Automatic Makeup and Facial Beauty Enhancement

Tianyuan Zhang, Mao-Chuang Yeh, Xingyu Xiang, and Xiaoxuan Gu
College of Engineering, University of Illinois at Urbana-Champaign

1 Introduction

As the increasing popularity of smart phone among young people, we tend to record and share our daily life on social media with selfies taken by cell phone. This kind of social trend encourages the creation and development of smart phone makeup apps. Our project is inspired by one of the apps called makeup plus which enables users to apply the effect of eyeshadow, lipstick, and blush to a portrait. Our objective is to understand the methodology behind all these effects and simulate it on laptop with the assistance of Matlab.

2 Methodology

2.1 Method I:

The main idea of our first method is to start with key point detection of facial features. Then, an appropriate sized box is automatically generated according to the key points, and we cut these parts out by segmentation techniques. Finally, we do color shift or filtering with these parts and blend them back to the original image.

Cascade Object Detector: For facial feature detection, we first use a cascade object detector to detect the outline of the face region based on Viola–Jones Algorithm [4]. Cascade object detector operates in 4 steps as follow: (i) Haar feature selection, (ii) creating an integral image, (iii) Adaboost training, (iv) Cascading Classifiers.

Figure 1: Left: Original image; Right: Detected region and points

Face Feature Localization: The output of cascade object detector gives an approximate location and scale of the face. Then, we want to locate all facial features in the detected face region. Nine facial features are located: the left and right corners of each eye, the two nostrils and the tip of the nose, and the left and right corners of the mouth. The appearance of each facial feature is assumed independent of the other features and is modeled discriminatively by a feature/non-feature classifier. For feature position searching, tree-structured covariance is applied which increases the efficiency of distance transform methods [2].

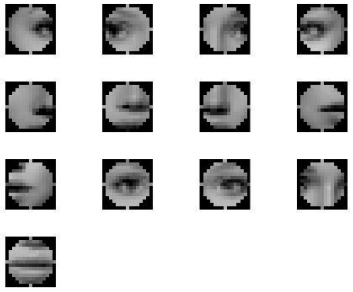

Figure 2: Detected facial features

K-Mean Segmentation: What is required in this method: (i) L*a*b* color space, (2) K-means clustering, (iii) Statistics and Machine Learning Toolbox. K-Mean segmentation [citation] operates in following steps:

Step 1: Using Imread to import image as usually, for the convenience of the implementation, we already cropped a square shape of the face. Of course, the square is cropped from the image based on the detected key points on the eyes.

Step 2: Having a shift from RGB Color Space into L*a*b* Color Space (L: luminosity layer; a: red-green axis; b : blue-yellow axis). In order to do this shift: (i) the makecform and applycform functions are used, (ii) using the Euclidean distance metric to measure the difference of two colors.

Step 3: Using K-Means Clustering to Classify the Colors in 'a*b*' Space. Specify the number of clusters to be partitioned, such as “nColors = 4”. Specify a distance metric to quantify how close two objects are to each other. This is a way to separate colors into different groups. Using “Kmeans” function.

Step 4: Label Every Pixel in the Image Using the

Results from Kmeans. Kmeans returns an index corresponding to a cluster. Every pixel will be labeled with the index of cluster.

Step 5: Create Images Segment the original Image by Color. After each pixel get the label, the objects can be separated by color. For example, if "nColors = 4 " has been set, it will result in three images.

Step 6: Segment the eyes/ mouth into separate image. The index of the cluster containing some color's objects should be arbitrarily decided, for Kmeans will not return the same cluster_idx value every time.

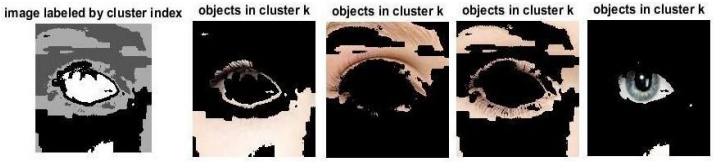

Figure 3: K-Mean segmentation process

Although the segmentation of eyes seem good, the output of the mouth cannot make us satisfied, for the mouth has an area which reflects the light and the color of that areas is similar with the skin. As the consequence, the lips and the surrounding skin cannot be separated perfectly. The following picture is the consequence, and we decide to adapt other strategy to do the implementation:

Figure 4: K-Mean segmentation applied to mouth

Grabcut Segmentation:(alternative segmentation method) Grabcut [3] is an algorithm was needed for foreground extraction with minimal user interaction. Grabcut segmentation can be operated in the following steps. Step 1: Define graph – usually 4-connected or 8-connected. Step 2: Set weights to foreground/background – Color histogram or mixture of Gaussians for background and foreground . Step 3: Set weights for edges between pixels. Step 4: Apply min-cut/max-flow algorithm. Step 5: Return to step 2, using current labels to compute foreground, background models.

Figure 5: Grabcut segmentation process.

2.2 Method II:

The weakness of method I as discussed above is apparent. Using segmentation by Grab cut [3] or clustering doesn't result in very decent outcome of the landmark feature. Even though the cropped images are small enough that only contain the preferable regions, segmentation procedures do not perfectly cut out the wanted region of face feature. This failure may due the nature of these segmentation algorithms which depend mostly on color gradient and edge. Yet for most of our wanted features (lips, eyes), the separation between them and the skin nearby do not have very sharp gradient differences.

So we shift our focus to finding a more accurate facial landmark detector that output more key points for each facial feature. We found Chehra [1], a very promising and robust facial detector that produces 50 key points. Chehra is created base of a Parallel Cascade of Linear Regression algorithm. Below is a comparison of key points generation using OpenCV face detector and Chehra:

Figure 6: Left: Cascade detector; Right: Chehra Detector

We take advantage of the Chehra key points detection and apply blending and color shifting for each facial features. We first generate masks by the given points and a gaussian filter is applied to each mask to eliminate edge effect. Especially for eyeshadowing, we need to subtract the original mask of the eye to preserved the color of eye itself. Following figure show the mask of each feature:

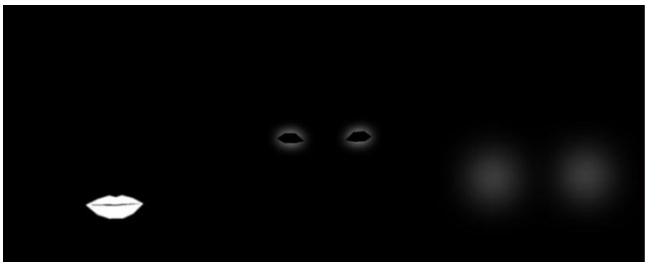

Figure 7: Masks of mouth, eyes, and cheeks.

The final result is much better than using

segmentation and clustering. The output is robust to face from different angle due to the performance of the Chehra key points detection algorithm. Following is the comparison of an image before and after putting on makeup:

Figure 8: Left: Original image; Right: Final result

3 Enhancement

After completing the general makeup effect, we would like to do some enhancement, such as changing the shape of eyebrows. In our project, we use morphing techniques to realize this function. The morphing process operates in the following 3 steps:

Step 1: Define corresponding points and shapes:

First, we need to detect the key points of eyebrow; The we have a eyebrow function to get a proper shape according to the key points.

Step 2: Shift defined shapes:

In order to get the morphing image of the eyebrow, another 'shift' function needs to be established.

Step 3: Remap surrounding areas:

If we want the morphed eyebrow can fit the original skin, the areas need to be modified to adjust the new image. To be more specific , if you have changed the shape of eyebrow, then the upper and lower skin need to be remapped based on a arbitrary ratio. Otherwise, the picture will look like fake.

Step 4: In order to get a natural entire image, do a triangular transformation on the end of eyebrow.

Here are the results of eyebrow morphing:

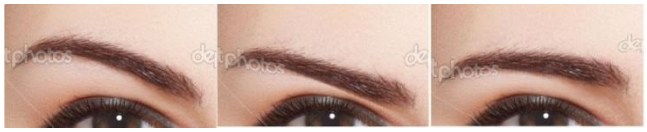

Figure 9: Left: Original image; Middle: Morphing effect 1;Right: Morphing effect 2

4 Final Results

For quick automatic makeup, we create a user-friendly interface where users can take a photo with laptop camera and import it so that they can do makeup upon their portraits.

Here are other results of our project:

Figure 10: Left: Original image; Right: Final result

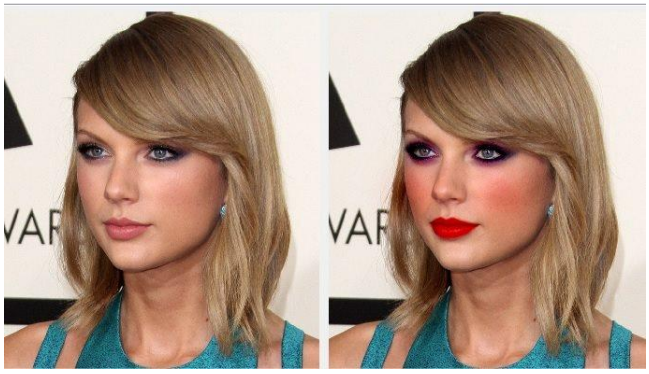

Figure 11: Left: Original image; Right: Final result.

References

[1] Asthana, A., Zafeiriou, S., Cheng, S., & Pantic, M. (2014, June). Incremental face alignment in the wild. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on (pp. 1859-1866). IEEE.

[2] Everingham, M., Sivic, J., & Zisserman, A. (2006, September). Hello! My name is... Buffy''--Automatic Naming of Characters in TV Video. In BMVC (Vol. 2, No. 4, p. 6).

[3] Rother, C., Kolmogorov, V., & Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. ACM Transactions on Graphics (TOG), 23(3), 309-314.

[4] Viola, P., & Jones, M. J. (2004). Robust real-time face detection. International journal of computer vision, 57(2), 137-154.

[5] Color-Based Segmentation Using K-Means Clustering. http://www.mathworks.com/help/images/examples/color-based-segmentation-using-k-means-clustering.html