig. 6a.

Property 2. *If $\alpha \approx \beta$, the junction metric should not suggest any ordering preference, i.e., $\mathcal{A} \leftrightarrow \mathcal{B}$.*

This is based on the fact that when α and β have similar sizes, see Fig. 5 (example 1), we should not arbitrarily decide a layering between their related image regions solely by α and β, see Fig. 6c.

Property 3. *If $\alpha+\beta \approx \pi$, the junction metric should not suggest any ordering preference, i.e., $\mathcal{A} \leftrightarrow \mathcal{B}$.*

This relates to a cartoon-hair situation, where multiple hair strands go below another hair strand altogether, see Fig. 5 (example 2). In this case, $\alpha+\beta \approx \pi$, and we again should not arbitrarily decide a layering between the two related regions solely by α and β, see Fig. 6d.

Property 4. *If β is close to 0 while α is not close to 0 or π, we regard $\mathcal{B}$ as on top of $\mathcal{A}$, i.e., $\mathcal{A} \rightarrow \mathcal{B}$.*

This property is related to another common situation with cartoon hairs (particularly due to image rasterization), where a hair tip lands on the edge of another hair strand, see Fig. 5 (example 3). Concerning this, when we compare α and β, we regard the region with a tiny sharp angle to be on the top because hair regions with sharp angles are very likely to be hair tips, i.e., β, which is the smaller angle, is close to zero. However, since we have to avoid the situations related to Properties 2 and 3, α should not be close to zero or π at the same time, see the blue areas in Fig. 6e.

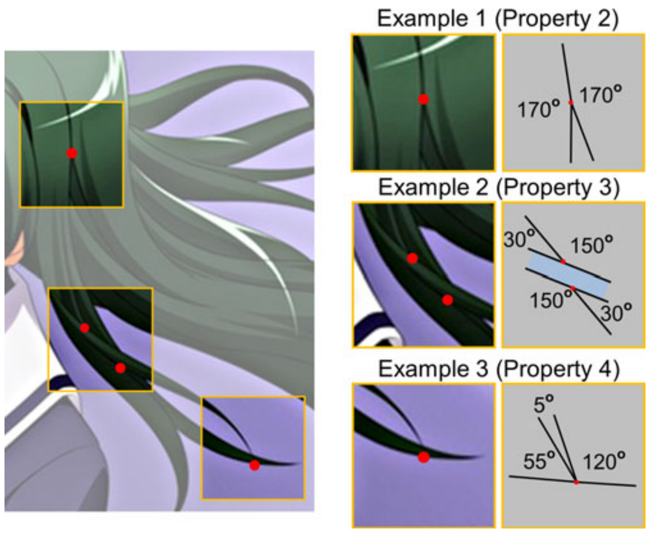


Fig. 5. Example Cartoon Hairs for Properties 2, 3 and 4.

To capture the above layering properties, we design and formulate the following junction metric Φ_J:

$$\Phi_J(\alpha, \beta) = \left|(\alpha \bmod \pi) - \pi/2\right| - \left|(\beta \bmod \pi) - \pi/2\right|,$$

where mod is the modulus operator; the sign of Φ_J indicates the layering order between $\mathcal{A}$ and $\mathcal{B}$ while its magnitude indicates the tendency: if $\Phi_J \approx 0$, $\mathcal{A} \leftrightarrow \mathcal{B}$; if $\Phi_J > 0$, $\mathcal{B} \rightarrow \mathcal{A}$, and vice versa. Note that when $\alpha \approx \pi$ and $\beta \approx \pi/2$, Φ_J is relatively large and positive, indicating $\mathcal{A}$ as on top, and when $\alpha \approx \pi/2$ and $\beta \approx \pi$, Φ_J is relatively large and negative, indicating the opposite situation. Moreover, Φ_J also fulfills properties 3 and 4, which are related to cartoon hairs. Note also that to avoid rasterization noise when computing junction angles, we fit a short boundary curve from each junction point around the segmented region.

4.3 The Region Overlap Metric Φ_R

Another local feature we employed for computing the layering is *region overlap*, which is also motivated by the amodal completion law. Given regions $\mathcal{A}$ and $\mathcal{B}$ with junction points p_1 and p_2 (see Fig. 8), we first construct a straight line from p_1 to p_2, say L_{AB}, and then, define the region overlap metric as

$$\Phi_R(\mathcal{A}, \mathcal{B}) \;=\; \delta_{\mathcal{A}} - \delta_{\mathcal{B}} \;,$$

where $\delta_{\mathcal{A}}$ and $\delta_{\mathcal{B}}$ are the area of $\mathcal{A}$'s and $\mathcal{B}$'s portion on the right and left of L_{AB}, respectively. For the example shown in Fig. 8, $\delta_{\mathcal{A}}$ is the shaded portion of $\mathcal{A}$ to the right of L_{AB} while $\delta_{\mathcal{B}}$ is zero. Hence, Φ_R is positive, suggesting $\mathcal{A}$ as on

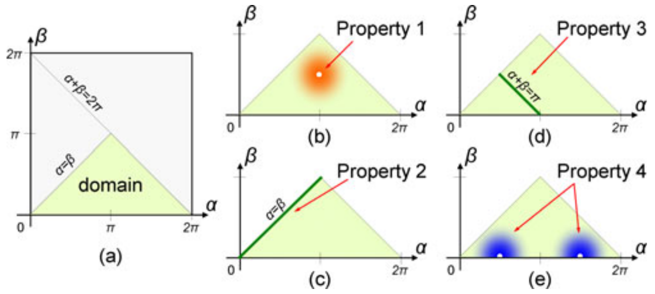


Fig. 6. (a) Domain of junction angles α and β; (b-e) Properties that define the junction metric for layering.

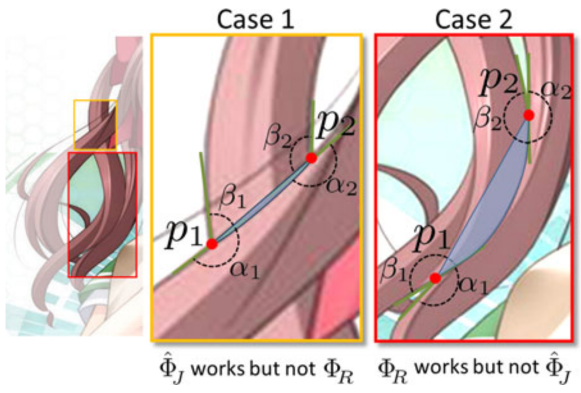

Case 1
Case 2
p_1
p_2
α_1
α_2
β_1
β_2
$\hat{\Phi}_J$ works but not Φ_R
Φ_R works but not $\hat{\Phi}_J$

Fig. 7. Case 1: Φ_R does not work because the estimated overlap region is too small. Case 2: Φ_J does not work due to layering Property 2. By combining Φ_J and Φ_R, they complement each other, and help solve the layering for both cases.

top of $\mathcal{B}$. In detail, we implement the computation by using half edge data structure to represent the segmented hair regions and arranging the edges in anticlockwise order. Hence, we can use the Green's theorem to compute the signed area of $\delta_{\mathcal{A}}$ or $\delta_{\mathcal{B}}$, which is Φ_R.

4.4 The Layering Metric Φ

Given neighboring regions $\mathcal{A}$ and $\mathcal{B}$, our layering metric Φ combines junction metric Φ_J and region overlap metric Φ_R to determine the layering order between them:

$$\Phi(\mathcal{A},\mathcal{B}) \;=\; w\cdot\hat{\Phi}_J(\mathcal{A},\mathcal{B}) \;+\; \Phi_R(\mathcal{A},\mathcal{B})/|L_{AB}|^2 \;, \tag{1}$$

where $\hat{\Phi}_J(\mathcal{A},\mathcal{B}) = s(p_1)\Phi_J(\alpha_1,\beta_1) + s(p_2)\Phi_J(\alpha_2,\beta_2)$ denotes the sum of Φ_Js at the two junction points between $\mathcal{A}$ and $\mathcal{B}$, and $s(p)$ is 0 if p is a cusp point, otherwise 1. This $s(p)$ helps to avoid Φ_J at cusp points. To normalize the area in Φ_R, we divide Φ_R by $|L_{AB}|^2$, and use a weighting term w (set to be 0.25) to balance the two metrics. Again, a positive Φ indicates $\mathcal{A}$ as on top of $\mathcal{B}$ similar to Φ_J and Φ_R, and vice versa. Fig. 7 shows two examples, where we can successfully determine the local layering order by combining the strengths of Φ_J and Φ_R.

4.5 Layering Optimization

Since the layering metric is local, it may not ensure global consistency. Hence, we develop the following combinatorial optimization method to address this issue. First, we calculate the layering order between all pairs of neighboring regions using Eq. (1). Then, we construct an undirected graph $G' = (V', E')$, where nodes in V' correspond to the shared (occluding) boundaries between hair regions, and edge e_{ij} in E connects nodes (shared boundaries) that share a common junction point in the cartoon image. Note that G' is a dual of G, see Section 3. Next, we define $x_i \in \{-1,+1\}$ as an unknown variable, which indicates two possible states {below, above} of each node, and solve for x_i by energy minimization:

$$\underset{\{x_i\}}{\operatorname{argmin}} \left[-\sum_{i\in V'} \Phi(v_i)x_i \;-\; \sum_{e_{ij}\in E'} \phi_{ij}x_i x_j \right] , \tag{2}$$

where the first term is the layering metric Φ from Eq. (1), while the second term considers edges $e_{ij} \in E'$, i.e., neighboring boundary contours in the cartoon image: $\phi_{ij} = tan(|(\theta_{ij} \bmod \pi) - \pi/2|)$, and θ_{ij} is the angle between the corresponding boundary lines at the junction. The key idea here (see Fig. 10) is: if two neighboring boundary contours are smoothly connected at a common junction, i.e., $\theta_{ij} \approx \pi$, they tend to have the same state. After the optimization, we can then obtain a local layering order between every pair of hair regions, and apply topological sort to further compute the rendering order among them.

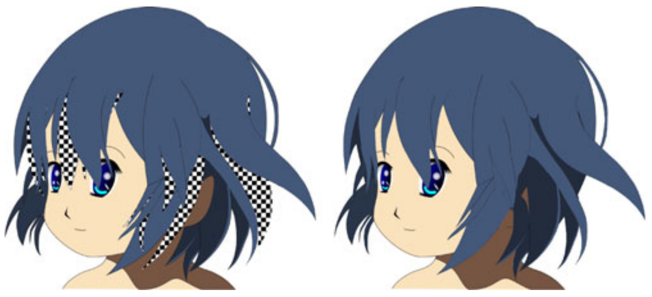


Fig. 9. 2.5D hair completion for animation and manipulation. Left: without completion; right: with completion.

5 Cartoon Hair Completion

To avoid holes in hair animation and manipulation, we next have to complete the hair strands, see Fig. 9. Single-view image completion has always been a challenging problem because the hidden parts are unknown and could have arbitrary shapes. Since we focus on hair animation and manipulation, we do not need a general completion method to fully reconstruct the hair strands, i.e., until they reach the cartoon character's head. Rather, we develop an efficient and practical solution for completing 2.5D cartoon hair, capable of delivering various hair animation and manipulation effects.

Our method considers three cases of cartoon hair completion, see Fig. 11a, 11b and 11c: The first case happens when two hair strands intersect each other, so the one behind is divided into two disconnected regions. The second case happens when the root part of a hair strand (closer to the head) is occluded by other hair strands, whereas the third case happens when the tip part of a hair strand (far away from the head) is occluded by others.

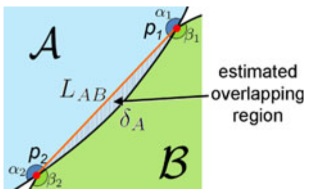


Fig. 8. Region overlap metric. By using Green's theorem, we estimate the signed area of the overlapping region.

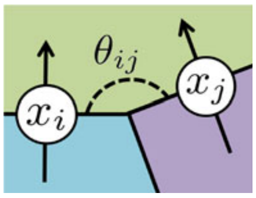


Fig. 10. Layer optimization. x_i and x_j are two neighboring layering states. They have to be in the same direction when $\theta_{ij} \approx \pi$.

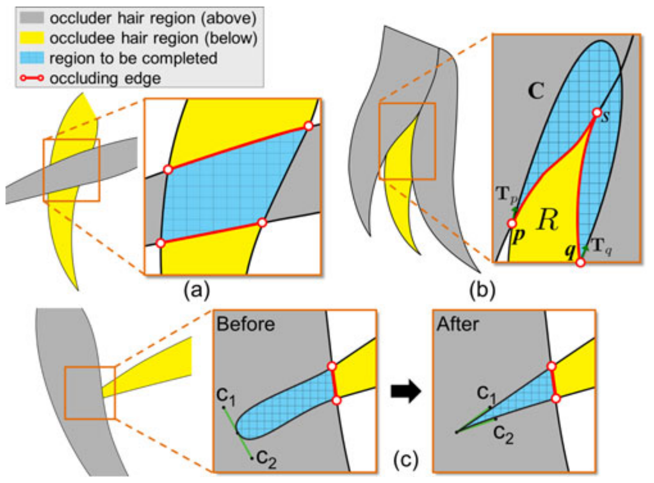


Fig. 11. Hair completion cases: (a) intersecting hairs, (b) occluded root, and (c) occluded tip before and after refinement.

5.1 Case 1: Intersecting Hairs

To identify intersecting hairs, our method starts by matching pairs of occluding edges from occludee hair regions around a common occluder, see again Fig. 11a. We consider three matching criteria: 1) color difference between of the two occludee regions around their corresponding occluding edges; 2) sum of distances between the corresponding endpoints of the two occluding edges; and 3) tangent vectors at the occluding edge endpoints.

If a match is found within a threshold, we complete the occluded area by constructing two Bézier curves with C^1 continuity to connect the corresponding endpoints of the occluding edges. Then, we merge the two divided hair regions, together with the completed area, into a single hair strand in our 2.5D model.

5.2 Case 2: Occluded Root

The second and third cases account for occluding edges that are not paired with others. Since we do not have a clear shape in the completion, the second and third cases are more complicated than the first one. Here we first try to merge (or connect) neighboring occluding edges, e.g., ps and sq in Fig. 11b, and then grow the related occludee hair strands to form the completion area.

Notations. We denote R as the occludee region to be completed, $O(R)$ as the set of occluder regions above R, p and q as the two outermost endpoints on R's (merged) occluding edge, and $\mathbf{T}_p$ and $\mathbf{T}_q$ as the related tangents at p and q, respectively. See Fig. 11b.

Our method. We devise an active-contour method based on a novel Hamiltonian function, and adopt it in a curve deformation model [36]. In short, our method iteratively refines a curve, say $\mathbf{C}_i(t)$, to form the completion area with the following constraints: $\mathbf{C}_i(0)=p$; $\mathbf{C}_i(1)=q$; $\mathbf{C}'_i(0)=\mathbf{T}_p$; $\mathbf{C}'_i(1)=-\mathbf{T}_q$; and $\mathbf{C}_i$ should be smooth and lie under $O(R)$.

To adopt the deformable model [36] for our problem, we first construct an external force field F_{ext} for pushing $\mathbf{C}_i$. To do so, we extract two short curve segments at p and q, say $\mathbf{c}_p$ and $\mathbf{c}_q$, see Fig. 12 (left & middle), and solve the following Hamiltonian function for A_p and B_p given $\mathbf{c}_p$: $H_p(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A_p \mathbf{x} + \mathbf{x}^T B_p$, which should satisfy

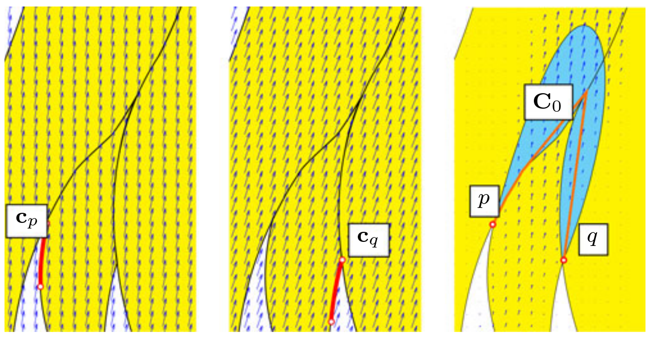


Fig. 12. Left & Middle: Vector fields (F_p & F_q) constructed from red short curve segments at p and q, respectively; Right: Combined vector field F_{ext} for pushing $\mathbf{C}_0$ (orange).

$$\dot{\mathbf{c}}_p(t) = \left(\frac{\partial H_p(\mathbf{c}_p(t))}{\partial y}, -\frac{\partial H_p(\mathbf{c}_p(t))}{\partial x}\right) \text{ and } \frac{\partial H_p(\mathbf{c}_p(t))}{\partial t} = 0 \ ,$$

where $\mathbf{x}$ is the 2D image space; A_p and B_p are unknowns solved by the above constraints. Then, we compute

$$F_p(\mathbf{x}) = \mathbf{R}(-\pi/2)\nabla H_p = \mathbf{R}(-\pi/2)(A_p\mathbf{x} + B_p) \ ,$$

which is the vector field derived from $\mathbf{c}_p$, see Fig. 12 (left). Note that we rotate ∇H_p by $\pi/2$ clockwise because ∇H_p faces outward relative to R. Similarly, we solve H_q for A_q and B_q given $\mathbf{c}_q$, and compute $F_q(\mathbf{x})$. Lastly, we combine F_p and F_q to form

$$F_{ext}(\mathbf{x}) = \Omega(\mathbf{x})(tF_p(\mathbf{x}) + (1-t)F_q(\mathbf{x})) \ ,$$

where $\Omega(\mathbf{x}) = Sigmoid(\Upsilon_p(\mathbf{x}) \cdot \Upsilon_q(\mathbf{x}))$; $\Upsilon_p(\mathbf{x}) = max(H_p(p) - H_p(\mathbf{x}), 0)$, $\Upsilon_q(\mathbf{x}) = max(H_q(\mathbf{x}) - H_q(q), 0)$ are used to clamp negative values to avoid unnatural hair growth; and $t \in [0,1]$ is an interpolation factor based on distances from p and q.

After obtaining F_{ext}, see Fig. 12 (right), we compute the convex hull of the occluding boundary of R from p to q, say $\mathbf{C}_0$, see the orange curve in Fig. 12 (right). Then, we employ Algorithm 1 with $\mathbf{C}_0$ (as an initial curve) and F_{ext} as inputs to iteratively refine $\mathbf{C}_0$ to form the completion area. Note that N is the normal of the curve; V_i is the velocity to push the curve outward; and $\mathbb{A}$ is a pentadiagonal matrix, see [36] for its detail.

Algorithm 1. ITERATIVE_REFINE ($\mathbf{C}_0, F_{ext}$)

```
1:  $i \Leftarrow 0$
2:  while true do
3:      $i \Leftarrow i + 1$
4:      for each sample point $\mathbf{C}_{i-1}(t)$ do
5:          $V_i(t) \Leftarrow ||F_{ext}(\mathbf{C}_{i-1}(t)) \cdot N(\mathbf{C}_{i-1}(t))||N(\mathbf{C}_{i-1}(t))$
6:          $\mathbf{C}_i(t) \Leftarrow (I - \mathbb{A})^{-1}[\mathbf{C}_{i-1}(t) + V_i(t)]$
7:          if $\mathbf{C}_i(t)$ outside $O(R)$ then
8:              return $\mathbf{C}_{i-1}$
9:          end if
10:     end for
11:     if $||V_i|| < \epsilon$ then
12:         return $\mathbf{C}_i$
13:     end if
14: end while
```

5.3 Case 3: Occluded Tip

The third case is similar to the second case, except that we need to achieve a sharp instead of round tip. After we form a round tip by the method in case 2, see Fig. 11c (middle), we determine the farthest point on the extrapolated contour, approximate it with a cubic Bézier curve, and then adjust the tangents at the farthest point to produce a sharp tip, see Fig. 11c (right).

5.4 Overall Procedure

To differentiate among the above three cases for each occluding edge, we start with the matching process described in case 1. If an occluding edge can be paired with others, we perform the completion procedure described in case 1. For each unpaired occluding edge, we differentiate between cases 2 and 3 by computing the average distance of the edge to the center of the head. If the edge is closer to the head as compared to the non-occluded region, we go for case 2, else case 3.

After obtaining the shape of the completion area, we employ an existing inpainting method [37] to synthesize the texture and color in the occluded region using the textures sampled from the remainder of the hair image. Some artifacts may appear if the hair occludes nontrivial image features such as eyes and mouth, where manual inpainting is required to complete the features.

6 Cartoon Hair Animation and Manipulation

To support hair animation and manipulation, we construct a skeleton for each completed hair strand as follows: First, we determine the tip of each hair strand by identifying the sharp corner(s) in each completed hair strand. Then, we look for corners that are locally above the neighboring regions by using the layering results we obtained earlier. Next, we construct a medial axis from the hair tip along the hair strand as the hair skeleton. Sometimes, a hair strand may have branches, so when we grow multiple medial axes from different tips, we join the medial axes in a hierarchical skeleton.

6.1 Cartoon Hair Animation

In general, hair simulations require modeling both the blowing wind and hair strands in 3D, and coupling 3D fluid dynamics with material deformation. This, however, could be too complicated for conventional cartoons. Hence, we propose a simplified simulation, which is efficient for generating wind-blown hairs.

Our method is based on the following assumptions. First, the movement of hair strands is approximated by the corresponding skeletons. Second, the hair dynamics is approximated by a linear elastic model with piecewise rigid deformation. Lastly, we ignore hair-hair interactions, and only employ 2D fluid simulations.

In detail, we move the hair skeleton by the aerodynamic forces from the wind, and employ the lattice Boltzmann model to solve the incompressible Navier-Stokes equation [38] with the hair skeleton as the boundary. After the simulation, we obtain pressure p, which is normal to the skeleton curve. Moreover, to prevent the skeleton from excessive bending, we model bending force $\mathbf{f} = \gamma\kappa\mathbf{n}$, where γ is the stiffness; κ is local curvature; and $\mathbf{n}$ is the normal vector along the skeleton curve. Together with the gravity force $\mathbf{g}$, the total force acting on the skeleton is: $\mathbf{F} = -p\mathbf{n} + \mathbf{f} + \mathbf{g}$.

Next, we uniformly sample the skeleton curve, and compute $\mathbf{F}$ at each sample node. Then, we perform time integration to deform and animate the skeleton according to the following dynamic equation: $\ddot{\theta} = \tau\mathbf{F}\cdot\mathbf{n}$, where θ is the change in angle between successive edges at each sample node; τ is the rate of rotation, which allows for small or large amount of hair movement. Note that we deform the skeleton by considering only rotation at sample nodes, and compute the rotations to bend the hair strand starting from its root till the tip. To improve the visual realism, we also vary the rotation rate along the hair by $\tau = \tau_0 d^\lambda$, where τ_0 is the base rotation rate and d is the distance from the hair root along the skeleton; λ is the user-controllable parameter for tuning the amount of hair bending.

Once we obtain the skeletons and their motion trajectories, we deform the mesh to follow the skeleton motion by binding each mesh vertex to one or more bones of the skeleton with appropriate blending weights per vertex. In our model, we set the weights as the inverse distance from the skeleton curve. Other standard methods, such as [39], could also be used here.

Fig. 13 presents our hair animation results. A single wind source is used to simulate the wind blowing on the cartoon hairs. From these results, we can see that our method can properly animate the hair strands with appropriate 2.5D layering while retaining the original drawing style of the artists in the given cartoon images.

6.2 Cartoon Hair Manipulation

Hair editing. Our 2.5D hair model enables flexible editing of hair strands in cartoon images, e.g., layering order, hair length, braiding, see Fig. 14. We achieve these effects by manipulating the hair skeleton and adopting the as-rigid-as-possible method [40] for the deformation. In detail, we formulate it as a minimization problem with a deformation term, a smoothness term, and a constraint term, and the objective is defined as $\Omega = w_R\Omega_R + w_H\Omega_H + w_C\Omega_C$, where w_R, w_H and w_C are the weighting coefficients, which are set to $20, 1, 1000$, respectively in all our experiments. Ω_R is a deformation term defined as $\Omega_R = \sum_{i\in\mathbf{V},k\in\mathbf{S}} w_i^k \|(\mathbf{v}_i - \mathbf{s}_k) - \mathbf{T}_k(\mathbf{v}'_i - \mathbf{s}'_k)\|^2$, where w_i^k is the weight of vertex v_i for skeleton $s_k \in \mathbf{S}$ ($\mathbf{S}$ is a set of bones) and is set to be the inverse of the distance from the skeleton; and symbol $'$ indicates variables of the deformed skeleton. $\mathbf{T}_k$ is the transformation matrix that constrains the resizing of hair along the tangential direction of the skeleton to maintain the overall shape; it is defined as a rigid transformation:

$$\mathbf{T}_k = \mathbf{R}_{(1,0)}^{e_{k'}} \begin{bmatrix} s & 0 \\ 0 & 1 \end{bmatrix} \mathbf{R}_{e_k}^{(1,0)} ,$$

where $e_k = s_{k2} - s_{k1}$ and e'_k denotes the deformed edge vector; $\mathbf{R}_v^{v'}$ is the rotation matrix that transforms vertex v to v'; and $s = \|e_{k'}\|/\|e_k\|$ is a scale factor. This transformation allows (and prevents) scaling along tangential (and normal) direction of the bones. Ω_H is a smoothness term defined as $\Omega_H = \sum_{(i,j)\in\mathbf{E}} w_{ij}\|\mathbf{v}'_j - \mathbf{v}'_i\|^2$, where $w_{ij} = cot\theta_{ij} + cot\theta'_{ij}$ are discrete harmonic weights, θ_{ij} and θ'_{ij} are angles opposite to the edge (i,j) in the original mesh. Ω_C is a constraint term defined as $\Omega_C = \sum_{i\in\mathbf{C}} \|\mathbf{v}_i - \mathbf{v}'_i\|^2$, where $\mathbf{C}$ is a set of

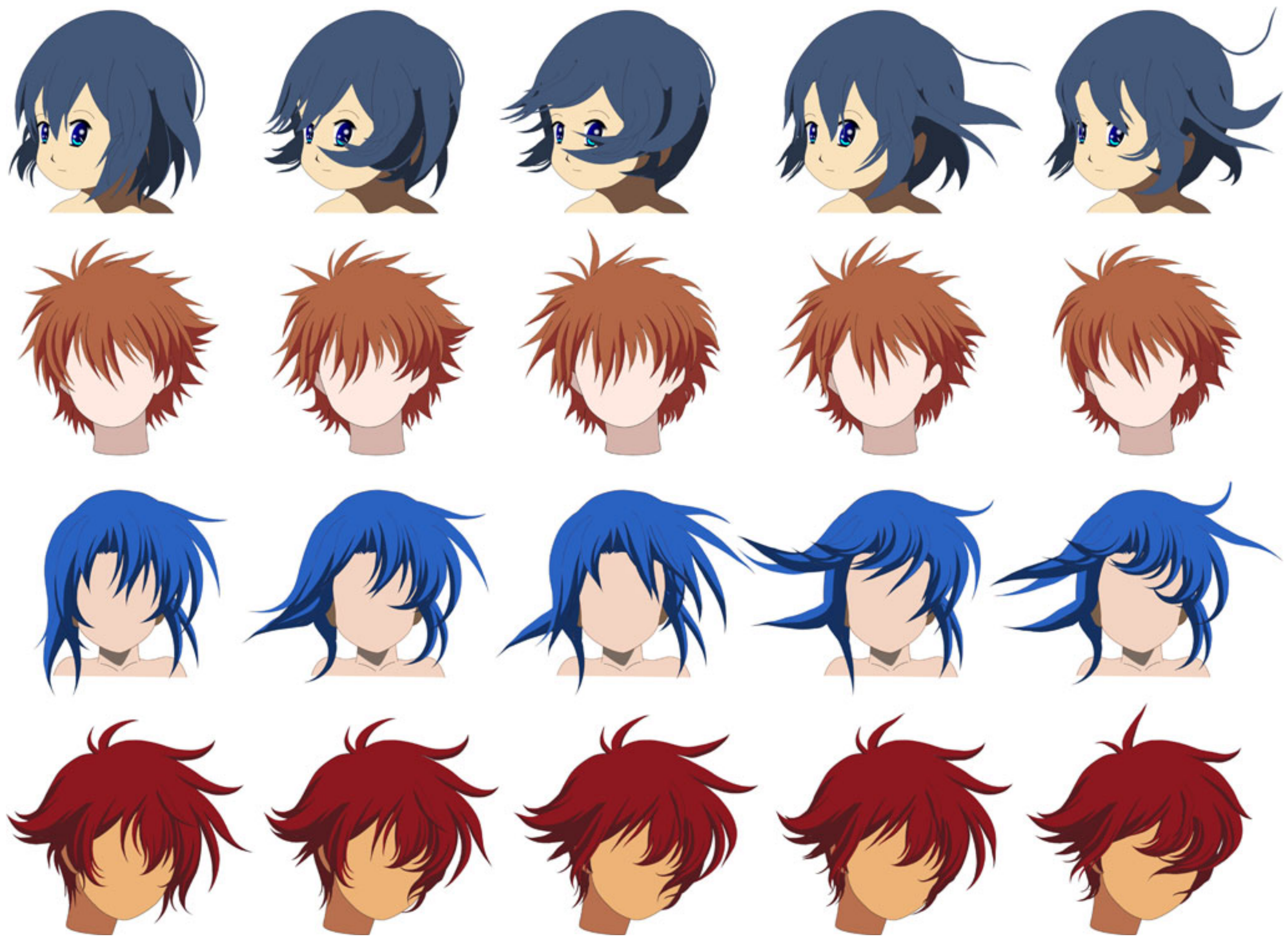


Fig. 13. Hair animation results: snapshots of 2.5D cartoon hair animations produced from our method.

constraint vertices. We set boundary constraints at the root of the hair strand.

Hair braiding. We adopt the twist operation in [6] for producing the hair braiding effect, see Fig. 14. Here we darken the texture of the hair strand as its back texture, and twist the hair strand geometry while mixing its front and back textures. Since we can only present static images in the paper, readers can refer to the supplementary video for the related animation results.

7 Discussion

Implementation and performance. The proposed system is implemented in C++ and evaluated on a desktop computer with a 3.4 GHz CPU and 4GB memory. The time taken to manually segment the hair in the four cartoon images shown in Fig. 13 (from top to bottom) are 8, 12, 9, and 15 minutes, respectively. These cartoon characters have face features, but we avoid them in the figure for clarity in the results presentation. The average processing

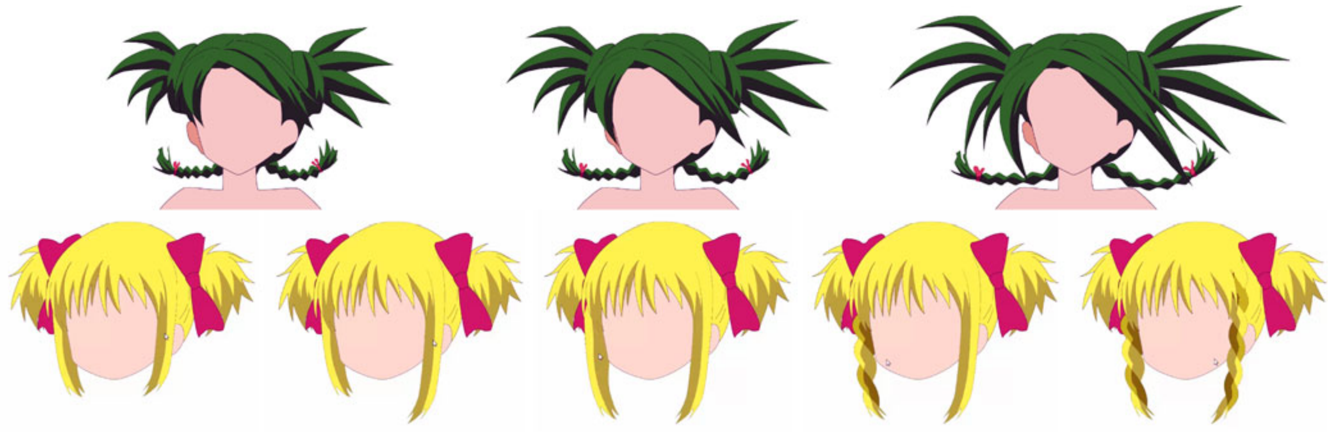


Fig. 14. Hair manipulation results: our tool can scale the hair strands to change the hairstyle (top row); and perform hair braiding by twisting the hair strands (bottom row) from single input images (leftmost).

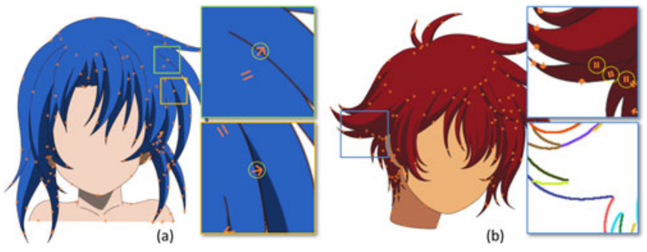


Fig. 15. Examples, where our layering metric does not match the human judgment results: (a) layering ambiguity and (b) inaccurate edge detection.

TABLE 1
Comparison: Human Subject versus our Layering Metric

cartoon images (see Fig. 13)	accuracy (vs G.T.)		average time (sec.)	
	subjects	our metric	subjects	our metric
First row	90.1%	91.4%	192.0	0.031
Second row	92.3%	93.5%	382.1	0.042
Third row	95.9%	97.0%	216.7	0.032
Fourth row	90.7%	92.3%	231.2	0.020
Average	92.3%	93.6%	255.5	0.031

time (over four segmented cartoon images) for layering computation, hair completion (excluding color and texture inpainting), skeleton generation, and deformation are around 0.031, 1.608, 0.415, and 0.015 seconds, respectively. After the 2.5D modeling, the hair animation process, which involves a fluid simulation, takes around 1.5 seconds to compute each frame.

Evaluating the layering metric. We conducted an experiment to evaluate the layering metric presented in Section 4 with seven subjects: ages from 23 to 27 (mean 24.43); four males and three females; and volunteer-based. Fig. 13 shows the four cartoon images we employed with a total of 475 layering cases, each corresponding to a pair of neighboring image regions around a cartoon hair.

In detail, we implemented a simple interface in Python to present the cartoon image to the subjects with a zoom-in view for each layering case. The subjects can enter their layering judgment by a mouse click, i.e., which region is on the top or whether the two regions have equal depth. Then, the subject can hit Spacebar to finalize a decision, and proceed to the next layering case. Our interface records the layering judgment and the time taken per case. The subjects have no time constraint in making a decision, but they mostly took a few seconds per case, except for some more complicated ones, which may require more than 10 seconds.

After collecting the data, we first compute the ground truth (G.T.) of each layering case by voting over the subject's input because different subjects may perceive different layering orders. Then, we compare the subject's inputs against the ground truth to compute the average accuracy of human subjects. Similarly, we compute the average accuracy of our layering metric by comparing our results against the ground truth. From the second and third columns in Table 1, which summarize the accuracy results, we can see that our layering metric can achieve similar or even better accuracy compared to human subjects. Moreover, our layering is automatic and simple to compute; it only takes around 0.031 seconds to compute, see the last column in Table 1.

The few failure cases, where our layering result does not match the human judgment result, are mainly due to two reasons: 1) some cases are truly ambiguous even for human judgment, see the two examples shown in Fig. 15a; and 2) our method predicts equal layering depth based on computing the Laplacian; hence, for unclear boundaries between hair strands, Cheng's curve extraction algorithm may miss the related contour line, see Fig. 15b, and so, our layering metric may mis-classify it as non-occluding boundary with equal depth.

Comparative study: Cartoon videos. We compared our hair animation results with three real cartoon videos. First, we extract a single image frame from each of these videos, and apply our method to produce a 2.5D cartoon model for each extracted image. Then, we adjust the fluid simulator to create a similar wind flow, and generate hair animations. After that, we can compare our hair animation results with the original cartoon videos. Readers can refer to the supplementary video for the related animations. The results show that our hair animations are visually comparable to the original videos even though our method takes only a single image as input.

Limitations. First, we cannot handle messy hairstyles because the hairs in this case cannot be properly segmented into strands for 2.5D modeling. Also, thin hair strands like real human hairs may not be suitable for use in our system. Second, our layering method currently cannot handle intertwined hair strands because it does not consider local layering [23], e.g., two graphical elements overlay each other with different layerings in different parts of the image. Third, our hair simulation model assumes no hair-hair interaction, i.e., the hair strands are animated independently.

8 Conclusion

This paper presents a novel 2.5D approach to modeling, animating, and manipulating hairs in a single cartoon image. In summary, we have three key contributions: First, we derive an effective layering metric from the Gestalt psychology and our observation on cartoon images; it can automatically optimize the layering order among hair strands in a single cartoon image. Second, we develop a novel layer completion method that can automatically fill the occluding parts of hair strands; by this, we can construct a 2.5D hair model to support hair animation and manipulation with appropriate layering. Lastly, we devise a simplified simulation model to animate the skeletons in hair strands, and also develop a wide variety of hair manipulation operations, including hair editing and hair braiding. To demonstrate the capability of our approach, we also compare our results with conventional cartoon videos, and apply our method to produce hair animation and manipulation results on assorted cartoon characters.

Future work. First, we plan to extend our 2.5D layering model to support local layering similar to that in [41]. Second, we would like to explore ways to produce pseudo 3D effect in the hair animation to further improve the visual quality. Third, we plan to develop cartoon hair re-shading (or relighting) method on the hair strands when we animate the hairs. Lastly, we are interested in exploring ways to extend our method to other parts of a cartoon character by studying whether or how to revise our layering method, as well as the related completion and inpainting process.

Acknowledgments

We would like to thank the reviewers for the many constructive comments that helped improve the paper. This work was supported in part by the Ministry of Science and Technology (contracts MOST-103-2221-E-006-107 and MOST-103-2221-E-006-106-MY3), Taiwan, and MOE Tier-1 (RG 29/11) and MOE Tier-2 (MOE2011-T2-2-041), Singapore.

References

[1] Z. Lin, L. Wang, Y. Wang, S. Kang, and T. Fang, "High resolution animated scenes from stills," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, no. 3, pp. 562–568, May/Jun. 2007.

[2] X. Xu, L. Wan, X. Liu, T.-T. Wong, L. Wang, and C.-S. Leung, "Animating animal motion from still," *ACM Trans. Graph. (SIGGRAPH Asia 2008)*, vol. 27, no. 5, pp. 117:1–117:8, 2008.

[3] M. Okabe, K. Anjyo, T. Igarashi, and H.-P. Seidel, "Animating pictures of fluid using video examples," *Comput. Graph. Forum*, vol. 28, no. 2, pp. 677–686, 2009.

[4] M. Chai, L. Wang, Y. Weng, X. Jin, and K. Zhou, "Dynamic hair manipulation in images and videos," *ACM Trans. Graph. (SIGGRAPH 2013)*, vol. 32, no. 4, pp. 75:1–75:8, 2013.

[5] A. Rivers, T. Igarashi, and F. Durand, "2.5D cartoon models," *ACM Trans. Graph. (SIGGRAPH 2010)*, vol. 29, pp. 59:1–59:7, 2010.

[6] C.-K. Yeh, S. Peng, L. Peng-Yen, C.-W. Fu, L. Chao-Hung, and L. Tong-Yee, "Double-sided 2.5D graphics," *IEEE Trans. Vis. Comput. Graph.*, vol. 19, no. 2, pp. 225–235, Feb. 2013.

[7] Y. Watanabe and Y. Suenaga, "A trigonal prism-based method for hair image generation," *IEEE Comput. Graph. App.*, vol. 12, no. 1, pp. 47–53, Jan. 1992.

[8] L.-H. Chen, S. Saeyor, H. Dohi, and M. Ishizuka, "A system of 3D hair style synthesis based on the wisp model." *Vis. Comput.*, vol. 15, no. 4, pp. 159–170, 1999.

[9] T.-Y. Kim and U. Neumann, "A thin shell volume for modeling human hair," in *Proc. IEEE Comp. Anim.*, 2000, pp. 104–111.

[10] ——, "Interactive multiresolution hair modeling and editing," *ACM Trans. Graph. (SIGGRAPH 2002)*, vol. 21, no. 3, pp. 620–629, 2002.

[11] C. Yuksel, S. Schaefer, and J. Keyser, "Hair meshes," *ACM Trans. Graph. (SIGGRAPH Asia 2009)*, vol. 28, no. 5, pp. 166:1–166:7, 2009.

[12] S. Hadap and N. Magnenat-Thalmann, "Interactive hair styler based on fluid flow," in *Proc. Eurographics Workshop Comput. Anim. Simul.*, 2000, pp. 87–99.

[13] ——, "Modeling dynamic hair as a continuum," *Comput. Graph. Forum*, vol. 20, no. 3, pp. 329–338, 2001.

[14] C. Byoungwon and K. Hyeong-Seok, "A statistical wisp model and pseudophysical approaches for interactive hairstyle generation," *IEEE Trans. Vis. Comput. Graph.*, vol. 11, no. 2, pp. 160–170, Mar. 2005.

[15] A. Selle, M. Lentine, and R. Fedkiw, "A mass spring model for hair simulation," *ACM Trans. Graph. (SIGGRAPH 2008)*, vol. 27, no. 3, pp. 64:1–64:11, 2008.

[16] S. Paris, H. M. Briceño, and F. X. Sillion, "Capture of hair geometry from multiple images," *ACM Trans. Graph. (SIGGRAPH 2004)*, vol. 23, no. 3, pp. 712–719, 2004.

[17] S. Paris, W. Chang, O. I. Kozhushnyan, W. Jarosz, W. Matusik, M. Zwicker, and F. Durand, "Hair photobooth: Geometric and photometric acquisition of real hairstyles," *ACM Trans. Graph. (SIGGRAPH 2008)*, vol. 27, no. 3, pp. 30:1–30:9, 2008.

[18] T. L. Herrera, A. Zinke, and A. Weber, "Lighting hair from the inside: A thermal approach to hair reconstruction," *ACM Trans. Graph. (SIGGRAPH Asia 2012)*, vol. 31, no. 6, pp. 146:1–146:9, 2012.

[19] L. Luo, H. Li, and S. Rusinkiewicz, "Structure-aware hair capture," *ACM Trans. Graph. (SIGGRAPH 2013)*, vol. 32, no. 4, pp. 76:1–76:12, 2013.

[20] T. Sakai and V. Savchenko, "Skeleton-based anime hair modeling and visualization," in *Cyberworlds (CW), Conf. Int. 2013*, pp. 318–321, Oct. 2013.

[21] E. Sugisaki, Y. Yu, K. Anjyo, and S. Morishima, "Simulation-based cartoon hair animation," in *Proc. Winter School Comput. Graph.*, 2005, pp. 117–122.

[22] J. Shin, M. Haller, and R. Mukundan, "A stylized cartoon hair renderer," in *Proc. ACM Adv. Comp. Entertainment Tech.*, 2006, pp. 64–64.

[23] J. McCann and N. Pollard, "Local layering," *ACM Trans. Graph. (SIGGRAPH 2009)*, vol. 28, no. 3, pp. 84:1–84:7, 2009.

[24] K. Wiley, "Druid: Representation of interwoven surfaces in 2 1/2d drawing," Ph.D. dissertation, Univ. New Mexico, Albuquerque, NM, USA, 2006.

[25] L. Zhang, H. Huang, and H. Fu, "EXCOL: An EXtract-and-COmplete Layering Approach to Cartoon Animation Reusing," *IEEE Trans. Vis. Comput. Graph.*, vol. 18, no. 7, pp. 1156–1169, May 2012.

[26] X. Liu, X. Mao, X. Yang, L. Zhang, and T.-T. Wong, "Stereoscopizing cel animations," *ACM Trans. Graph. (SIGGRAPH Asia 2013)*, vol. 32, no. 6, pp. 223:1–223:10, 2013.

[27] M.-M. Cheng, "Curve structure extraction for cartoon images," in *Proc. Harmonious Human Mach. Env.*, 2009, pp. 13–25.

[28] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: Int. J. Geogr. Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[29] J. R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator," in *Applied Computational Geometry*, ser. Lecture Notes in Computer Science, New York, NY, USA: Springer-Verlag, 1996, vol. 1148, pp. 203–222.

[30] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 9, pp. 1124–1137, Jul. 2004.

[31] S.-H. Zhang, T. Chen, Y.-F. Zhang, S.-M. Hu, and R. R. Martin, "Vectorizing cartoon animations," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 4, pp. 618–629, May 2009.

[32] D. Sýkora, D. Sedlacek, S. Jinchao, J. Dingliana, and S. Collins, "Adding depth to cartoons using sparse depth (in)equalities." *Comput. Graph. Forum (Eurographics 2010)*, vol. 29, no. 2, pp. 615–623, 2010.

[33] Z. Jia, A. Gallagher, Y.-J. Chang, and T. Chen, "A learning-based framework for depth ordering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 294–301.

[34] G. Palou and P. Salembier, "Monocular depth ordering using t-junctions and convexity occlusion cues," *IEEE Trans. Image Process.*, vol. 22, no. 5, pp. 1926–1939, Jan. 2013.

[35] A. Desolneux, L. Moisan, and J.-M. Morel, *From Gestalt Theory to Image Analysis: A Probabilistic Approach*, 1st ed. New York, NY, USA: Springer, 2007.

[36] C. Xu, D. L. Pham, and J. L. Prince, "Image segmentation using deformable models," *Handbook Med. Imaging*, vol. 2, pp. 129–174, 2000.

[37] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE Trans. Image Process.*, vol. 13, no. 9, pp. 1200–1212, Dec. 2004.

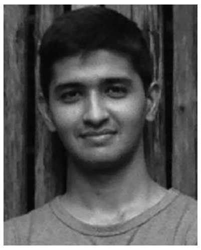
[38] D. Yu, R. Mei, L.-S. Luo, and W. Shyy, "Viscous flow computations with the method of lattice boltzmann equation," *Progress Aerospace Sci.*, vol. 39, no. 5, pp. 329–367, 2003.

[39] I. Baran and J. Popović, "Automatic rigging and animation of 3D characters," *ACM Trans. Graph. (SIGGRAPH 2007)*, vol. 26, no. 3, pp. 72:1–72:8, 2007.

[40] T. Igarashi, T. Moscovich, and J. F. Hughes, "As-rigid-as-possible shape manipulation," *ACM Trans. Graph. (SIGGRAPH 2005)*, vol. 24, no. 3, pp. 1134–1141, 2005.

[41] E. Eisemann, S. Paris, and F. Durand, "A visibility algorithm for converting 3D meshes into editable 2D vector graphics," *ACM Trans. Graph. (SIGGRAPH 2009)*, vol. 28, no. 3, pp. 83:1–83:8, 2009.

Chih-Kuo Yeh received the BS degree from the Department of Information Engineering and Computer Science, from Feng Chia University, Taichung, Taiwan, in 2005, the MS degree from the Institute of Bioinformatics from National Chiao Tung University, Hsinchu, Taiwan, in 2007, and the PhD degree from the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, in 2015. His research interests include scientific visualization, computer animation and computer graphics.



Pradeep Kumar Jayaraman received the BTech degree in information technology from Anna University, Chennai, India, in 2009. He is currently working toward the PhD degree in the School of Computer Engineering at Nanyang Technological University, Singapore. His research focuses on computer vision, computer graphics and applications.

Xiaopei Liu received the PhD degree from the Chinese University of Hong Kong, Hong Kong, in 2010, majoring in computer graphics and image analysis. He is currently a research fellow at Nanyang Technological University in Singapore. His current research interests include fluid dynamics and its related visualization and rendering techniques. He is also interested in image analysis and synthesis, as well as GPU-based parallel computing techniques.

Chi-Wing Fu received the BSc and MPhil degrees in computer science and engineering from the Chinese University of Hong Kong, Hong Kong, respectively, and the PhD degree in computer science from Indiana University, Bloomington, IN. He is currently an Associate Professor in the School of Computer Engineering at the Nanyang Technological University, Singapore. His research focuses on interactive methods in the areas of computer graphics, geometric design, visualization, and human-computer interaction. He now serves as an associate editor of Computer Graphics Forum (CGF). He is a member of the IEEE.

Tong-Yee Lee received the PhD degree in computer engineering from Washington State University, Pullman, WA, in May 1995. He is currently a Chair Professor in the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan. He leads the Computer Graphics Group, Visual System Laboratory, National Cheng-Kung University (http://graphics. csie.ncku.edu.tw/). His current research interests include computer graphics, nonphotorealistic rendering, medical visualization, virtual reality, and media resizing. He is a senior member of the IEEE Computer Society and a member of the ACM.