Final Project: Face Replacement

(Option 2)

Computer Vision

CIS 581

Zhengxuan Wu, Yangyang Zhou

Department Of Computer Science

University Of Pennsylvania

Abstract:

This is a MATLAB based programming project focusing on human-face detection and human-to-human faces swapping. This project is adapting on open-source libraries on-line and previous projects during this semester for this class for completing the whole task, specifically including Face++ online deep learning face landmarks detection, image morphing, Poisson blending, etc.. Finally, motion compensation is introduced in this project in cancelling the shaking noisy of the face warping results. In conclusion, this project successfully complete the face swapping on easy test-set and middle test-set in the end.

Introduction

This project is a result-orientated programming project. We first divide the whole project into 6 different phases. As shown in the following flowchart, we start with 'Face Detection'. After comparing two online libraries, the Face++ method was adopted to detect feature landmarks on replacement faces and all target faces videos as well. It is quite desired to get landmarks for regions including eyes, nose, eyebrow, mouth and cheek. Next, as landmarks have been obtained, they are treated as the control points to realize TPS morphing. After morphing, a mask around the region of face is generated via calculating the boarder points of landmarks. Thus, within the mask region, blending can be applied. In order to achieve better results, we have performed two consequent blending. The first method is 'equalization with histogram', and the second one 'Poisson Blending'. At the end, motion compensation is achieved using simply linear interpolations between neighborhood landmarks. All concerns in steps described above have been presented with intermediate results. Presentation slides have been attached for reference.

Method

Face Detection & Landmark Detection

The goal of this portion of the project is to auto-detect the face(s) and the landmark points on the face(s). Couple methods were used when we initialized our research on this part. We first used code-book and SIFT feature detection library by Vedaldi. However, the detection was slow and inaccurate, and we need to define the face(s) which we want to detect before-hand which will cause trouble in blending later. We also tried the most popular face-landmark detection source by Prof. Zhu, X., and Prof. Ramanan, D.. at University of California in *Face Detection, Pose Estimation and Landmark Localization in Wild*. However, we found out that the method was slow and sometimes inaccurate even in small angle of face-turning. However this method can detect most of the faces in all of the videos. We finally came down using the Face++ online based deep learning method of detecting our faces and landmarks. Our pictures are first uploaded to the API of the Face++, and the feedback was then downloaded from the server which including face landmarks information and so on. This maximum angle distortion of the face was about 60-70 degrees for this method. However, in normal cases, Face++ pertain a very high accuracy, and

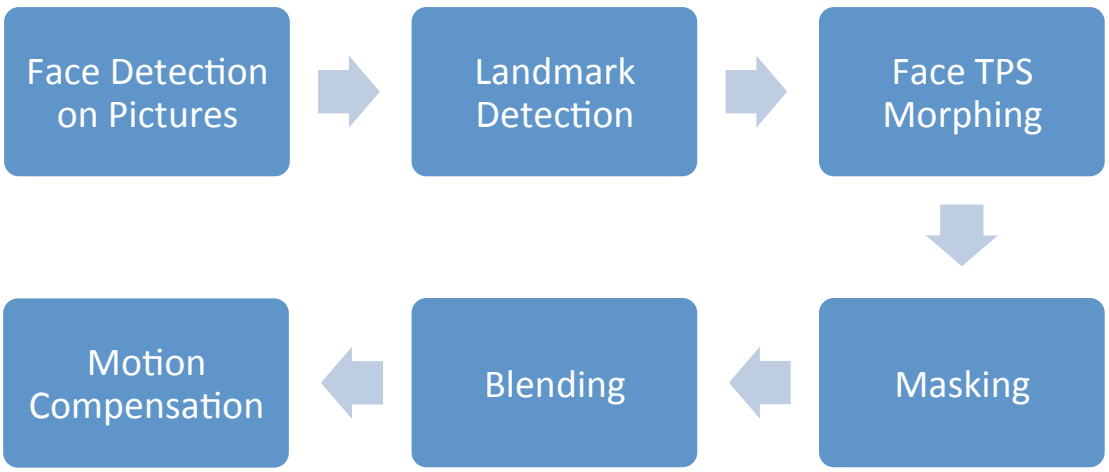
efficiency. Thus, we finally used Face++ as our source code for detecting the landmarks for the faces.

Face TPS Morphing

TPS Morphing was adapted from the code from the second project of this class. TPS morphing is commonly used in face morphing because it take into account the shape context and a lot of biology stuffs thus becoming very trustworthy for this project. The corresponding point in TPS morphing was generated by the landmark auto-detector Face++ using the method introduced on the first part. We clearly know that the TPS morphing is based on the following formula,

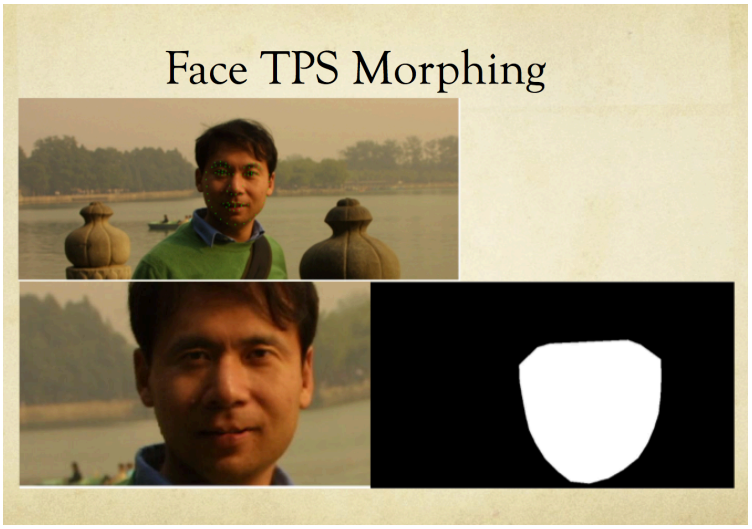
$$f(x,y) = a_1 + a_x x + a_y y + \sum_{i=1}^{p} w_i U\left(\|(x_i,y_i)-(x,y)\|\right)$$

Thus, the post-morphed points are know, so as the pre-morphed points in the original pictures. The only thing we need to do in the MATLAB is to solve the matrix function for those un-known coefficients. After the coefficients are calculated, we can then use back-ward interpolation for all of the points in the face region, and then copy the pixel values back to the current images.



Masking

One of the most important tasks in face morphing is to choosing the mask for blending. As it mentioned before, we firstly started with pure rectangular masks to blend the face using the built-in face detection function in MATLAB. However, the edges are really hard to eliminate out. And the we started to use those landmarks. Returning by the Face++, we have the coordinates of all the landmarks. We then use convex-hull function built-in in MATLAB to produce a very preliminary face region mask by adding a simple Gaussian filter as shown below.



However, we can see that the forehead region, and some part of the cheek are missed out. In other words, the blending face region is not a complete face region which will lead to information loss. However, there is a paper wrote by students from Stanford University, Kim, I., Shim, J., Yang, J. called *Face Detection,* which introduced a color based face region detection method. Basically, they mentioned that face are very sensitive to a certain color pixels. We first transfer the RGB color based picture into YCbCr components by using the formula below,

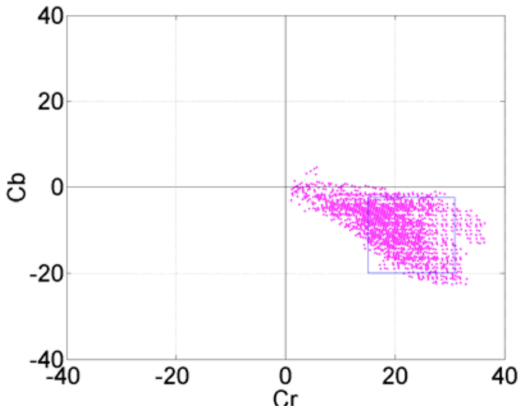
$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

$$Cb = -0.169 * R - 0.332 * G + 0.500 * B$$

$$Cr = 0.500 * R - 0.419 * G - 0.081 * B$$

In the YCbCr color space, the luminance information is contained in Y component; and, the chrominance information is in Cb and Cr. Therefore, the luminance information can be easily de-embedded. In the skin color detection process, each pixel was classified as skin or non-skin based on its color components. The detection window for skin color was determined based on the

mean and standard deviation of Cb and Cr component, obtained using 164 training faces in 7 input images. The Cb and Cr components of 164 faces are plotted in the color space in Figure below.



And then, based on practical training data, we found out that the thresholds from Cd and Cr for certain faces. The next step is to separate the image blobs in the color filtered binary image into individual regions. The process consists of three steps. The first step is to fill up black isolated holes and to remove white isolated regions which are smaller than the minimum face area in training images. The threshold is set conservatively. The filtered image followed by initial erosion only leaves the white regions with reasonable areas as illustrated in Figure below which is an multi-faces example.

We then applied this to the original mask, and here are the results. We can clearly see that the forehead portion is being added successfully. However, a small portion of ear part is being added as well.

Blending

The main goal for blending is to diminish the intensity difference between two pictures or faces, and the contrast differences. On the other hand, a good blending will improve the skin texture, and solve any shadow problem that is brought by the selected pictures. However, after the face masks are improved, blending actually became relatively straightforward. The Poisson Blending method are adopted from online library directly. The objective of this "Poisson Blending" algorithm is to compose a source image and a target image in the gradient domain. The reason "Poisson Blending" achieves a more realistic looking composition than naively pasting two similarly colored images together is because the human visual system is more sensitive to contrast than intensity values. Below is an example of how the blending works for our test-set. Clearly, Poisson Blending works out the best result.

Motion Compensation

There are a couple of imperfect aspects for our primary output videos shown. The most distinguished problem is the shaking. For each frame, it is impossible for landmarks to track all face features as the exactly the same position. For example, the landmark indicating the apex nasi is 1 pixel away from real apex nasi in one frame, but for next frame, there might be 2 pixels away. Therefore, the unstable nature of landmarks in different frames causes the shaking in the video flow. In addition to previous hard work, we have tried two methods of motion compensation in order to improve visualization. And the first one, 'Intermediate Landmarks' is adopted at the end, and the other method is being proved to be feasible for future research conducts.

One is the comparatively simple way, called as 'Intermediate Landmarks'. Intermediate landmarks position is considered as the weighted intermediate value between three frames: the previous, current and following frames.

Landmark= 0.25* landmark_previous + 0.5* landmark_current + 0.25* landmark_follow

Remarkably, the output video runs smoother. Large degree of shaking has been removed. However, the face expression after replacement may not be as accurate as before. Finally, we choose to retain this result (motion compensation with intermediate landmarks) as our final result, simply because it creates the most comfortable vision sense.

The other method is quite complicated, named as 'Motion Vectors Estimation'. Here is the brief explanation of Motion Vectors Estimation. Firstly, in mask region of target faces, pixels are

selected based on a block parameter. Here, block value of 5 is chosen so that for image is divided in a series of 5x5 blocks. Secondly, block movements are calculated as the indicators of frame changes between the current frame and the next one. Logarithm behind is that find the location of every blocks in the following frames, based on block property matching. Movement dates are saved in motion vectors. Thirdly, landmarks are grouped into corresponding blocks. After assign motion vectors data of each block to landmarks, we are able to know how much the landmarks move to the next frame in horizontal and vertical directions respectively. Finally, landmark movement can be controlled instead of automatically detected in the previous work.

As explained, this method only requires the detection of landmarks for the first frame. Later, landmarks are moved based on motion vectors. Thus, the shaking problem has been completely solved.

Figure 1. Motion Vectors

The left the current frame and the right one the following frame. Blue arrays within mask area are motion vectors showing the block movements between frames.

Figure 2. Landmark Movement

Green dots are landmarks for the first frame. Red dots are landmarks for the current frames. The background image is the current image.

Figure 3. Failure of Motion Vectors Estimation

(Landmark after movements does not match with the face features.)

Finally, this method is not adopted, because it causes the problem 'imperfect morph',

which outweighs the shaking problem. It is hard to guarantee the accuracy of Motion Vector calculation. As long as there is a mismatch for finding corresponding blocks in the next frame, the method fails. By comparison, Face++ detect landmarks more accurately than the Motion Vector calculation.

Results & Conclusion

The initial video test results on easy and middle set are pretty good. Although some shaking of face is involved, the overall quality is very good. The results are included in the final project file, please go there for more information. Besides, the shaking, one of the most difficult tasks is to tackle the face orientation problem. In the hard test, when the face turns around to over 70 degrees, face detection failed. However, this problem might be very easy to solve by using a secondary library. Due to the limit of time, we have not solve this issue yet, but the plan is pretty clear which is including a back-up library for face and landmark detection. One of the back-up options will be using the landmark detection mentioned above by Prof. Zhu from University of California, Irvine.

Reference

1 Face++:

Link: http://www.faceplusplus.com

2 Face Detection, Pose Estimation and Landmark Localization in the Wild (X. Zhu, D. Ramanan)

Link: https://www.ics.uci.edu/~xzhu/face/

3 Poisson Blending

Link:http://www.mathworks.com/matlabcentral/fileexchange/45799-poisson image-editing

4 Motion compensation:

BlockMatcher

http://www.mathworks.com/help/vision/ref/vision.blockmatcher-class.html

5. Face Region Detection On Color Based Method

Link: https://web.stanford.edu/class/ee368/Project_03/Project/reports/ee368group02.pdf