การสร้างโมเดลผมสามมิติโดยอัตโนมัติจากรูปถ่ายจำนวนน้อย

นายณัฐพล วนากิตติเสถียร

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560

Automatic 3D Hair Model from Small Set of Images

Mr. Nuttapon Vanakittistien

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2017

Thesis Title	Automatic 3D Hair Model from Small Set of Images
By	Mr. Nuttapon Vanakittistien
Field of Study	Computer Engineering
Thesis Advisor	Assistant Professor Nuttapong Chentanez, Ph.D.

Accepted by the Faculty of Engineering, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

.....Dean of the Faculty of Engineering

(Associate Professor Supot Teachavorasinskun, D.Eng.)

THESIS COMMITTEE

.....Chairman

(Assistant Professor Attawith Sudsang, Ph.D.)

.....Thesis Advisor

(Assistant Professor Nuttapong Chentanez, Ph.D.)

.....Examiner

(Assistant Professor Nattee Niparnan, Ph.D.)

.....External Examiner

(Peam Pipattanasomporn, Ph.D.)

ณัฐพล วนากิตติเสถียร : การสร้างโมเดลผมสามมิติโดยอัตโนมัติจากรูปถ่ายจำนวนน้อย (Automatic 3D Hair Model from Small Set of Images) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร.ณัฐพงศ์ ชินธเนศ, หน้า.

เราได้นำเสนอระบบในการสร้างโมเดลผมที่เข้ากับทรงผมของผู้ใช้จากรูปถ่าย โดยโมเดลผมประกอบด้วยเส้นผมที่เป็นเส้นหลักจำนวนหนึ่งสำหรับการจำลองการเคลื่อนไหว และโมเดลสามารถจำลองการเคลื่อนไหวได้ตามเวลาจริง เป้าหมายของเรานั้นแตกต่างจากงานก่อนหน้าที่มีวัตุประสงค์เพื่อสร้างผมที่มีรายละเอียดที่ใกล้เคียงความเป็นจริงและนำไปประยุกต์ใช้กับงานที่ไม่ได้จำลองทันทีหรือสร้างโมเดลแบบตาข่ายของผมที่อยู่ภายนอกเพื่อสร้างเป็นภาพ เป้าหมายหลักของเราคือให้ผู้ใช้สามารถนำทรงผมของตนไปใช้ในเกมหรืองานตามเวลาจริงอื่น ๆ โดยถ่ายรูปศีรษะของผู้ใช้ 8 มุมโดยใช้โทรศัพท์มือถืออัจฉริยะ และใช้เครื่องมือเพื่อแยกส่วนรูป ผู้เล่นจะได้ทรงผมของตนใน NVIDIA's HairWorks ซึ่งเป็นการจำลองการเคลื่อนไหวที่ถูกนำไปใช้ในหลายๆเกม เราได้แสดงผลการทดลองที่ได้จากระบบของเราไว้ในวิทยานิพนธ์นี้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2560

5770172821 : MAJOR COMPUTER ENGINEERING
KEYWORDS: HAIR MODELING / HAIR SIMULATION / PHYSICAL SIMULATION / PHYSICS BASED ANIMATION / RECONSTRUCTION

NUTTAPON VANAKITTISTIEN: Automatic 3D Hair Model from Small Set of Images. ADVISOR: ASST. PROF. NUTTAPONG CHENTANEZ, Ph.D., pp.

We present a system for creating 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 work which aims to create realistic high resolution hair for off-line applications or create mesh of the exterior of the hair volume for image manipulation. Our primary aim is for user to be able to put his/her hairstyle into game or other real-time applications. By taking photos in 8 views of the user's head using a smart phone camera and segmenting 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 thesis.

Department: Computer Engineering Student's Signature .....
Field of Study: Computer Engineering Advisor's Signature .....
Academic Year: 2017

ACKNOWLEDGEMENTS

This accomplishment would not have been possible without many people who provide me supports. I would first like to thank my thesis advisor Asst. Prof. Nuttapong Chentanez of the Department of Computer Engineering, Faculty of Engineering at Chulalongkorn University. He always answered my help whenever I had a problem about my research or writing or ran into a trouble spot.

I would like to thank Asst. Prof. Attawith Sudsang, Asst. Prof. Nattee Niparnan and Dr. Peam Pipattanasomporn for my thesis commitee.

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

I also thanks Home Coming Genius scholarship from Department of Computer Engineering, Faculty of Engineering at Chulalongkorn University for 2 years of graduate scholarship.

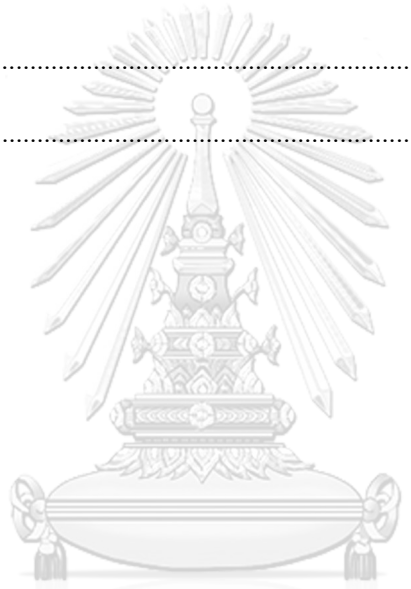
I also would like to thank all seniors, juniors and friends in the department of computer engineering, the Intelligent System Laboratory 2 (ISL2) at Chulalongkorn University for giving discussions about the thesis.

Finally, I must express my very profound gratitude to my family and to my friends for continuous encouragement throughout my years of study and providing me with unfailing support and through the process of researching and writing this thesis. Thank you.

CONTENTS

Page

Page



List of Figures

Page

List of Figures

Page

CHAPTER I
INTRODUCTION

Most characters in games has its own personality. When players create their own game characters, they likely would prefer to express themselves 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 the characters can there be. This potentially make the game more interesting and enjoyable. In most games, player can only select from a handful pre-created of hairstyles which prevent the player from matching their character hair to his/her own. These hairstyles are created by skillful artists who spend a lot of time crafting them. An average game player would likely not possess the skill required to create such a convincing hairstyle, therefore, there are some needs for automatic/assistive hairstyle creation for player.

There are many previous works that reconstruct hairstyle from images, however, most works do not aim to reconstruct guide hair strands. They either reconstruct each individual hair in which there are large numbers or a mesh that represent the exterior volume of hair. Neither of these representations is ideal for real-time hair simulation and rendering for modern games. In this paper, we propose a system that utilizes idea from several recent works on hair capture and add several components to allow average user to be able to capture his/her own hair and export the result to be used in NVIDIA's HairWorks[1]. This thesis extends from our conference paper by improving the image capture app, adding automatic pose estimation, improving background removal process, overlapping the orientation extraction with background removal, and optimizing the running times by almost two orders of magnitudes.

CHAPTER II
RELATED WORKS

2.1 Hair Capture and Reconstruction

A number of works attempt to reconstruct hair from photographs. Paris et al. [2] take photos and test various filters to decide which one is the best for estimating 2D orientation map. Wei et al. [3] propose a method for combining 2D orientation maps from several views into a single 3D orientation field. Paris et al. [4] propose a method for diffusing 3D orientation of known voxels to unknown voxels by utilizing a tensor representation. Jakob et al. [5] propose a method for capturing hair strand-by-strand. Chai et al.[6, 7] create the mesh of the exterior of the hair from a single photo of human face with a few guiding strokes drawn by user. Luo et al.[8] reconstruct hair from point cloud data. Hu et al.[9] propose a hair reconstruction technique that utilizes simulated hair strands examples database. Braided hairstyle is reconstructed in [10]. Hu et al.[11] compare user's strokes to a databases of hairstyles and use the information to synthesize a new hair model. Chai et al. [12] generate high quality hair model for 3D printing application. Chai et al.[13] no longer require user stroke for generating the hair model from a single photo, however they still need to make strong assumption about the appearance of hair at the back of the head. Most recently, Zhang et al.[14] use four-sided hair images for hair modeling. To generate a rough model shape, they need hair model database to match with each side of the images separately and combined them together.

2.2 Real-time Hair Simulation

In recently years, real-time hair simulation techniques based on Position Based Dynamics (PBD) [15, 16] are popular. Position based dynamics is one of many physically-based simulation methods for deformable objects in real-time. Although this method is not as accurate as some other off-line approaches, it conserves momentum produces visually plausible results. The core concept of PBD is the use of

position as the physical quantity in which velocity is derived from. Constraints are defined between particles to achieve desired interactions. PBD has been used for simulating many types of object in real world, such as cloth, soft body [15], liquid [17], smoke, sand [18] and hair [19]. Han and Harada[20] proposes a real-time hair simulation using local and global shape constraints. Kim et al.[21] propose a long range attachment constraint to enforce inextensibility for hair simulation with PBD. Umetani et al.[22] propose a method for simulating curly hair in PBD by using constraints on Daboux vectors. PBD hair simulation method is implemented in Nvidia HairWorks[1] which has been used in many games such as Call of Duty Ghosts, Farcry 3, and Witcher 3 and can be used in popular game engines such as Unreal Engine 4.

CHAPTER III

OVERVIEW

We start the process by collecting the data from user's camera in Image Data Acquisition. Then we segment the images and find 2D Orientation from the images. Next, we find Pose Estimation from front view, front left view and front right view from the images. The position of head model will approximately be adjusted in the right position. We then reconstruct the Visual hull from 8 views of images and generate 3D Orientation Field in hair volume, so that we can track the guided strands from root of hair model to the tip. Finally, we export our strands result to HairWorks to view our result in real-time simulation. The overview of our process is shown in figure 3.1.

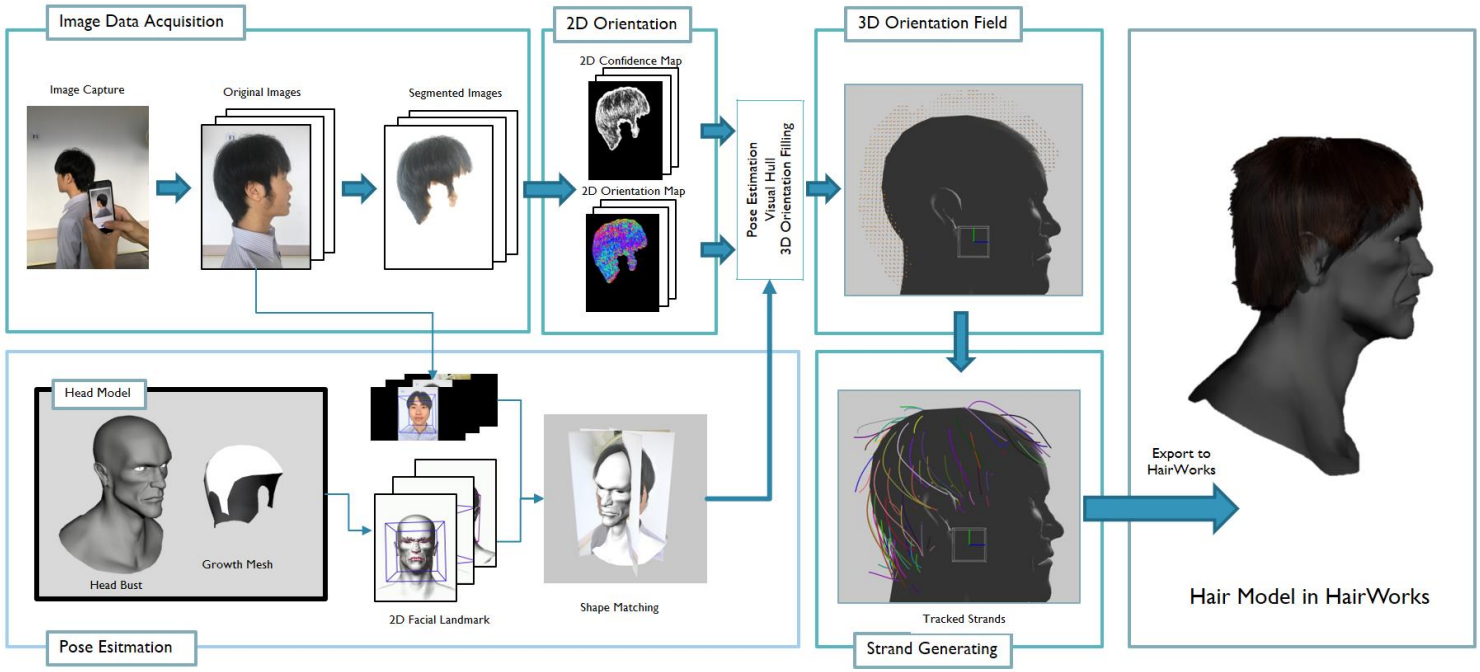


Figure 3.1: The overview of our guide hair strands capture method.

CHAPTER IV
IMAGE DATA ACQUISITION

4.1 Capturing Setup

We developed a prototype mobile application for iOS device to capture images of the face and the hairstyle from various views. In addition to capturing the images in various devices and platforms, we also developed web application which can access device's camera such as smartphone's camera and PC's webcam. We need another person to take photos from 8 views around player's head. Alternatively, the player can take photos of him/herself using a long selfie stick. Figure 4.1 shows examples of such images. Each view is 45 degrees apart starting from 0 degree (front of player's face) to 360 degrees. 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

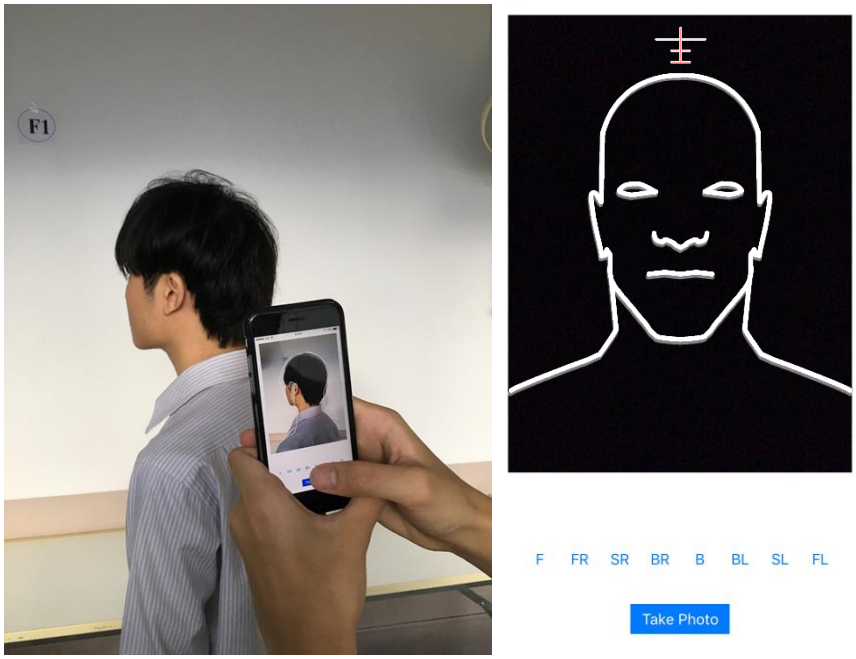


Figure 4.1: Left: Process of taking photo of a user head from 8 views. Another user can hold smart phone camera and use the capturing application, or the user can use a long selfie stick. Right: Our prototype mobile application interface, which was developed for iOS device.

match with the template. Moreover, we also add an indicator line above the head template. Users can align the top of the hair to the same height in every image. As shown in Figure 4.2, player's head photos were captured on our prototype application in 8 views. The 8 images are saved for further processing in the next step.

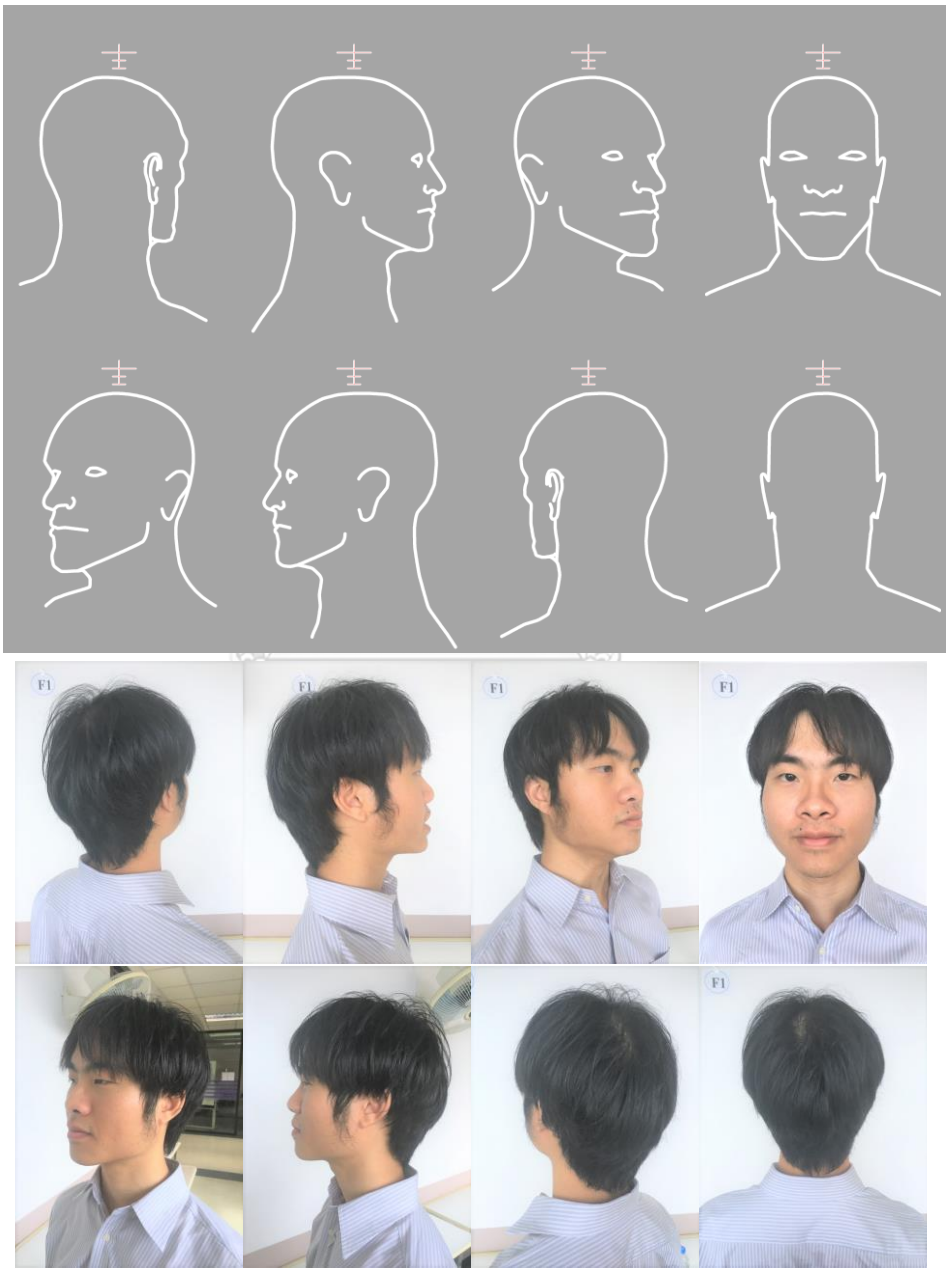


Figure 4.2: Template faces in 8 views and our captured images from aligning the camera in the same viewpoint

4.2 User-Image Segmentation

We develop a software for constructing the hair model. After the user imports the 8 images into our software, the first step is to segment the hair region out of each image. We utilize Grabcut [23] which allows the user to draw a few simple lines or shapes to indicate the hair or non-hair 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 3-5 strokes per image. An example of annotation is shown in Figure 4.2. 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 4.3: 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)

4.3 Pose Estimation

After the images are segmented, we next perform a pose estimation for fitting our head model to the image. We then use OpenFace[24] to find 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 sometimes, OpenFace is not able to reliably detect the facial landmarks from a single image, especially on the side views. The reason is our side view which rotates too much from the front view by using the tool. We solve this problem by running OpenFace on video instead of a single image. Each video contains translation and zooming of a face image. The OpenFace will then adjust the landmark frame by frame and the last frame will contain much more accurate landmarks compared to just using a single image. The video's resolution is 1920x1080 pixels and contains 64 frames. We create the video automatically from the input photo by translating the image around and using ffmpeg to produce the video stream from each frame and encode it to an mp4 file. This step is demonstrated in our accompanying video.

2D facial landmarks from 3 views are used for finding 3D facial landmark. We then use least square fit to solve this problem by using 2D facial landmarks as input and the orthogonal projection as projection matrix from 3 views whose angles are 45, 0 and -45 degrees from the up axis.

The 2D landmark locations of the original head model are found similarly by running OpenFace on a video of the rendered head model in the corresponding angle being translated around. As shown in figure 4.3, we repeat for the 3 views and then use least square fit to obtain the 3D locations. The 3D facial landmarks of the capture image and the original model are then used for computing the closet head model's translation and rotation by shape matching [25]. Let $\mathbf{x_i^0}$ be set of 3D landmark point from images, $\mathbf{x_i}$ is 3D landmark point on 3D head model. Then we find translation, $\mathbf{t}$, rotation, $\mathbf{R}$, and scale, $\mathbf{s}$, by minimizing the equation:

$$\sum_i w_i \left(sR(x_i - t) - x_i^0 \right)^2$$

where w_i is the weight of each point, which we fix the values be 1 for all $\mathbf{i}$. After the best fit transformation is computed, the user can still make adjustment, if needed, within our software.

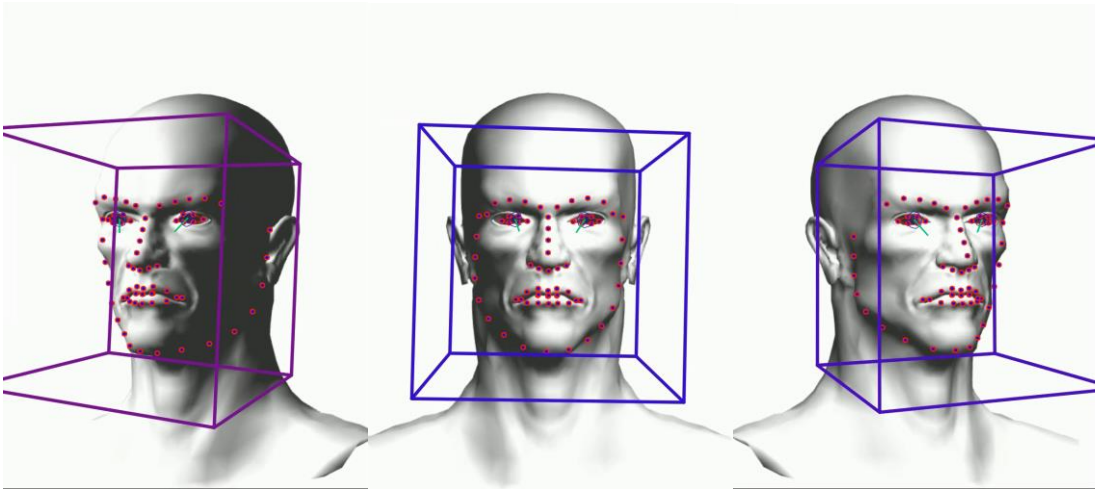


Figure 4.4: 2D Facial Landmark in 3 views from 3D head model, to find 3D Landmark for matching with 2D landmark for captured images

CHAPTER V
ORIENTATION FIELD

5.1 2D Orientation Acquisition

We follow the method from [2, 4, 7] for extracting 2D orientation using bank of Gabor filters [26]. Let the grayscale segmented image be I and the Gabor convolution kernels be

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

where $\mathrm{a} = \sigma_{\mathrm{u}}^{-1}(\mathrm{u}\cos(\theta) + \mathrm{v}\sin(\theta))$ and $\mathrm{b} = \sigma_{\mathrm{v}}^{-1}(\mathrm{v}\cos(\theta) - \mathrm{u}\sin(\theta))$. Our convolved image is then $\mathrm{F}(\mathrm{x},\mathrm{y},\theta) = |\mathrm{K}_{\theta} * \mathrm{I}|_{(\mathrm{x},\mathrm{y})}$. In our experiment, we found that the values similar to those used in [6] produce best result. We use the kernel size of 9×9, $\sigma_{\mathrm{u}} = 1.8$, $\sigma_{v} = 2.4$, and $\lambda = 4$. To make the result more robust, we perform super sampling in each pixel of the kernel by dividing each pixel into 10×10 uniform grid. The values of the super pixels are then averaged and used. 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 $O(x,y) = \tilde{\theta} = arg\max_{\theta}\big(F(x,y,\theta)\big)$. Moreover, each orientation pixel also has a confidence value. The 2D confidence map [2] is calculated using

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

where $\Delta F\big(\theta,\tilde{\theta}\big) = F(\theta) - F\big(\tilde{\theta}\big)$ 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 input image and repeat the above steps for 3 iterations as suggested in [7]. Figure 5.1 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 8 images.

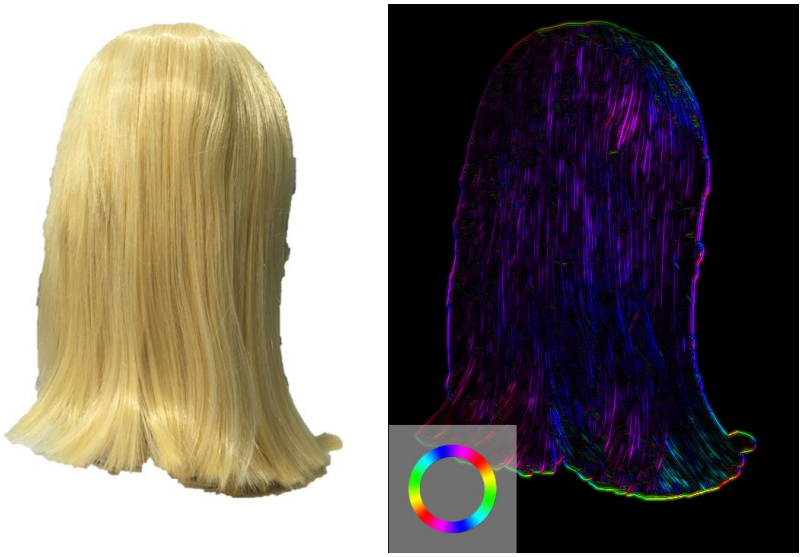


Figure 5.1: Left: Segmented image from the right-back view of a wig. Right: 2D orientation map after doing 2 iterations of iterative refinement. The direction and the confidence value are visualized as hue and brightness respectively

In the capturing step, in order for good orientation information to be extracted, the camera should see the anisotropic specular reflection of the hair strands. Gabor filter gives better result if the area with specular reflection on the hair strands in the image has higher contrast with respect to the non-specular area. We found empirically that dark or black colored hair colored hair tend to be more robust, for 2D orientation extraction. To improve the quality of orientation, brightness and contrast should be adjusted, or more iterations of refinements should be used. Greasy hair naturally tends to have higher contrast than matte hair. Nonetheless, our iterative improvements produce 2D orientation for most matte hair with sufficient quality for further processing steps.

5.2 VISUAL HULL

We use the visual hull [27] as the rough shape of the hair volume. It can be constructed from the segmented images. For simplicity, 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 degrees apart, which is shown in figure 5.2. For each voxel, we perform orthogonal back projection to each of the image plane to see if the voxel lies within the hair region in that view. If a voxel lies in the hair in more than 3 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 2 iterations of erosion followed by 2 iterations of dilation on the voxel grid. In our examples, we use grid resolution of $90 \times 67 \times 67$ for processing the visual hull. The orthography, the equal distance and the equally space assumptions clearly do not hold in practice as the mobile phone camera has 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 is also robust against these inaccuracies and can reconstruct guide hairs from various hairstyles.

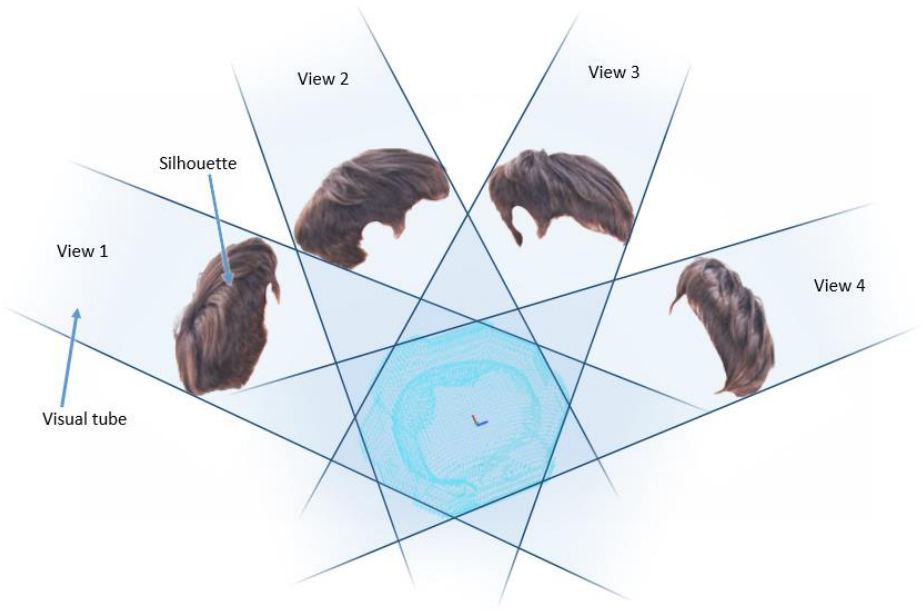


Figure 5.2: 2D orientation field from four of the eight views are shown in this figure. We assume that the photo taken are the orthogonal projections. Visual hull filled voxels are also visualized in the middle

The reason why we use orthogonal projection in our method is that users do not need to calibrate any cameras before use. We start to use Orthogonal Projection in our method at reconstructing the visual hull, which our algorithm gets the visual hull shape as windmill shape at the top view (in figure 5.3). Our visual hull shape looks inconsistent because of uncalibrated image with orthogonal projection instead of using perspective projection. However, the resulting visual hull is good enough for yielding reasonable quality result in practice.

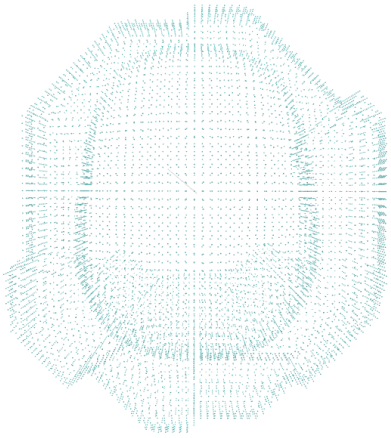


Figure 5.3: Visual hull of our method at the top view

5.3 3D Orientation Field

After obtaining the 2D orientation maps of the 8 views, we next compute the 3D orientation of the outer filled voxels by following the method from [3] 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 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. If there is only one visible view, we ignore that voxel (Figure 5.4). If there are two visible views with image plane normals $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$ 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 $N_i = n_i \times v_i$ (Figure 5.5). If there are three or more visible views, we can compute D by minimizing

$$\sum_j \sigma_j^2 (N_j \cdot D)^2$$

, subject to $||D||_2 = 1$, where σ_j is the uncertainty computed as one over the 5x5 box average of $w(x, y)$ around that pixel. The solution for D is the eigenvector corresponding to the smallest eigenvalue of $\sum_j \sigma_j^2 (N_j^T N_j)$ (Figure 5.6). Note that for the purpose of strand tracking, D and $-D$ are considered the same, hence we refer to it as orientation.

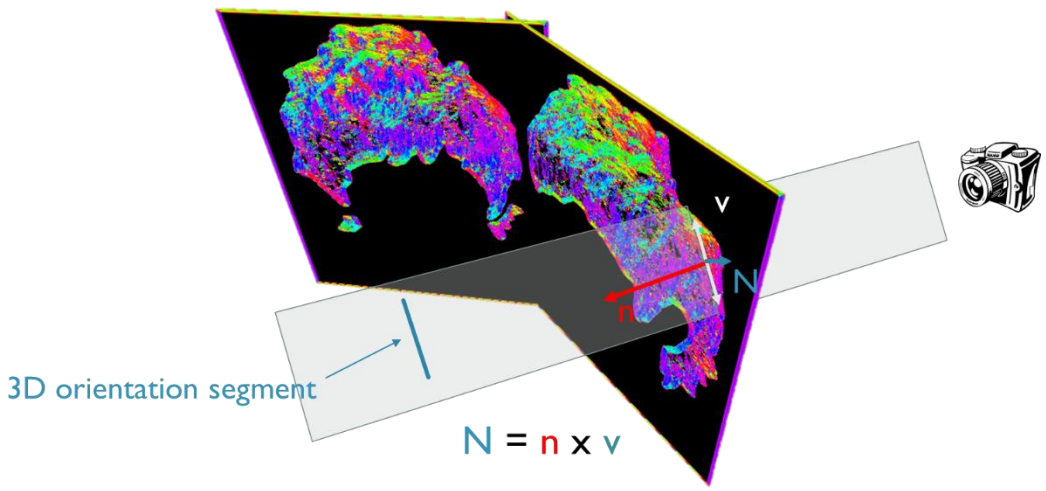


Figure 5.4: The 3d orientation is computed by checking if it is within hair region of that view and the ray from the center of the grid toward that voxel intersects with the image plane of that view.

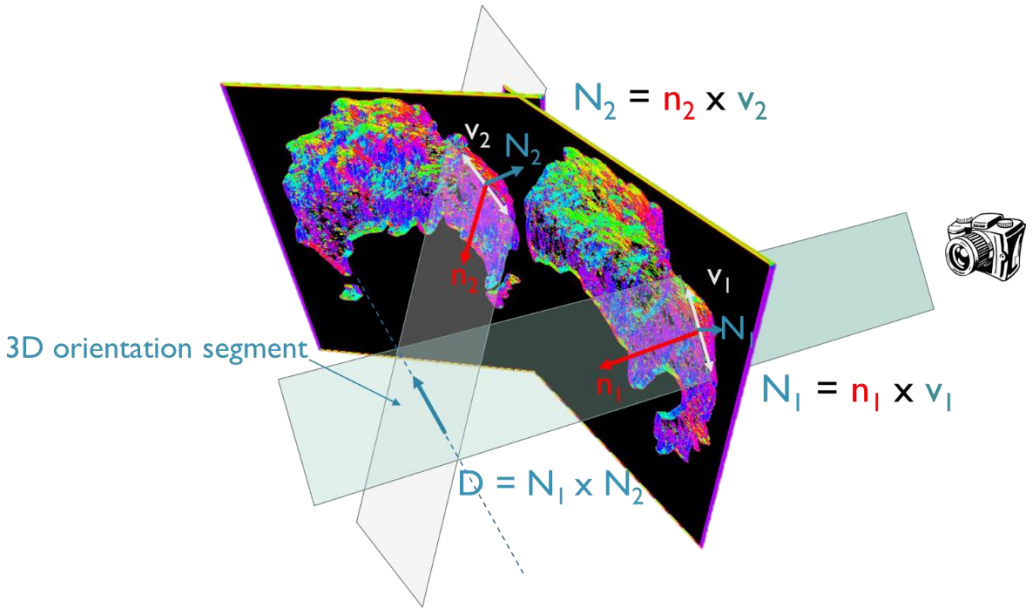


Figure 5.5: When 2 views are available, the 3d orientation will be computed by intersection of 2 planes. Each plane lies on 2d orientation pixel of that view, expressed as 3d vector, and lies on normal of image plane.

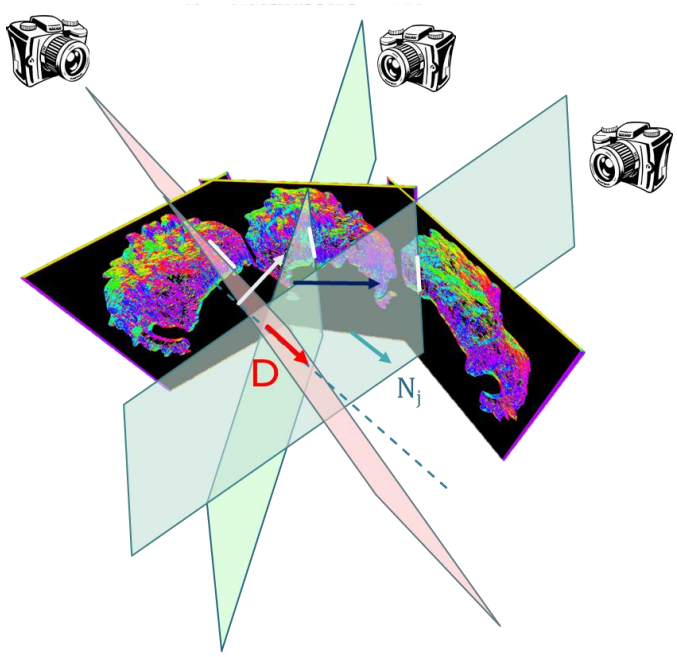


Figure 5.6: If there are 3 or more visible views, the 3d orientation will also be computed by intersection of all visible planes. Each plane is weighted by 2d confidence map.

5.4 Orientation Filling in Hidden Hair Volume

We now have the 3D orientation 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. We next identify the voxels that intersect with the growth mesh and set the 3D orientation of such voxels to the interpolated vertex normals of the intersected triangle. We are now ready to extrapolate the orientation to other voxels. First, we convert the 3D orientation, $\boldsymbol{D}$, at each known cell, into a 3×3 tensor, DD^T. Structure tensor represents orientation in 3×3 symmetric matrix. The orientation is the eigenspace corresponding to the largest eigenvalue of the tensor. Its advantage is that we can get the result which is the fastest variation of sum of the tensors. The example of structure tensor is shown in Figure 5.7 which represents in 2D space. We then perform an isotropic diffusion of the tensor to all other voxels using the method from [4], where we use the known voxels as Dirichlet Boundary conditions. We fix our outer voxels with 3d orientation from multi views of 2d orientation. Moreover, we fix the voxel that intersect with head mesh to normal of the mesh as well. As shown in Figure 5.8, the red orientation lines are fixed constraints and the blue orientation lines are results of diffusion in several iterations until we ensure that the extrapolated tensors seem not to be changing. We then convert each tensor back to 3D orientation by finding the eigenvector corresponding to the smallest eigenvalue of the tensor.

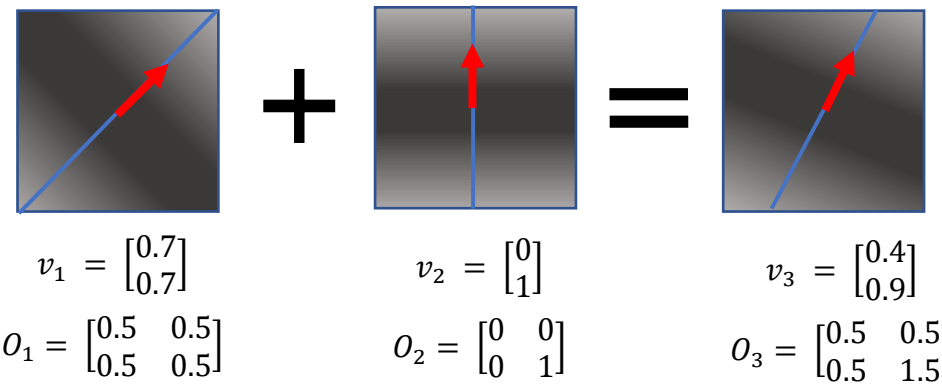


Figure 5.7: In 2D space. Let v be direction vectors and O be structure tensors we build, vv^T. Sum of these tensors is O_3 which is extracted to the eigenvector of the largest eigenvalue v_3. In the same method in our 3D space, v are 3D direction vectors and tensors are 3×3 symmetric matrix.

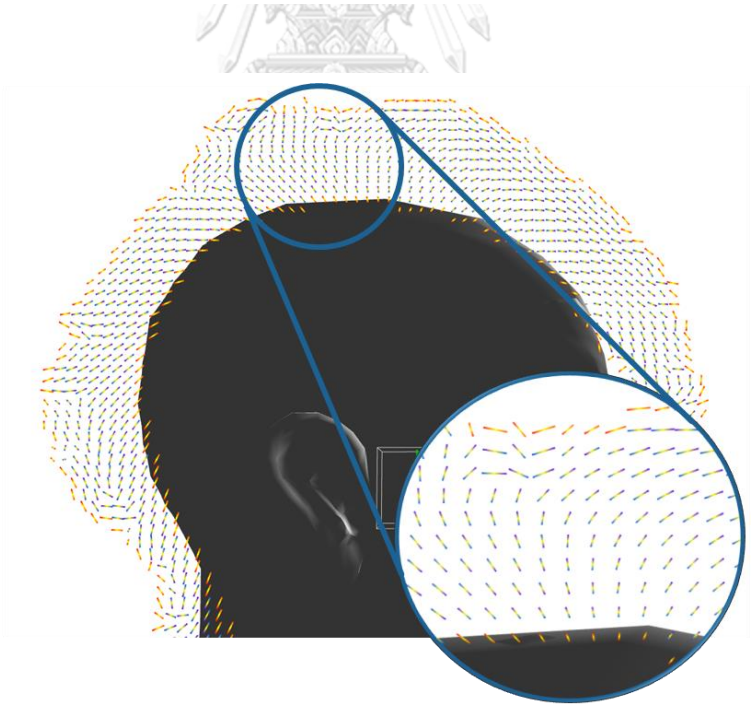


Figure 5.8: The red orientation lines: fixed constraints, the blue orientation lines: result of diffusion in several iterations until it seems not to be changing

CHAPTER VI
GUIDE HAIR STRAND GENERATION

6.1 Strand Tracking

We trace along the 3D orientation field using a method suggested by [4] enhanced with several modifications inspired by [6, 7]. Note that we only perform tracking to generate guide hair strands, not each individual hair strands. Several hundreds guide hair strands are sufficient for representing most hairstyles. During rendering, these guide hair strands are used for interpolating lot more rendered hairs using tessellation unit in 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 degree around $n \times [0,-1,0]^T$ axis. The user can choose G in our program where smaller value is used for long hair styles and larger value should be used for long hair styles. In our examples, values between 40-60 degree 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 visual hull, change the current state into "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 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 $\theta_{max} = \pi/3.7$ The guide hair strands obtained from our tracking algorithm can still be jagged due to inaccuracy in 3D orientation estimation. We hence smooth them using the smoothing method suggested in [28] for 5 iterations, resulting in a sufficiently smooth curves. We perform the tracking algorithm described for each of the growth mesh vertices to obtain all the guide hair strands.

6.2 Trimming Erroneous Strand

In practice, a few of the tracked strands are erroneous as the algorithm tracks down a strand and incorrectly tracks back up along another strand, resulting in a U-shape strand. In general, a hair strand tends to curve downward due to Earth's gravity. Although some hairstyles have S-shape where the tip rises, no hairstyle commonly found has U-shape.

We hence check if the strand should be trimmed or not. Let $\overline{p_1}, \overline{p_2}, \dots, \overline{p_n}$, be the smoothed position of the vertices along a strand. Let i be the smallest index where $\overline{p_i}.y - \overline{p_{i-1}}.y < 0$. We count the number of vertices $k > i$ where $\overline{p_k}.y - \overline{p_{k-1}}.y > 0$. If the number of such vertices is greater than $\mu(n - i - 1)$, we trim the hair strand at the vertex j, where j is the smallest index greater than i such that $\overline{p_{j+1}}.y - \overline{p_j}.y > 0$. The counting procedure aims to detect if the hair strand unnaturally forms a U-shape. In all our examples, we use $\mu = 0.3$.

CHAPTER VII
EXPORTING TO HAIRWORKS

HairWorks requires that all the strands have the same number of vertices, m. We hence re-sample each strand curve into $m-1$ equal length edges. After that, the hair model is ready to be exported for simulation in HairWorks. We write the strands into an .apx file as guide hair strands and also create a .fbx file for the head model. As shown in Figure 7.1, head model and guide hair strands are imported to the HairWorks Viewer Tool. We set the simulation and rendering parameters to be similar to what's used in the examples provided with HairWorks.

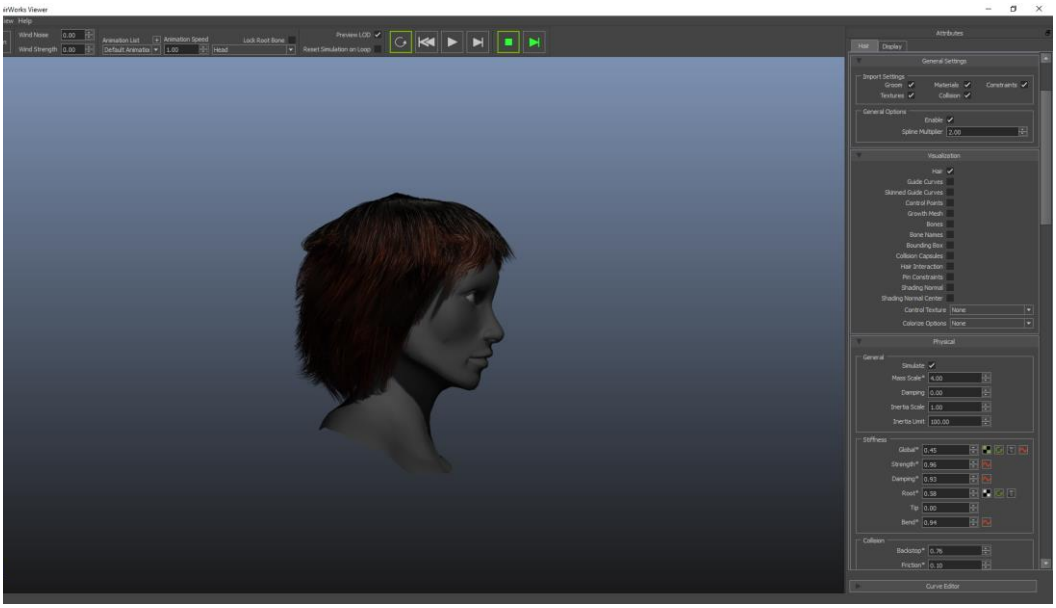


Figure 7.1: HairWorks Viewer Interface allows user to adjust various simulation and rendering parameters. We import the guide hair strands generated with our method this program.

CHAPTER VIII

SPEED OPTIMIZATION

We optimize our method to gain speed up over the experiment we did for [29]. In that work, our processing time was about 30-40 minutes on our PC Laptop (CPU Intel Core i7-4700HQ and GPU Nvidia GTX 860m. With our new optimization presented in this thesis, from image editing process to finishing exporting to HairWorks file, the processing time is about 30 seconds after the user finish the last segmentation. To summarize, we proposed did the following optimizations

1. Utilizing the background threads for 2D Orientation extraction to overlap this time with user segmentation of the next image and 3D pose estimation.
2. Using bounding volume hierarchy to accelerate rat casting when checking if a grid point is in the growth mesh or not.
3. Optimizing the 3D orientation field construction by code optimization down from 8-15 minutes to under 20 seconds.

CHAPTER IX

RESULT

We collect the data from a number of volunteers. All data from this paper are either taken with our smart phone (iPhone 6s in our experiment) or captured from still frames of Youtube video. The results from our algorithm are simulated and rendered in HairWorks. They visually match with the real-world counterpart closely. We also test our system with photographs of wig on a wicker head, as shown in Figure 9.1. More results can be found in our accompanying video. The resulting hair strands of all examples are simulated and rendered in real-time. We provide 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

As shown in Figure 9.4(a), the first and the second row show images are woman's hairstyles. The first one is short hairstyle and another row is very short hairstyle. The third row shows a hairstyle reconstructed from still frames of a Youtube video of a man recording his hairstyle from 360 degrees. We select 8 frames from the video that roughly match with the 8 views required and use them as input to our system.

In figure 9.4(a), the fourth row shows a long hair style, which we can also see this hair style's result in figure 9.3 and 9.4. The first row in Figure 9.4(b) shows a man's hairstyle with curly short hair. The second row and the third row in Figure 9.4(b) show men's short hairstyles. In most examples, the user only need to make small adjustment to the 3D pose found by our automatic pose estimation. However, in a few examples, the 2D facial landmarks are not accurate and hence require the user to make moderate adjustment to the pose. The rendering parameters like hair color, curliness, etc. are chosen by the user in the hairwork to obtain desired look.

9.1 Limitations

We can create guide hair strands suitable for real-time simulation and rendering from various hairstyles. Unfortunately, some complex hairstyles, such as fizzy hairstyle and highly curly hairstyle might not be tracked accurately using our method.

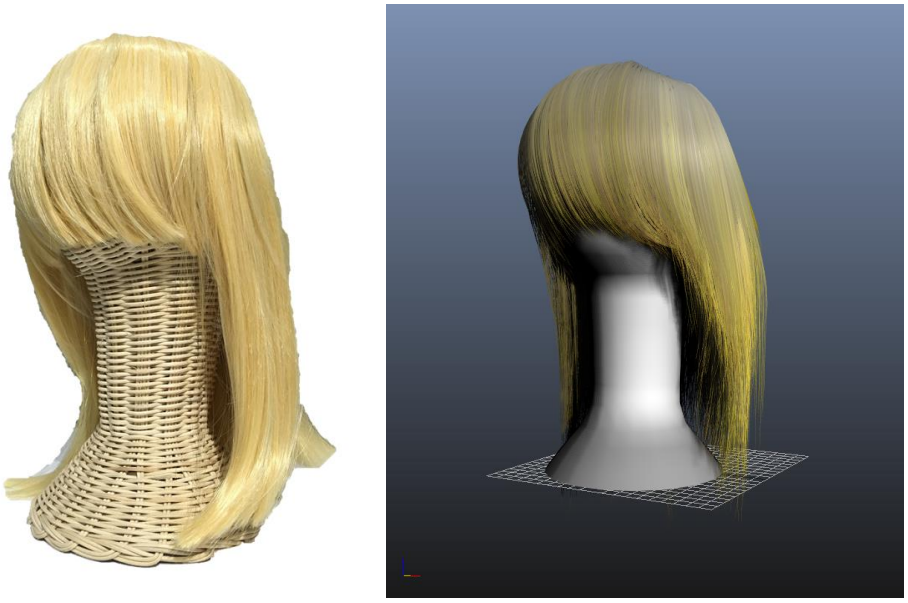


Figure 9.1: Left: The wicker head with wig. Right: Our result, simulated and rendered in real-time in HairWorks viewer from a similar viewpoint.

In case of curly hair in figure 9.2, the result shows the failure case as our limitation that the strands shape does not match with the real hairstyle very well. The Image in the figure shows visual hull shape and real hairstyle at the same side. The shape is quite similar to the real one. The strands as the result are tracked inaccurately which is clearly visible when viewed HairWorks. We believe the reason is that the 3D Orientation computed from multiple views of inconsistent 2D Orientation maps is not accurate. The erroneous 3D orientation field results in erroneous strands being produced.

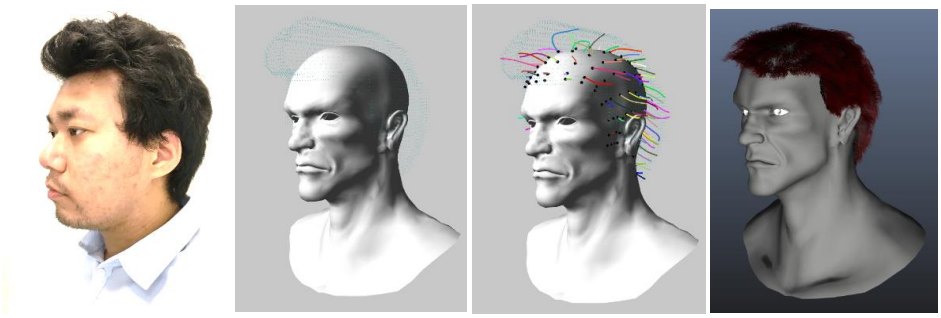


Figure 9.2: A failure case on a curly hairstyle, from left to right, a captured image, reconstructed visual hull, tracked strand, and the result in HairWorks from a similar viewpoint

Figure 9.3: A snapshot of animation of hair whose guide strands are captured with our method

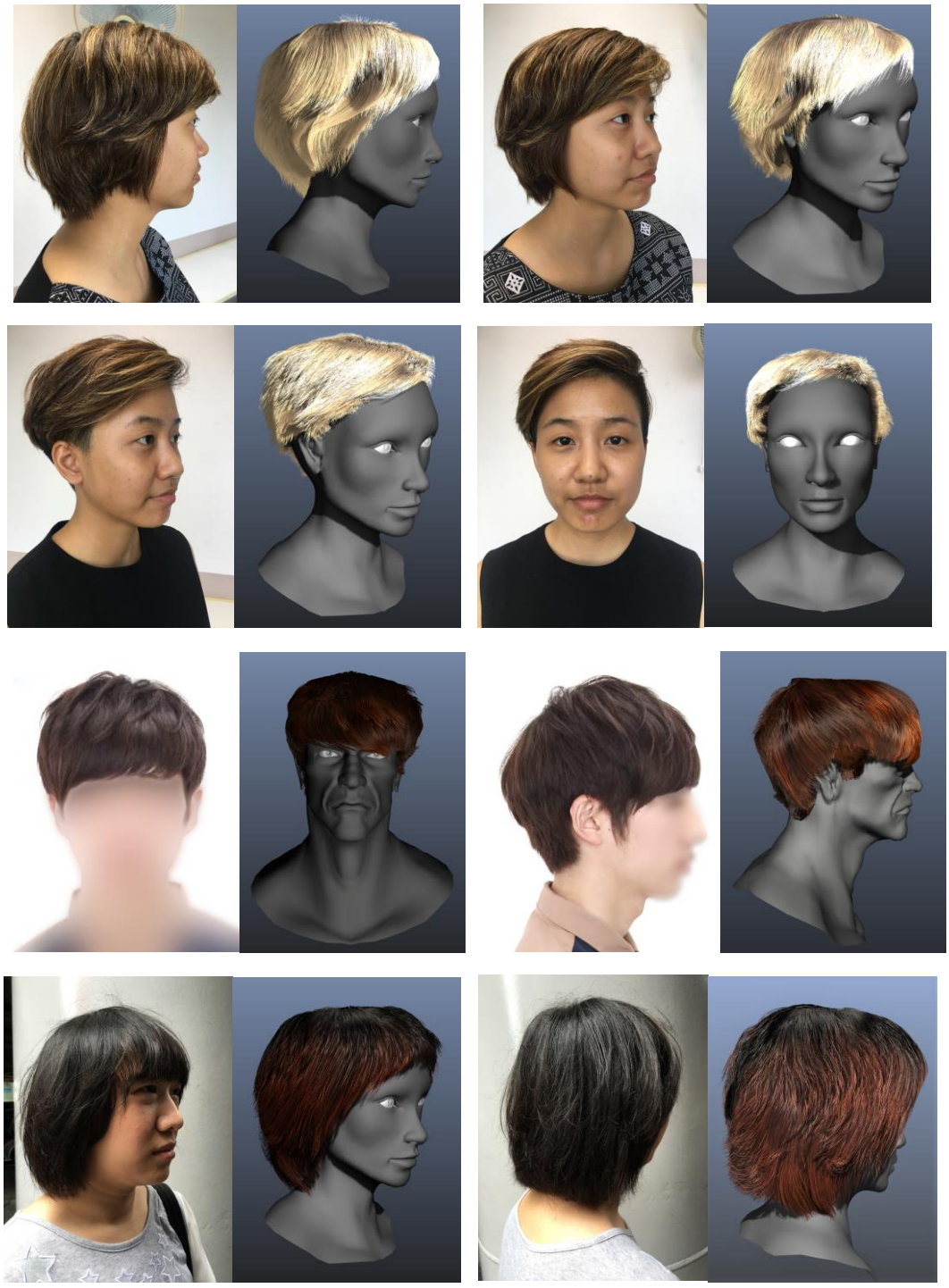


Figure 9.4(a): Left: Input images of four different hairstyles. Right: Our result, simulated and rendered in real-time in HairWorks viewer from a similar viewpoint.

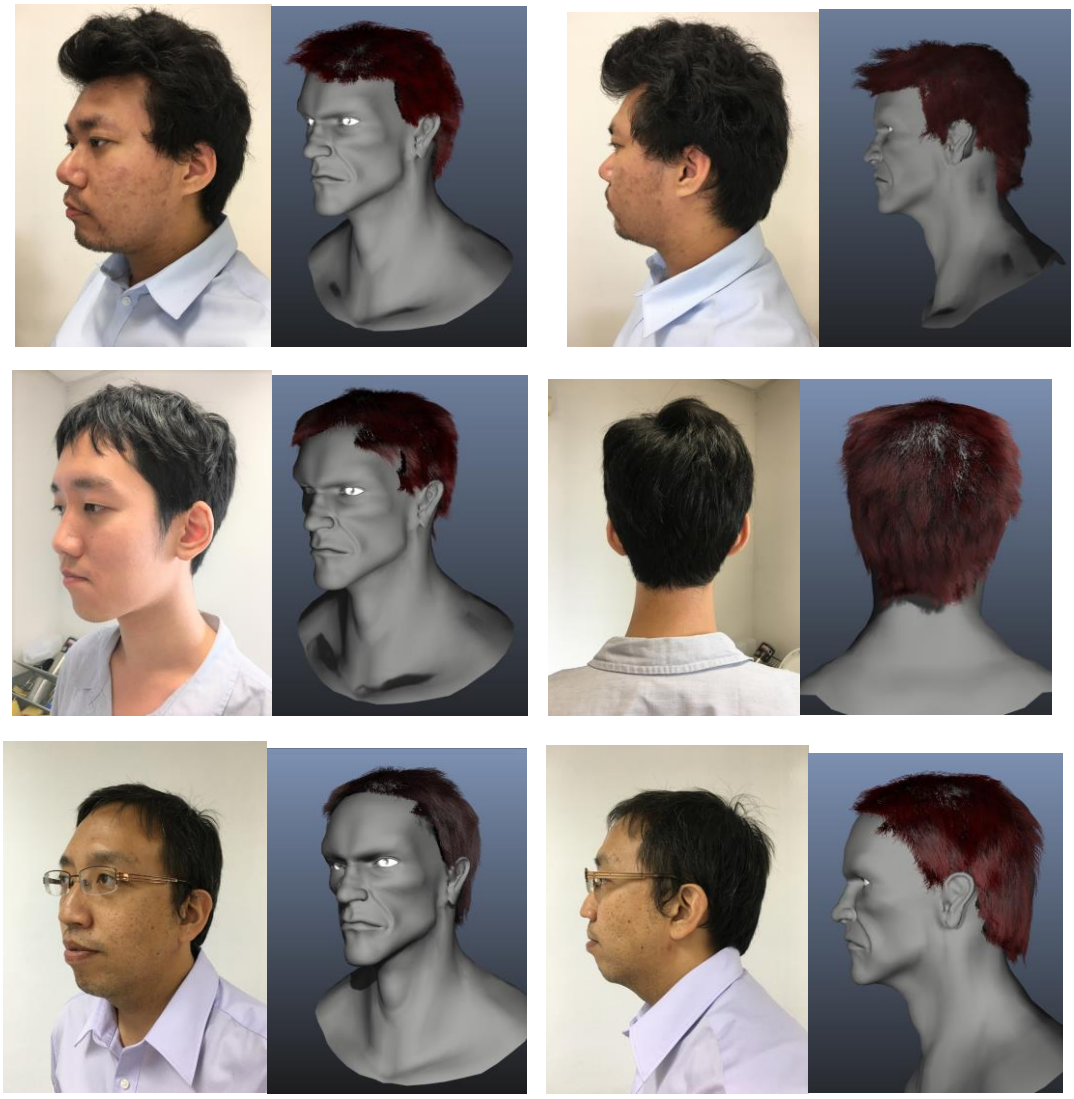


Figure 9.4(b): Left: Input images of three different hairstyles. Right: Our result, simulated and rendered in real-time in HairWorks viewer from a similar viewpoint.

In our work, the resources used mainly come from HairWorks examples, including the head model. There are only two models: male and female. A problem appears when we want to capture a very short hairstyle. We must adjust the model to fit with the visual hull whose shape is significantly different from the model. To generate a reasonable quality strands, the model's vertices must lie inside visual hull. If any vertex does not lie inside the visual hull, the strands at that vertices cannot be tracked. For the future work, if the head model shape could be created automatically to fit the captured images, we should be able to utilize it to track the strands more accurately.

For very short hairstyle cases, the visual hull likely misses some portion of the head. To solve this issue, users can rescale and re-position the visual hull so that it fits better with the data.

In future work, changing the camera's templates to perspective projection is an interesting idea to explore. It may be possible to obtain intrinsic camera parameters of popular mobile phone camera models. This would allow for potentially more accurate visual hull and orientation to be extracted.

9.2 Survey

We made a questionnaire and ask 32 people to take it. The questionnaire consists of 8 questions, each question has 4 choices. In the first 4 questions we ask the participant to choose a reconstructed hairstyle that best matches with a given photo. Based on the answers we received, we found that most people choose the correct answers except for the second question. As shown in Figure 9.5:Question 2, in this example, the correct choice is number 2, while most people chooses number 4, which is one of the Hairworks's provided example. Visually, it does look very similar to the photo so it comes at no surprise that people choose this choice.

The last questions ask the user to choose the photo that best matches with a given reconstructed hair. We found that in question 7, most people chose a wrong

answer. Choice 2 and 3 are very visually similar, however choice 2 is a dyed hairstyle which happens to visually matches better with the rendered image. For the other questions, most people still chose the correct answers. Figure 9.5-9.8 show our questionnaire and the number of people choosing each of the choices.

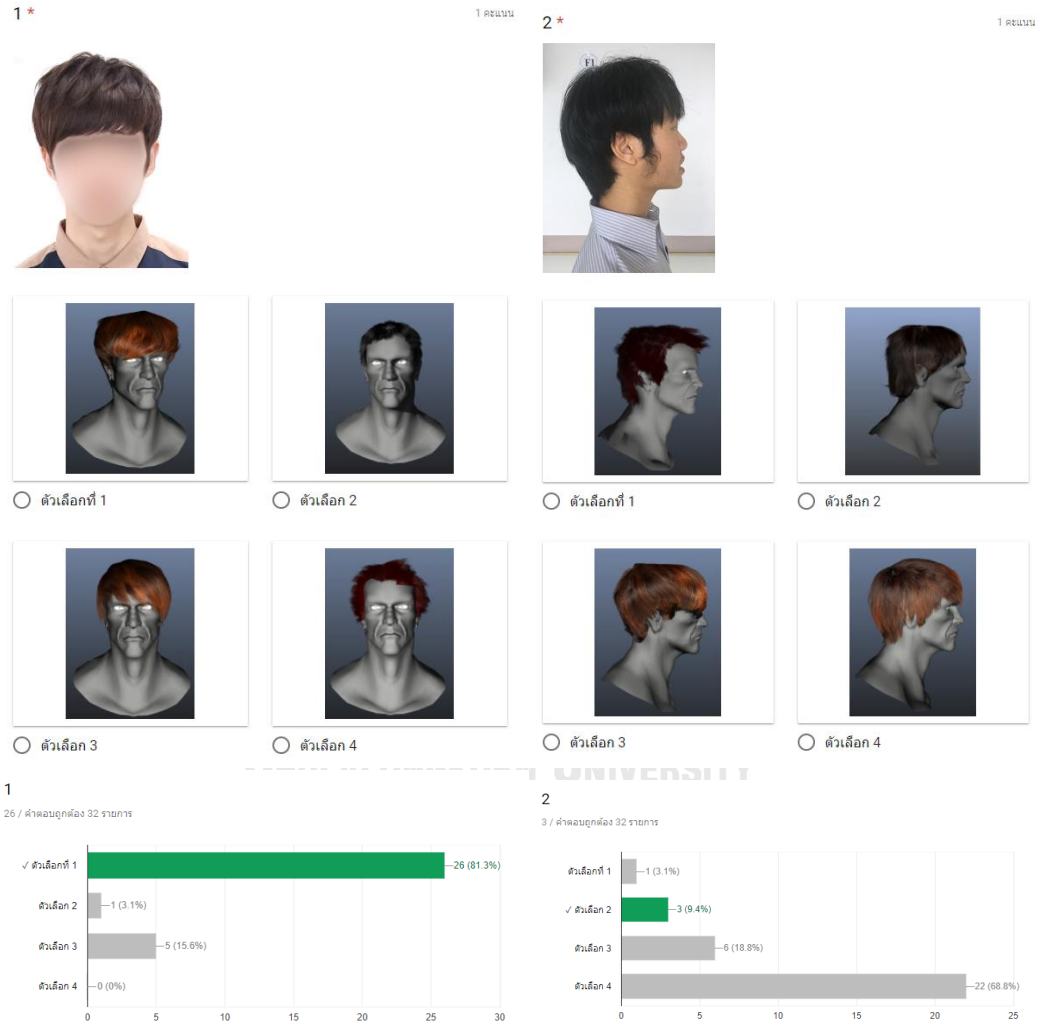


Figure 9.5: Question 1 and 2, with stacked bar of selected choices below each question

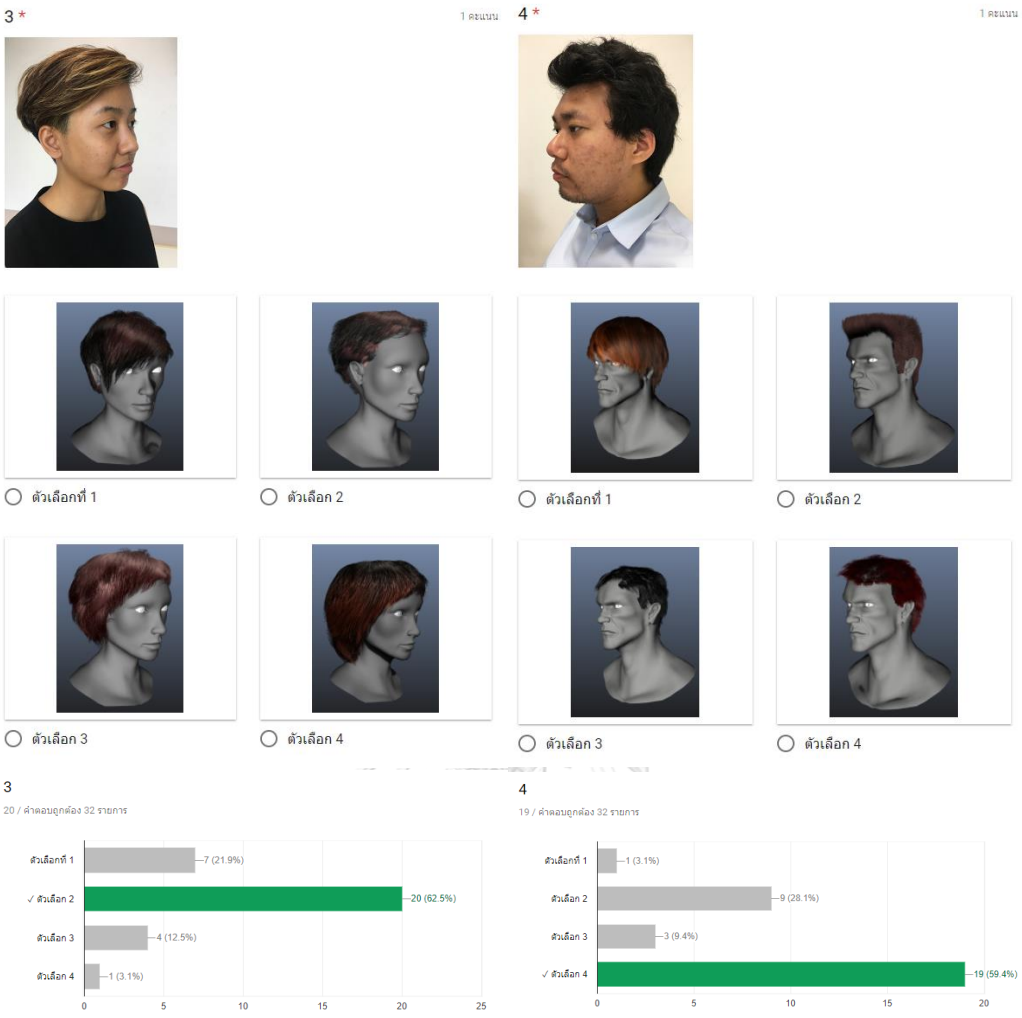


Figure 9.6: Question 3 and 4, with stacked bar of selected choices below each question

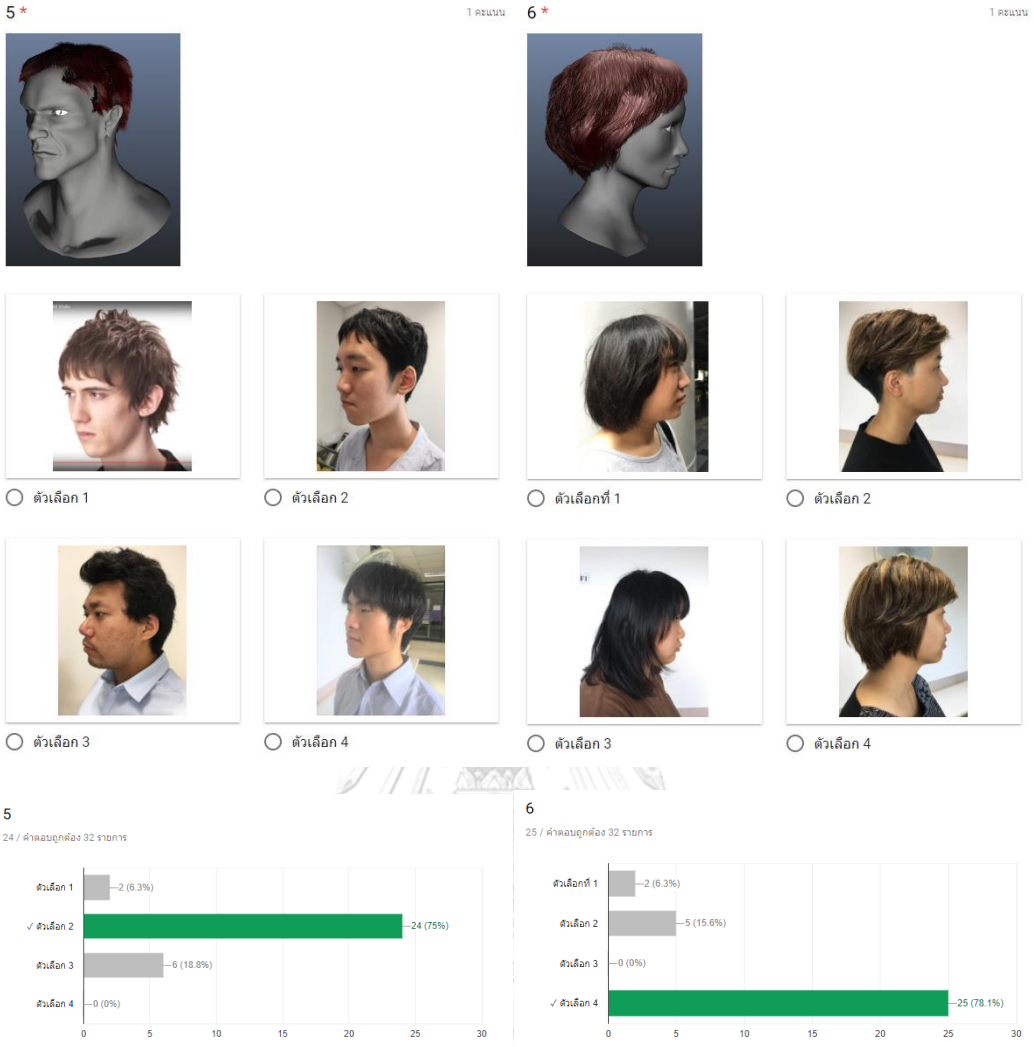


Figure 9.7: Question 5 and 6, with stacked bar of selected choices below each question

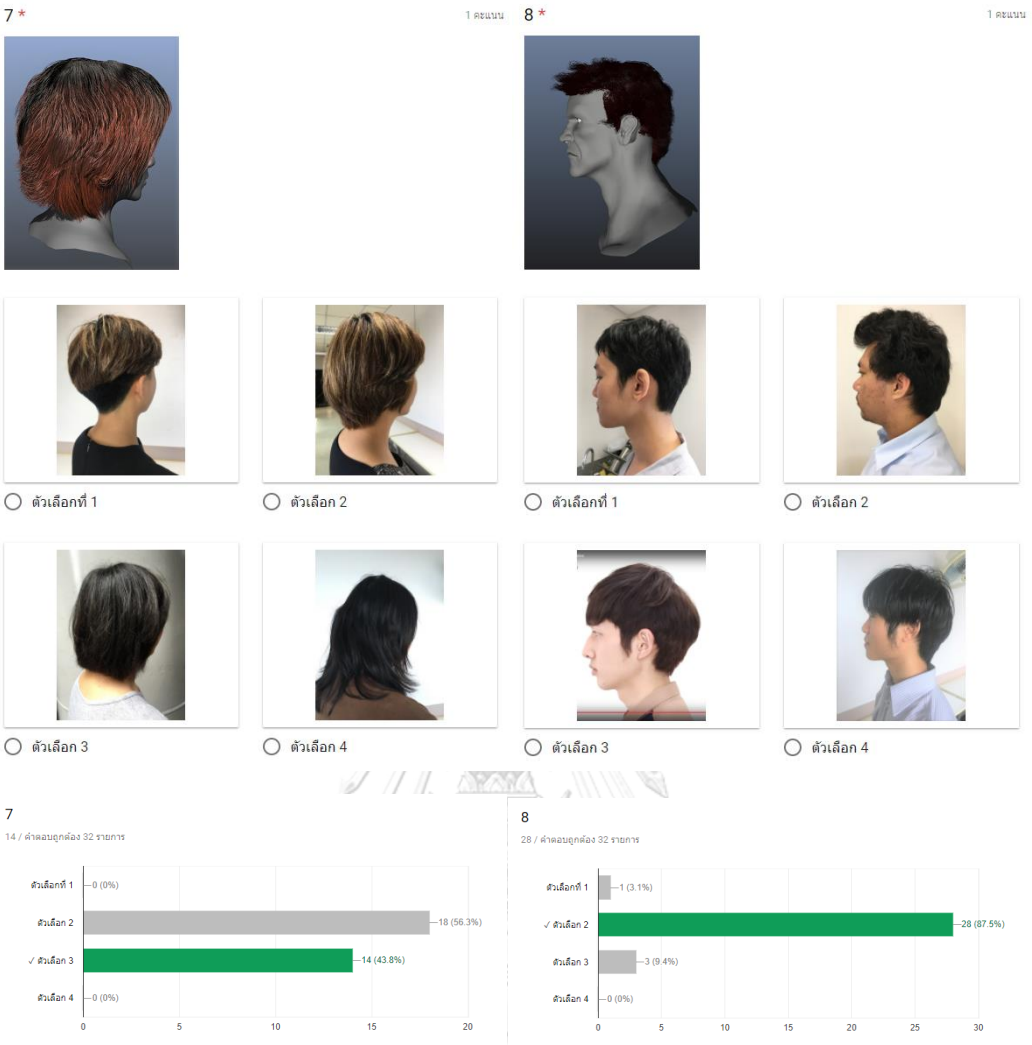


Figure 9.8: Question 7 and 8, with stacked bar of selected choices below each question

CHAPTER X
CONCLUSION

With our system, users can put their hairstyles into games with ease. A user can acquire the hairstyle data by using a smart phone camera and our mobile app to take photo of 8 views around his/her head. The images should then be segmented and use as input into our algorithm which generate 2D orientation fields, pose estimation, visual hull, 3D orientation field and finally track the guide hair strands. The resulting 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, estimating growth mesh position and scale automatically, optimizing the algorithm to run faster and improving the tracking algorithm to be able to handle more complex hairstyle

REFERENCES

1. NVIDIA, *NVIDIA HairWorks*. 2015.
2. Paris, S., H.M. Briceño, and F.X. Sillion. *Capture of hair geometry from multiple images*. in *ACM Transactions on Graphics (TOG)*. 2004. ACM.
3. Wei, Y., et al. *Modeling hair from multiple views*. in *ACM Transactions on Graphics (TOG)*. 2005. ACM.
4. Paris, S., et al., *Hair photobooth: geometric and photometric acquisition of real hairstyles.* ACM Trans. Graph, 2008. **27**(3): p. 30.
5. Jakob, W., J.T. Moon, and S. Marschner, *Capturing hair assemblies fiber by fiber.* ACM Transactions on Graphics (TOG), 2009. **28**(5): p. 164.
6. Chai, M., et al., *Dynamic hair manipulation in images and videos.* ACM Transactions on Graphics (TOG), 2013. **32**(4): p. 75.
7. Chai, M., et al., *Single-view hair modeling for portrait manipulation.* ACM Transactions on Graphics (TOG), 2012. **31**(4): p. 116.
8. Luo, L., H. Li, and S. Rusinkiewicz, *Structure-aware hair capture.* ACM Transactions on Graphics, 2013. **32**(4): p. 1.
9. Hu, L., et al., *Robust hair capture using simulated examples.* ACM Transactions on Graphics (TOG), 2014. **33**(4): p. 126.
10. Hu, L., et al., *Capturing braided hairstyles.* ACM Transactions on Graphics (TOG), 2014. **33**(6): p. 225.
11. Hu, L., et al., *Single-view hair modeling using a hairstyle database.* ACM Transactions on Graphics (TOG), 2015. **34**(4): p. 125.
12. Chai, M., et al., *High-quality hair modeling from a single portrait photo.* ACM Transactions on Graphics (TOG), 2015. **34**(6): p. 204.
13. Chai, M., et al., *Autohair: Fully automatic hair modeling from a single image.* ACM Transactions on Graphics (TOG), 2016. **35**(4): p. 116.

14. Zhang, M., et al., *A data-driven approach to four-view image-based hair modeling.* ACM Trans. Graph., 2017. **36**(4): p. 1-11.
15. Müller, M., et al., *Position based dynamics.* Journal of Visual Communication and Image Representation, 2007. **18**(2): p. 109-118.
16. Bender, J., et al. *Position-based methods for the simulation of solid objects in computer graphics*. 2013. Eurographics.
17. Macklin, M. and M. Müller, *Position based fluids.* ACM Transactions on Graphics (TOG), 2013. **32**(4): p. 104.
18. Macklin, M., et al., *Unified particle physics for real-time applications.* ACM Transactions on Graphics (TOG), 2014. **33**(4): p. 153.
19. Müller, M., T.-Y. Kim, and N. Chentanez, *Fast Simulation of Inextensible Hair and Fur.* VRIPHYS, 2012. **12**: p. 39-44.
20. Han, D. and T. Harada, *Real-time hair simulation with efficient hair style preservation.* 2012.
21. Kim, T.-Y., N. Chentanez, and M. Müller-Fischer. *Long range attachments-a method to simulate inextensible clothing in computer games*. in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2012. Eurographics Association.
22. Umetani, N., R. Schmidt, and J. Stam. *Position-based elastic rods*. in *ACM SIGGRAPH 2014 Talks*. 2014. ACM.
23. Rother, C., V. Kolmogorov, and A. Blake, *"GrabCut": interactive foreground extraction using iterated graph cuts*, in *ACM SIGGRAPH 2004 Papers*. 2004, ACM: Los Angeles, California. p. 309-314.
24. Baltrušaitis, T., P. Robinson, and L.-P. Morency. *Openface: an open source facial behavior analysis toolkit*. in *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*. 2016. IEEE.
25. Müller, M., et al. *Meshless deformations based on shape matching*. in *ACM Transactions on Graphics (TOG)*. 2005. ACM.
26. Jain, A.K. and F. Farrokhnia. *Unsupervised texture segmentation using Gabor filters*. in *Systems, Man and Cybernetics, 1990. Conference Proceedings., IEEE International Conference on*. 1990. IEEE.

27. Laurentini, A., *The visual hull concept for silhouette-based image understanding.* IEEE Transactions on pattern analysis and machine intelligence, 1994. **16**(2): p. 150-162.

28. Iben, H., et al. *Artistic simulation of curly hair*. in *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 2013. ACM.

29. Vanakittistien, N., A. Sudsang, and N. Chentanez, *3D hair model from small set of images*, in *Proceedings of the 9th International Conference on Motion in Games*. 2016, ACM: Burlingame, California. p. 85-90.

APPENDIX

VITA

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.