Introduction

This report documents the implementation of the Hair Swapping software given as part of my technical interview for Algomus, Inc.

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:

1. I'm able to write a clean and efficient code in C++ using OpenCV and concepts of modularity and abstraction.
2. I can follow instructions as well as take decisions and come up with alternative solutions when instructions are not completely clear.
3. I have good algorithms skills, and have no problems implementing complex algorithms.
4. 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.

Machinery:

The project was implemented in C++, using Microsoft Visual Studio and OpenCV version 2.4.13. Some bug with the Cascade detector prevented me from using the latest OpenCV version, which was generating a segmentation fault error on function detectMultiScale(). After searching on some forums, I found some other people who reported the problem was solved after rolling back to version 2.4.xx.

Database:

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

Description of the implemented algorithms:

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.

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.

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 connected component right above point J was used, like suggested in the paper. A hard offset from point J was once again used to determine the intersection point. 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. However, in my experiments matting did not significantly improve the hair extraction results, especially when large erosion structures are used (11x11 in [1]), so I kept the erosion structures small.

Skin Synthesis

Skin synthesis was implemented according to [1], 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). I was able to successfully implement the overall algorithm, however the results did not look similar to the ones in the paper for several reasons:

- The paper does not describe the algorithm in many details, so there might have been some differences in implementation
- I did not implement the Poisson editing technique for this simple reason: There is an implementation of the technique in OpenCV 3.0 and later versions, called Seamless Cloning. The only reason I wasn't able to use this is because I was using version 2.4.13 because of the aforementioned problems. Thus, seemed a waste of resources implementing such a complex algorithm.
- The final algorithm for generating a texture like figure 7d in [1] is not mentioned at all in the paper. Besides, in my opinion, figure 7c looks more natural.

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 be the same, except we sum this offset to this image's J point.

To calculate the hair mask after scaling the hair image, I noticed right away that we could not use a threshold of points above (0,0,0) as some of the hair pixels might actually be (0,0,0) if a person has black hair. So my next solution was to create a non-black background, with some very unlikely color for the hair, such as magenta. However I noticed that as an artifact of scaling, some of the background pixels are smoothed and become different than the color set for the background. The solution was to add an alpha channel to the unscaled hair image, and set the unscaled mask as the alpha channel. That way, it is possible to both use (0,0,0) for the background and avoid the smoothing effects.

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).

To calculate the cost function for the energy, I implemented a search on each pixel in the contour of the face, and searching for holes between skin and hair, moving up, left or right, depending on

whether the pixel was 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.

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.

Final considerations

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.

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.