1. Introduction

This report documents the implementation of the Hair Swapping software written by Rafael Iriya.

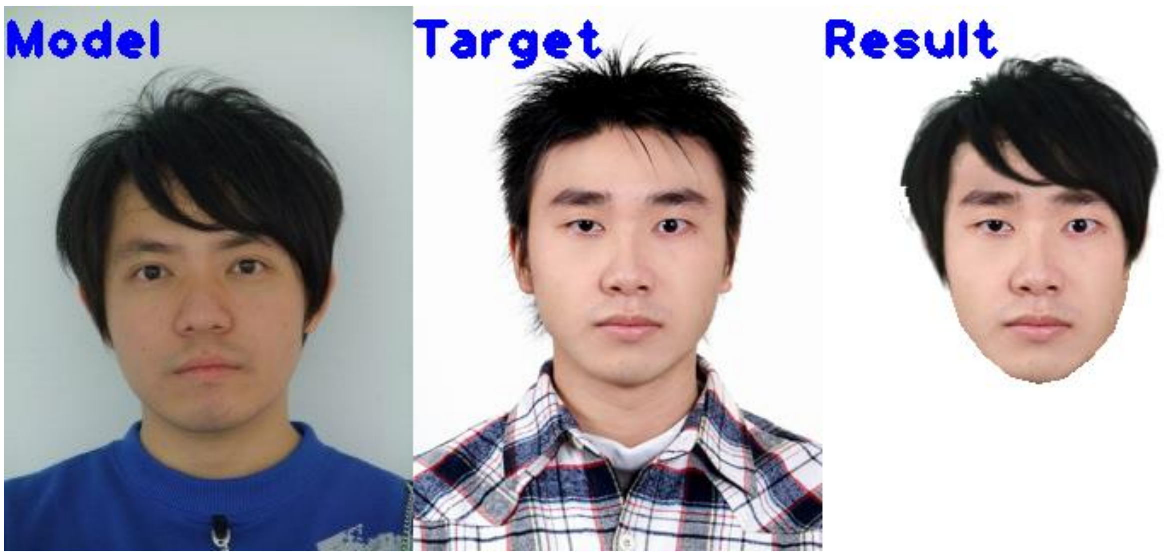
2. Scope

This project is an attempt to replicate the paper "Simulation of face/hairstyle swapping in photographs with skin texture synthesis" by Jia-Kai Chou and Chuan-Kai Yang [1]. However, due to time constraints and lack of clarity in some of the algorithms described in the paper, some design decisions had to be made, by either utilizing my own interpretation of the algorithm, a different algorithm, or choosing not to implement the algorithm.

My goal for this project was not to shoot for perfect results or struggle to obtain naturally looking images. My goal was to provide a complete solution for the problem of hair swapping, showing that:

i. I'm able to write a clean and efficient code in C++ using OpenCV and concepts of modularity and abstraction.
ii. I can follow instructions as well as take decisions and come up with alternative solutions when instructions are not completely clear.
iii. I have good algorithms skills, and have no problems implementing complex algorithms.
iv. I implemented enough routines to call it a hair swapping software, focusing on what I assume are the key points of the paper.

Having said all of this, the scope of this project lies in extracting the hair of a model image and placing it into a modified version of the target image. The modified image is basically the original target face with its hair swapped with the model's hair, in a uniform colored background.



Other decision made was to not implement algorithms whose code I could easily find an open source implementation. I believe this is close to a real project where we don't need to "reinvent the wheel". Some examples are the ASM face detection code and the matting procedure.

3. Projects:

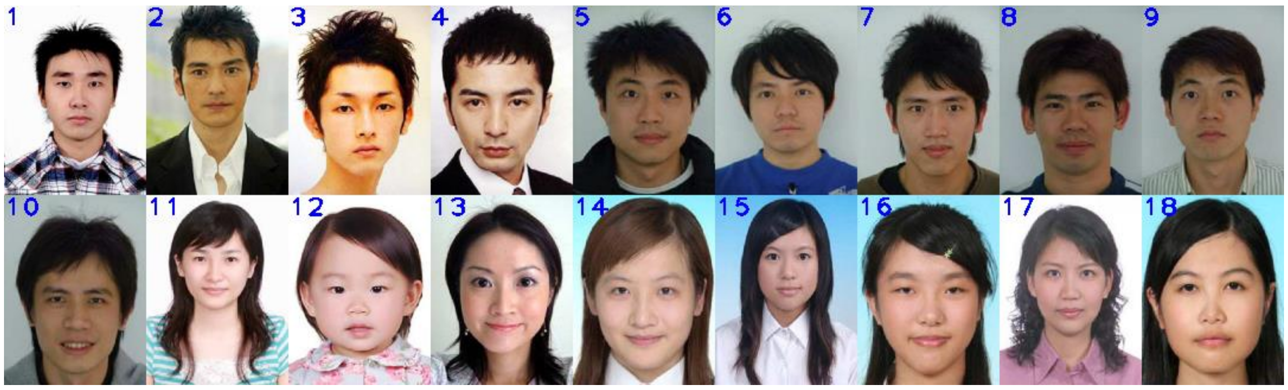
There are two separate projects that need to be compiled: Stasm and HairSwapping. Stasm is an open source software for face detection, which detects landmarks on the image, corresponding to facial key features. A custom main function was written to detect the keypoints and write the landmark locations in a .csv file. The HairSwapping project uses stasm.exe to detect the landmarks and reads them from the saved .csv file.

4. Tools:

The project was implemented in C++, using Microsoft Visual Studio 2017 and two versions of OpenCV, version 2.4.13 and version 3.4.0. Stasm.exe needs OpenCV version 2.4.13 or it will crash with a segmentation fault error on function detectMultiScale(). After searching on some forums, I found some other people who reported the problem occurs when later OpenCV versions are used. The HairSwapping project needs open cv 3.4.0, because it uses some features only available in versions after 3.3, such as the seamless cloning functions.

5. Database:

Some images from [1] were extracted, edited and scaled to be 320 x 240 each.



6. Description of the implemented algorithms:

6.1. Face detection:

The software STASM was used for face detection, just like in the original paper. The software STASM, provides key points for features in the face, which determines the boundaries of the face, and other parts like nose, mouth and eyes. For this implementation, a key assumption was made that the software is always going to return specific keypoints with the same index. For example, the left face edge is 0 and the right edge is 12. These points were a little different from what was mentioned in [1], however I followed the software's latest documentation.

These keypoints were used to determine the facial regions A,B,C according to [1] and using some arbitrarily defined offsets from the keypoints.

Point J was determined as 2 times the distance from chin to mouth, departing from the mouth, as mentioned in the paper.

The calculation of points I and K, according to the paper was very confusing. For example, the authors suggest the usage of arc A, to find point I, however arc A can only be obtained if you already know point I. My assumption is that the K-means must already have been performed and we need to follow the hair/skin edge. However it appears the X position needs to be the original position, calculated from points M and N. After this step, there is a very unclear step to determine the final position, calculating the tangents to the ellipses that connect the points. The authors suggest the tangents need to be aligned, but do not specify what this alignment would mean. All in all, it's very unclear why all these steps are needed when the authors state that this face boundary detection does not need to be that accurate. For this reason, I opted to do a simple implementation where points I and K are just a few pixels below point J, with X coordinates at ¼ and ¾ of the segment MN.

6.2. Hair Extraction:

The guidelines for obtaining the initial estimates for Hair, Background, Skin and Clothes were followed like [1].

For background and clothes, the mean of the top and bottom rows were respectively used.

For the hair initial estimate, the authors were not very clear what was the exact procedure, so the pixel from an arbitrary offset over the point J was used as initial estimate. This implementation is a little fragile, but worked fine for the tested dataset. A more robust approach would be comparing pixels below and above point J and calculate means on the two regions, and picking a point where the means are significantly different.

For the skin initial estimate, the entire face mask was used, including eyes nose and mouth. Since the majority of pixels are skin pixels, these other elements shouldn't contribute much. To further compensate the effect of those elements, the median was used instead of the mean, as the median acts as a sort of voting process.

A point of attention is the calculation of the clothes center. Following the exact guidelines from [1], for images where the clothes color were similar to the hair, or there were no clothes (picture starts from the neck up), the segmentation failed to produce good results, resulting in a bad extraction of the hair. This problem could be solved by including the position of the pixels in the K-means procedure and using a method to automatically detect the optimal number of centroids, like the Dunn's index or replacing K-means with DBSCAN or another clustering method.

For this reason, whenever the final clothes center obtained from K-means lied too close to the hair center, all pixels labeled as clothes were then labeled as hair, to prevent that hair pixels were labeled as clothes.

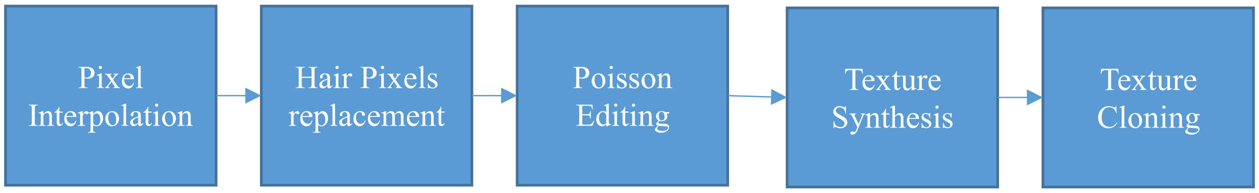
Since OpenCV implementation of K-means does not allow to use specific initial centroids, I implemented my own code. The early stopping criterion can be achieved with a hard equality, since the clusters only have integer values.

To extract the final pixels for the hair, the closest connected component right above point J was used, like suggested in the paper. However, the connected component constraint failed to produce good results in some cases, especially for long hair, where part of the hair is occluded by the neck or other part of the skin, breaking the connection.

For the matting procedure, I used an open source implementation by Kaiming, from the paper "A global sampling method for alpha matting" [2]. According to the authors, this implementation ranks very well in the alphamatting website.

6.3. Skin Synthesis

Skin synthesis was implemented according to [1], however I could not exactly replicate the algorithm from the paper, due to lack of information in many parts. For example, the paper does not describe exactly how side hair pixels are handled, how the Poisson editing is applied and how the final texture synthesis is performed. For this reason, I created a series of algorithms represented in the following block diagram:



6.3.1. Pixel interpolation

The procedure described in [1] was followed as closely as possible, where regions A,B,C where extracted and converted to CIELab, where an average chrominance was calculated and L was obtained by interpolation of pixels from the edge columns in A and C to the central column in B (brightest column). Besides the forehead, skin is also patched on the sides, before the eye region. Because some pixels in the reference columns were sometimes a little too dark, they were replaced with the average from the corresponding region, to avoid unnatural transitions in the linearly interpolated patch.

6.3.2. Hair Pixels Replacement

Since hair pixels usually have a completely different color from the skin, if the hair pixels in the face are not handled prior to the Poisson editing, the Poisson editing results in strong artifacts, and presents parts with a similar color from the hair, which is undesirable.

This is the most challenging part of the whole process, because the hair pixels need to be replaced with values as close as possible to the skin and any big transitions between the replaced values and the closest skin pixel are not handled by the Poisson editing.

Some ideas were attempted like generating random pixels values with the distribution of regions A and C, generating a texture from regions A and C, and using the linearly interpolated patch to cover the pixels. After extensive tests, the linearly interpolated patch showed the best final results.

To decide which pixels were being replaced, I tried two approaches: 1) covering up only the pixels with values close to the hair average. 2) using a pre-defined mask to select which pixels are always replaced. Approach number 1 ideally seems like a better approach, but it results in many uncovered pixels that are significantly different from both the hair and the skin, and those transitions are not handled by the Poisson editing. Approach number 2 presented better results overall, but in a few cases, the replaced pixels also showed a big transition from the closest skin pixels, generating unnatural transitions. The key to applying approach number 2 is that the mask for replacing the hair pixels cannot be exactly the same as linear interpolation masks. The Poisson editing procedure combines information from the source and the destination images, and if they are the same for some pixels, no changes are applied. For this reason, the mask for hair pixels replacement was defined to be a little smaller than the linearly interpolation mask. This replacement mask is shown in section 7.3.3.

6.3.3. **Poisson Editing**

The modified image obtained after hair pixels replacement is used as the destination image (**1**) for the Poisson Editing procedure. Another image with the linear interpolated patch cloned into the original image is used as the source image (**2**). However, only a region of interest that starts a few pixels above the eyebrows and extends to the last row with hair pixels is used for this task, to generate a more smooth transition between synthesized patch and the original forehead (**3**). OpenCV inbuilt SeamlessCloning() method is used for this task. The modified pixels in **3** are then cloned into **2** to combine the best of the linear interpolation and the seamless cloning **(4**).

6.3.4. **Texture Synthesis**

A simple approach inspired by [3] was implemented for texture synthesis. The original A or C region (whichever is larger) is used as a reference texture and divided into small 5x5 blocks. A greedy search algorithm selects the sub-blocks which will compose the synthesized texture, based on how similar are the edge pixels in each connecting sub-block. The first sub-block is chosen as the one that is closest to the average color of the reference texture. Then, the next connecting blocks are chosen with the greedy search algorithm, from left to right, top to bottom, based on how close they are to their left and top neighbors. Finally, to avoid large transitions between the connecting sub-blocks, a 5x5 median filter is applied to smooth the texture.

6.3.5. Texture cloning

The final step involves using Poisson editing once again to clone the entire synthesized texture into the image (4) obtained after the first Poisson editing process, resulting in the final synthesized face image.

6.4. Hair Swapping

The final step for hair swapping was implemented by first generating the hair and the synthesized face images by using appropriate masks, obtained in the previous steps. The connection point for the hair was calculated as the hair point closest to point J in the hair image. Then in the final image, the connection point is the same, except we sum this offset to this image's J point.

To calculate the best position for the scaled hair, it was unclear what the exact implemented algorithm in the paper was. It was also unclear how an iterative or greedy search algorithm would converge to the optimal solution. For this reason I opted to implement an exhaustive search algorithm, looking for all the combinations of scale and translation, within the given parameters. Two downsides of this approach were its high cost and the need to apply scale on X and Y separately, as scaling becomes tricky when X and Y have different scaling directions (one is scaled up and the other one is scaled down). The optimal scale is searched around the ratio between the size of the target head and the model head, calculated by the distance between points M and N.

To calculate the cost function for the energy, I implemented a search on each pixel in the contour of the face, up to the vertical position between mouth and nose. It searches for holes between skin and hair, moving up, left or right, depending on whether the pixel is a left edge, top edge and right edge. Because some pixels might fit into two different categories, a hashset was used to avoid double-counting hole pixels.

Finally, hair swapping was performed by alpha blending the hair matting image into the face image.

The paper also does not specify how to deal with the case of the hair covering all the holes, but also covering the face. For this reason, I implemented a second cost function, calculating the number of face pixels below the eye covered by hair pixels. This algorithm also allows some tolerance on the lateral edges of the face, which very likely can contain some hair. Also, the result looks more natural with some covered lateral pixels than with a hole.

Finally, the total energy can be calculated by a sum of the hole energy and the hair overlap energy. To allow flexibility, a weight parameter was developed to give a higher cost for holes than for overlaps.

7. Results

A complete framework was fully developed, inside the scope defined at the beginning of this document. However it is important to state that the parameters used in this implementation were not fully optimized, so it's not guaranteed that every combination of hair and face in the dataset will produce good results. The software might fail to produce natural results in some cases, but the developed algorithms were built on intelligent decisions, and produce results that are consistent with the implementations.

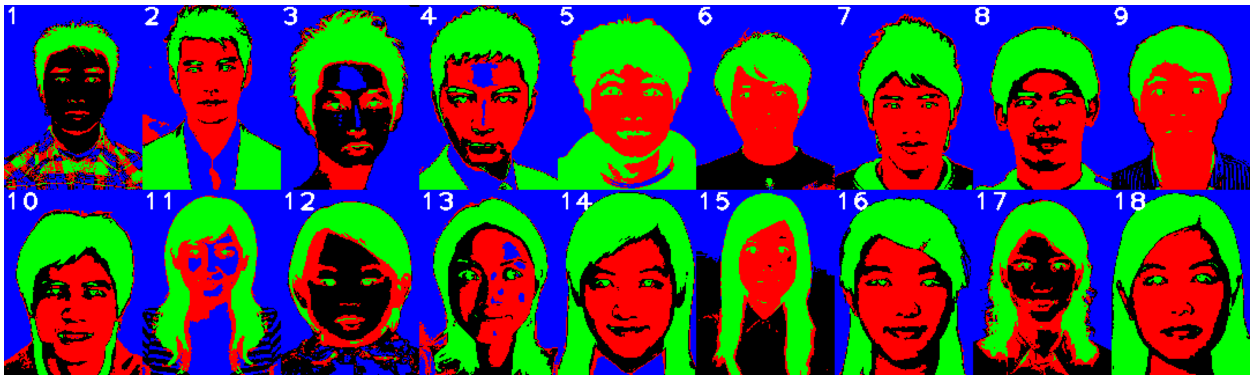
7.1. Segmentation Results:

Green = Hair

Red = Face

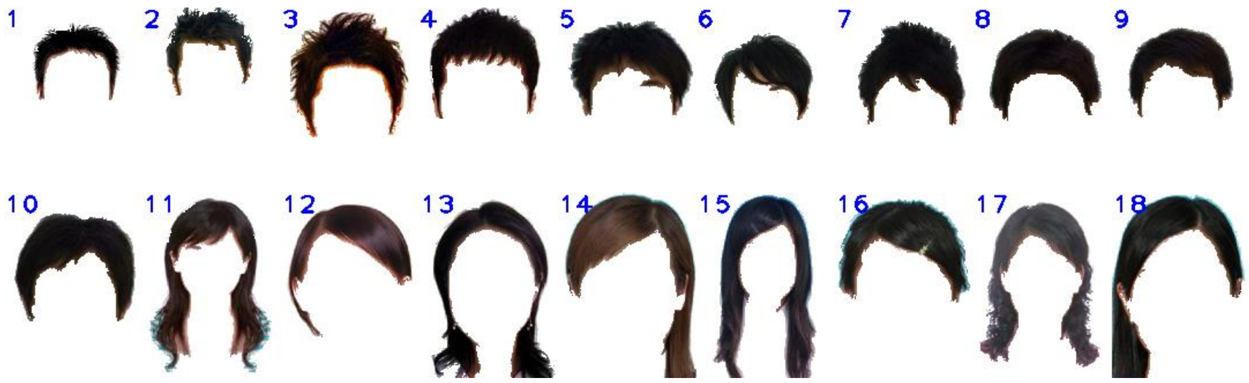
Blue = Background

Black = clothes

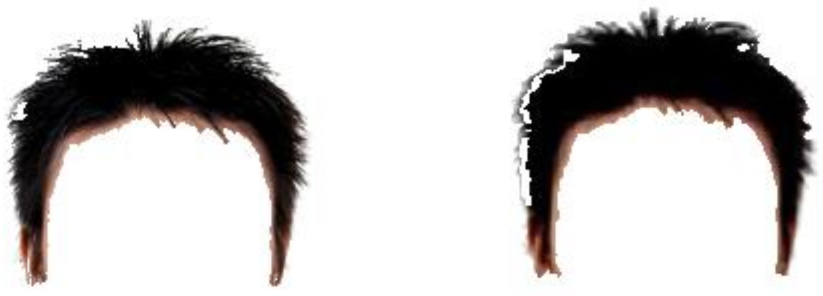


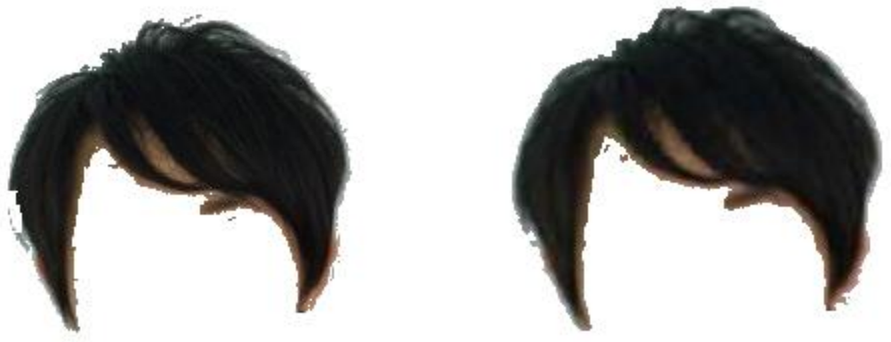
Segmentation was not perfect on improper initialization of the clothes centroid, due to not existence of clothes (3), multicolored clothes (1) or clothes with similar color to the hair (5). However since only the hair segmentation is necessary, these problems did not compromise the results in any way.

7.2. Hair extraction Results



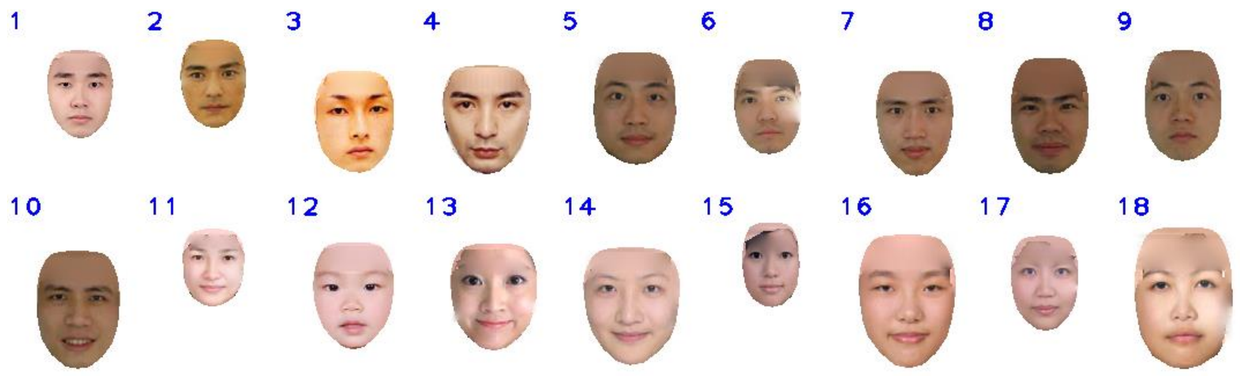
The final hair images don’t look as sharp as the ones shown in the paper, mainly because the images extracted from the paper had low quality and they were further interpolated to be 320 x 240 pixels. However I was able to obtain a few images from the author’s database, but unfortunately they weren’t many. The following images show the differences of using the original author’s images (left) and low quality images (right). We can see the images on the left have much finer details.



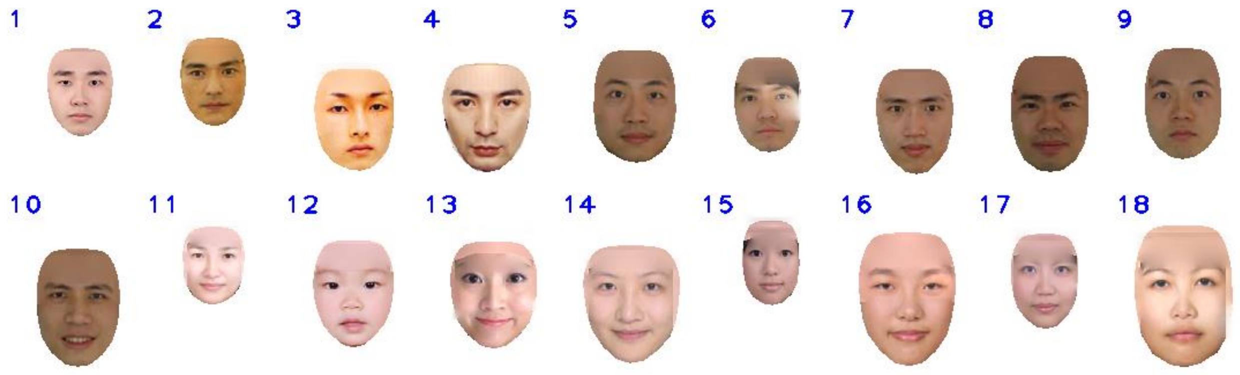


7.3. Skin Synthesis Results

7.3.1. Only hair pixels in the face were replaced with the linear interpolated patch

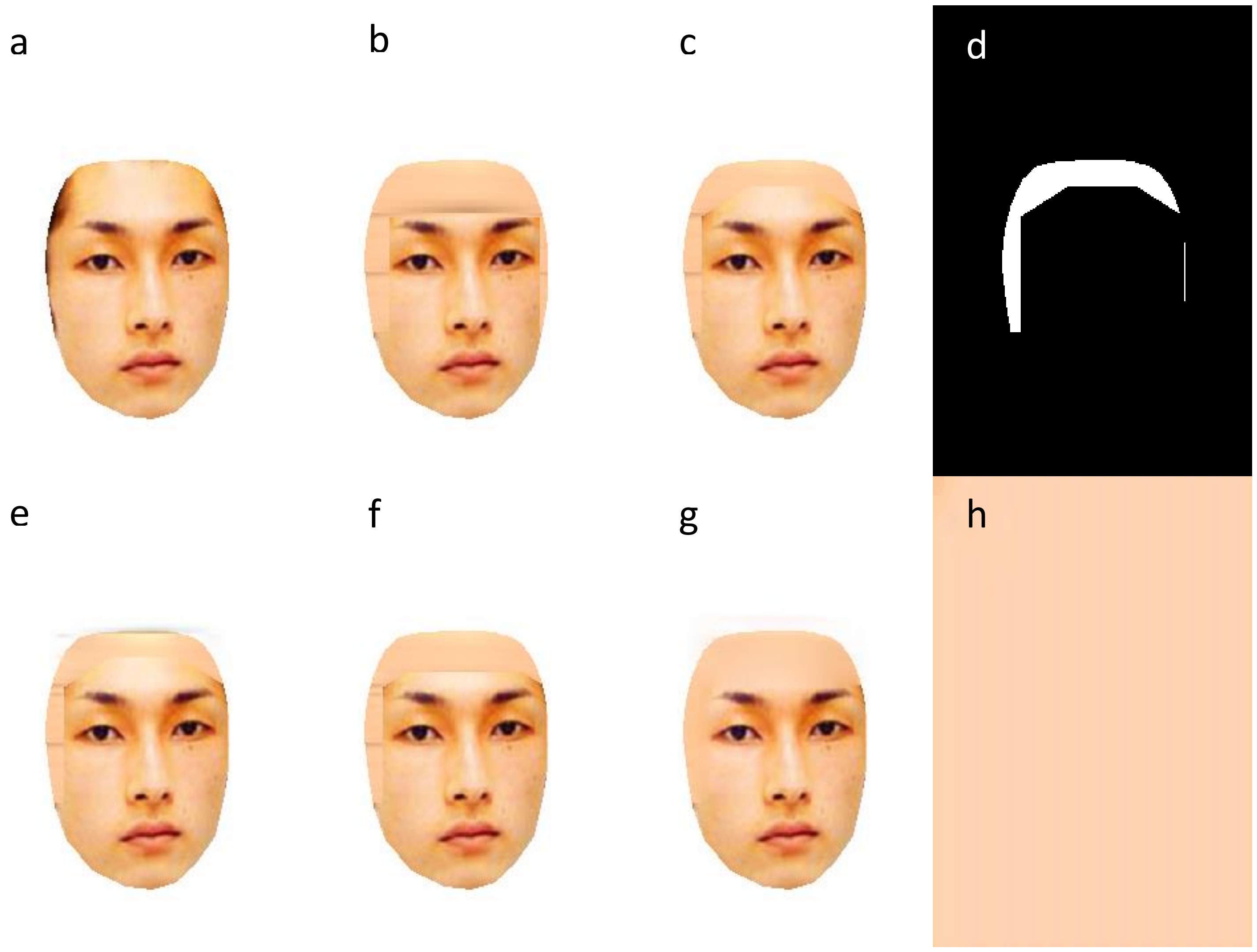


7.3.2. Predefined mask replaced with the linear interpolated patch



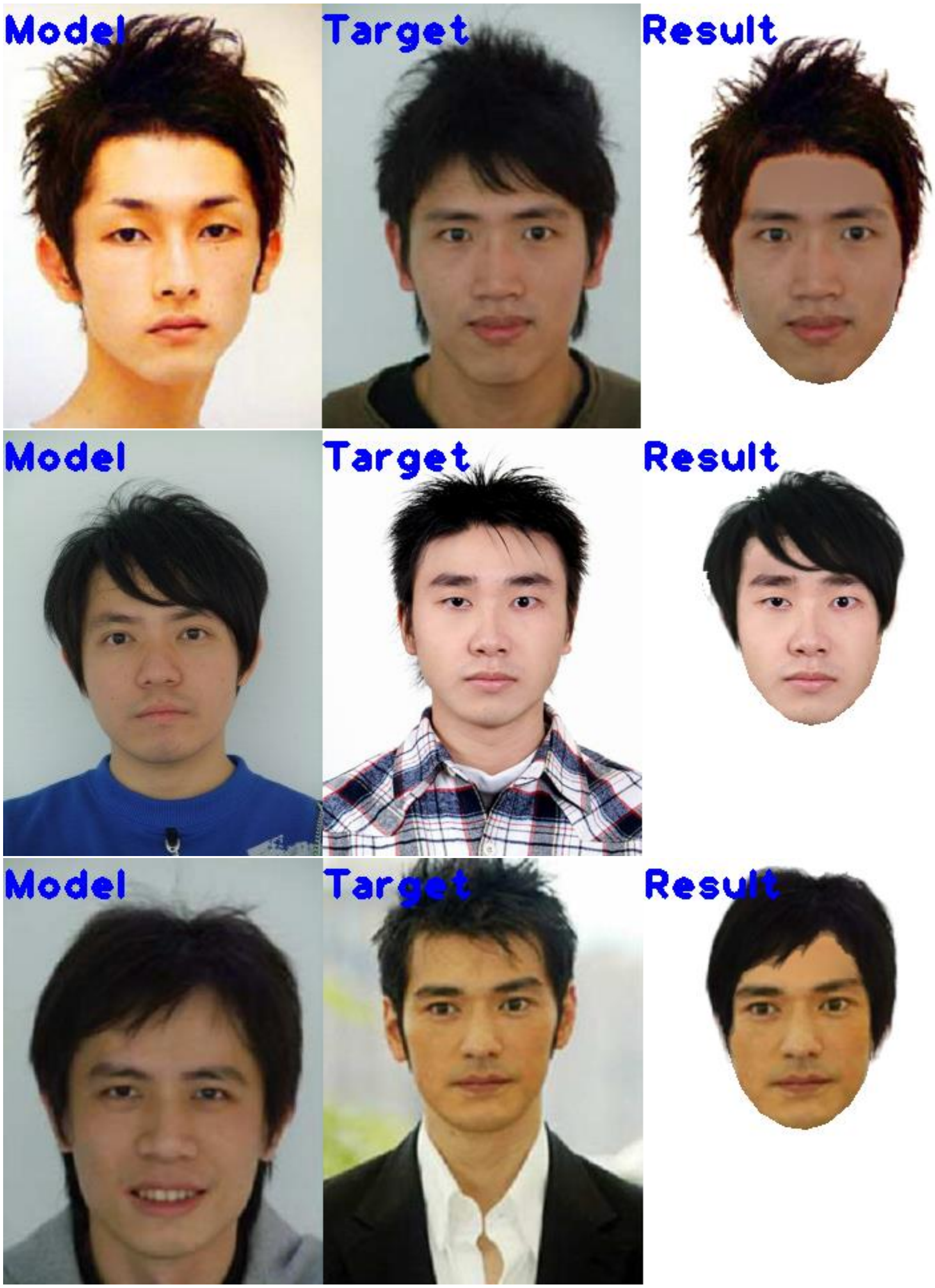
The skin synthesis process did not present natural looking results when hair pixels in the face were not properly substituted before the first Poisson editing procedure. If there is a big discontinuity in colors, even if it’s some transition pixels from hair to skin, the process fails to create a smooth transition. The same goes for when the hair is completely covered by replacing pixels. Inaccurate detection of the upper eyebrow point from Stasm also made difficult to completely cover some hair pixels in a few cases.

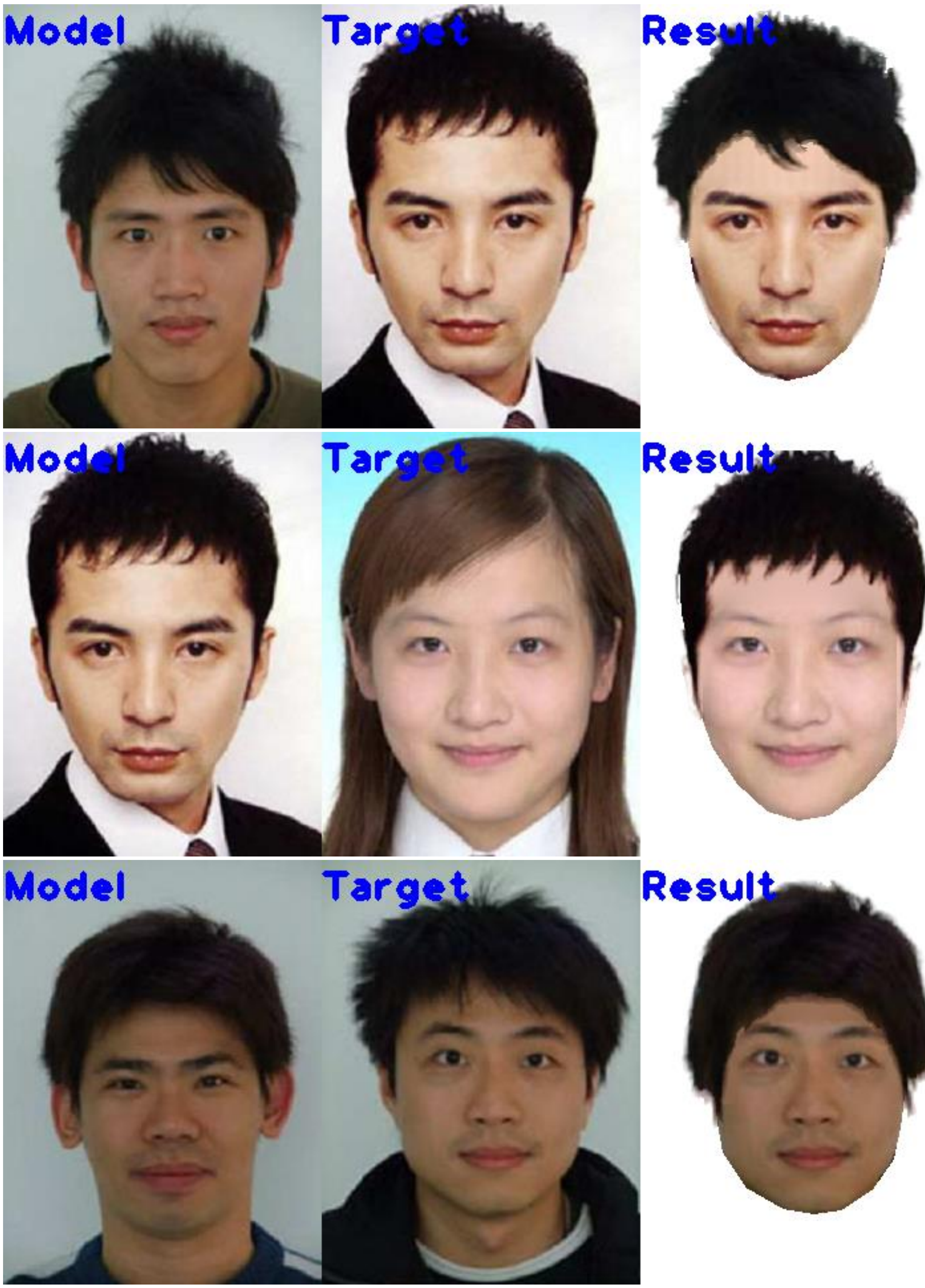
7.3.3. Skin Synthesis Process

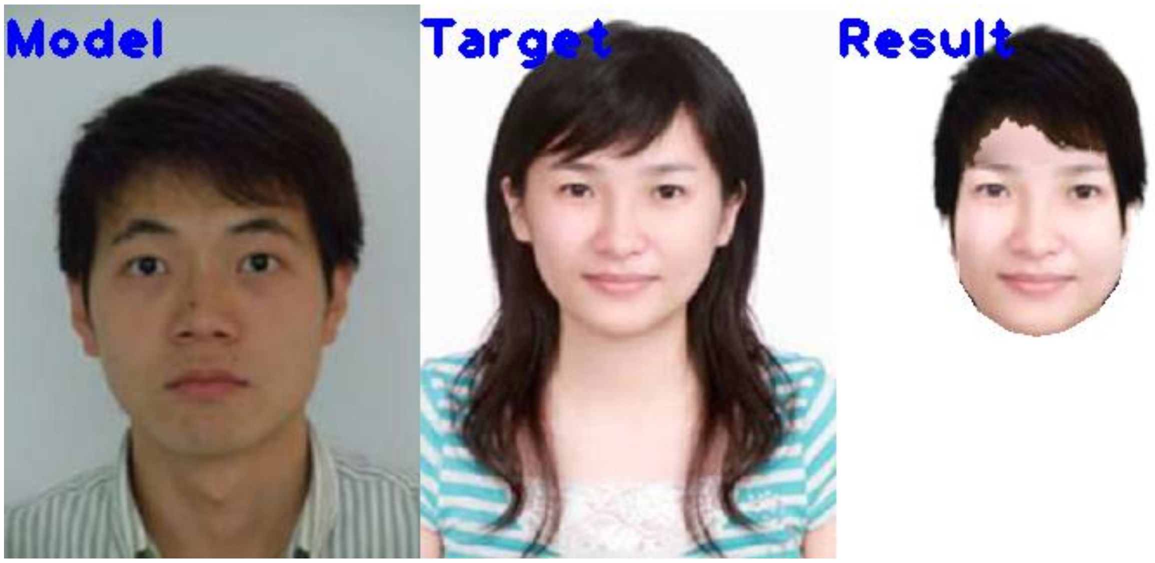


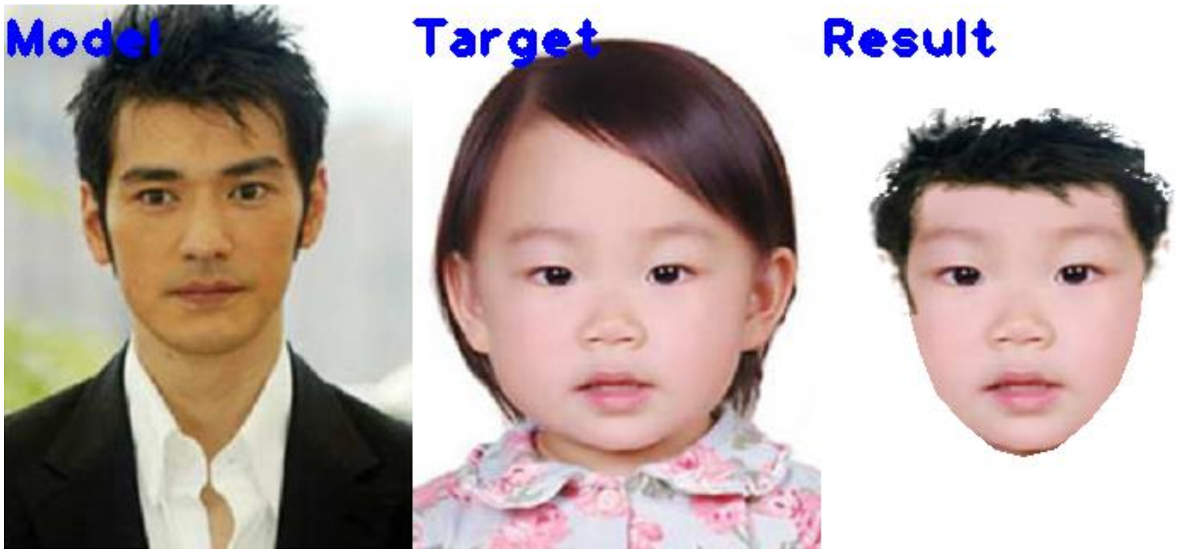
a) Original image. b) Linear Interpolation. c) Replaced Hair Pixels. d) Replacement mask. e) Poisson editing. f) Combined result of d and e. g) Final image with synthesized texture. h) Synthesized texture

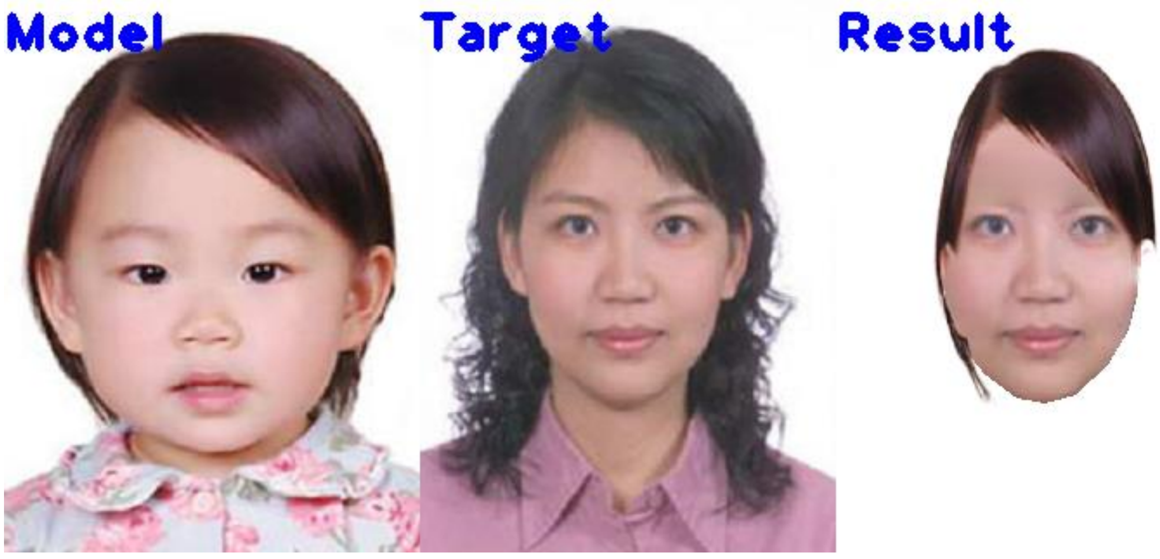
7.4. Hair Swapping Results

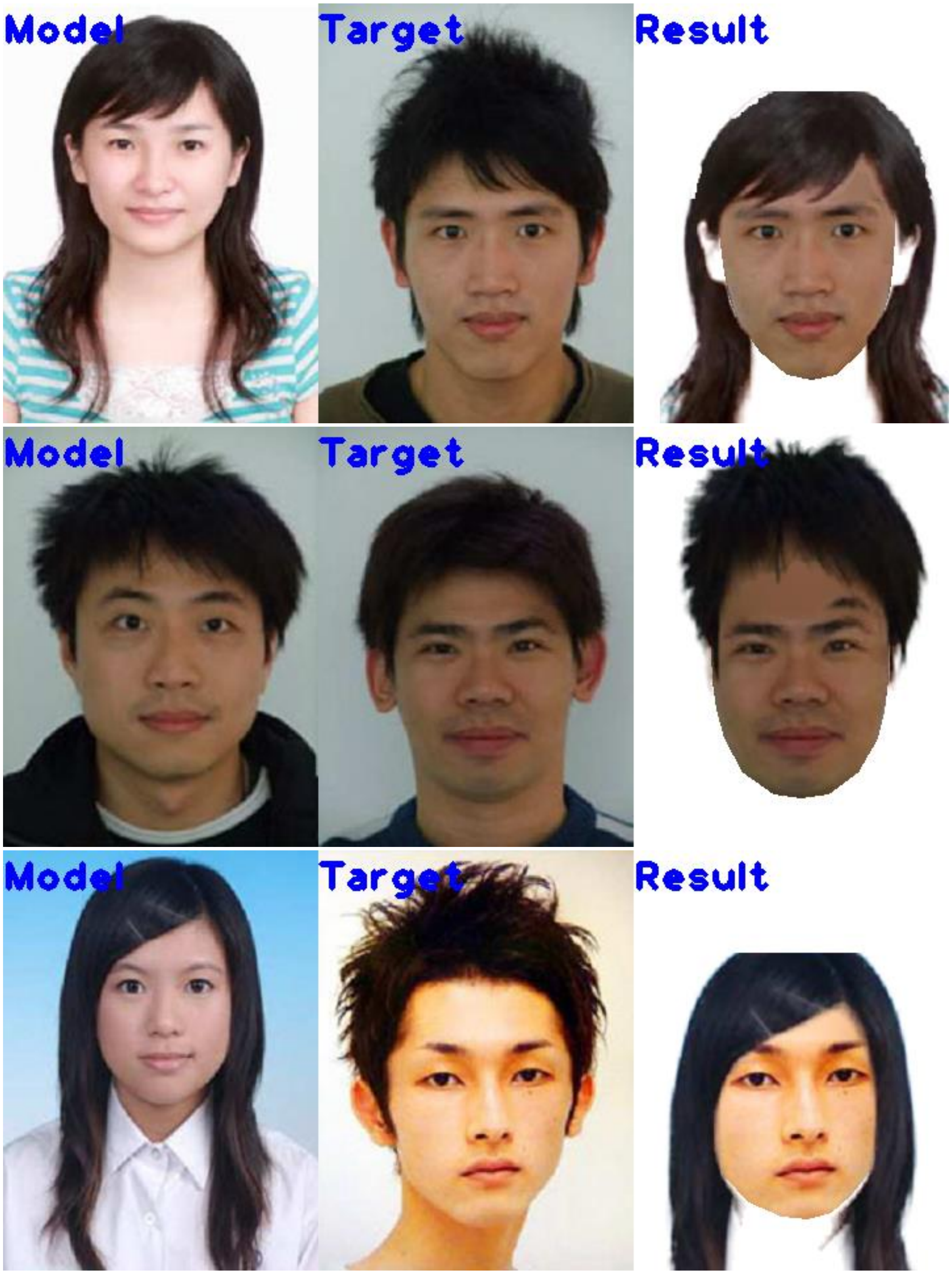


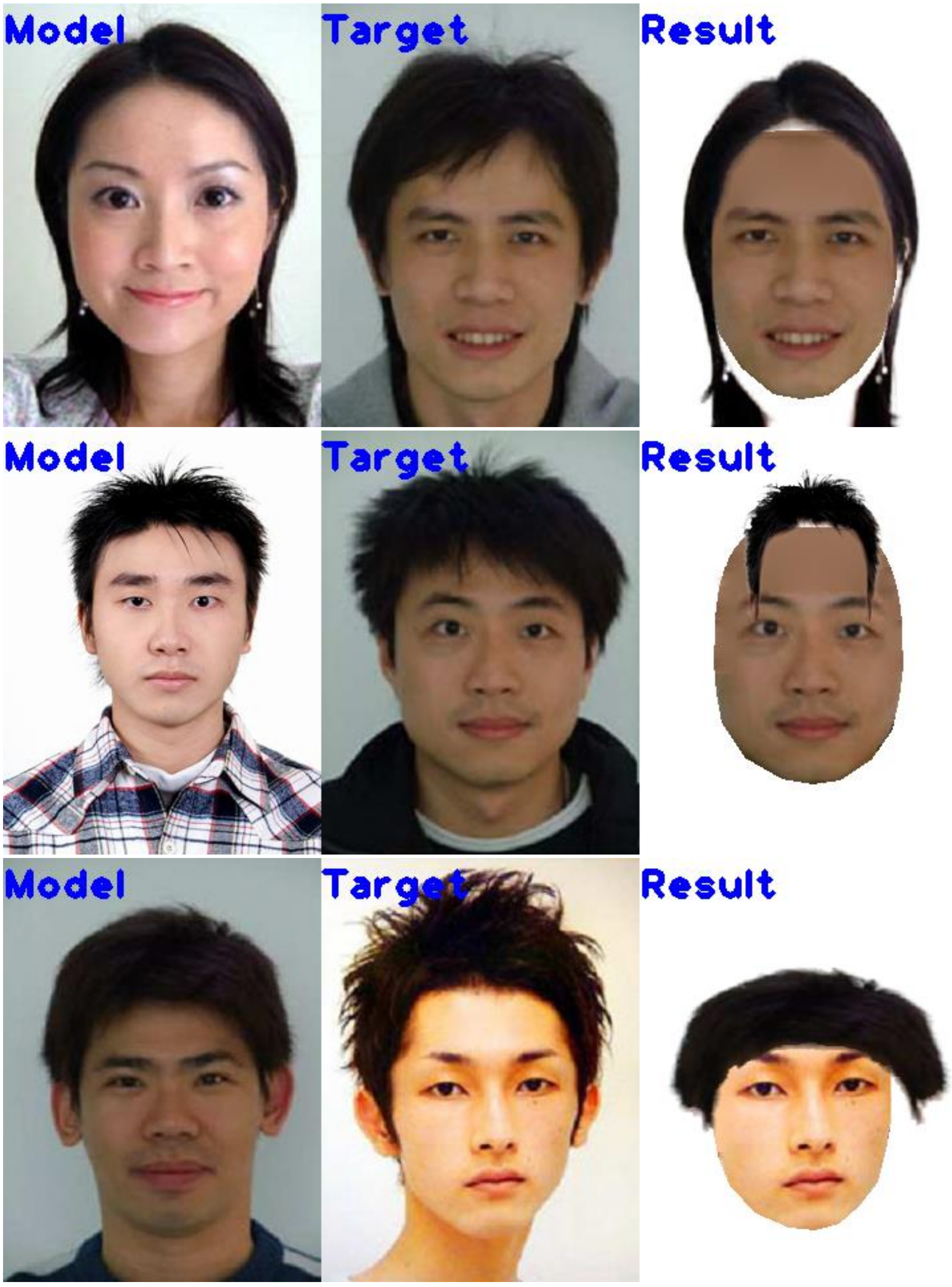

Model
Target
Result
Model
Target
Result
Model
Target
Result


Model
Target
Result


Model
Target
Result


Model
Target
Result


Model
Target
Result
Model
Target
Result
Model
Target
Result


Model
Target
Result
Model
Target
Result
Model
Target
Result

It is clear that for some images the search space for translation and scaling should be adjusted, as well as the weight factor for the hole energy vs hair overlap energy which could also be fine-tuned by extensively testing the code throughout several target and image combinations. Also, for some images, the swapped hair goes out of bounds and gets cropped. Color mismatches between the skin color on the model and the target resulted in some unnatural results, for the semi-transparent parts of the hair matting images. This problem could also be reduced with higher resolution images, which would improve the matting results. Some images also show unnatural results because the hair is attempted to be placed at the top of the head, but because it has bangs or hair in the forehead region, it should be placed below the top of the head. However, it is just obeying the cost function, which was not designed to handle this case, like on the original paper. Finally, the ear region is problematic for long hairstyles, because it needs reconstruction of the hair around that region. This should also be a problem in the original paper, even with ear synthesis.

References

[1] Chou, Jia-Kai, and Chuan-Kai Yang. "Simulation of face/hairstyle swapping in photographs with skin texture synthesis." *Multimedia tools and applications* 63.3 (2013): 729-756.
[2] He, Kaiming, et al. "A global sampling method for alpha matting." *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011.
[3] Efros, Alexei A., and William T. Freeman. "Image quilting for texture synthesis and transfer." *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001.