CMSC 733, Computer Processing of Pictorial Information
Project 2: Face Swap!
Due on: 11:59:59PM on Wednesday, March 08 2017

Prof. Yiannis Aloimonos,
Nitin J. Sanket

February 23, 2017

The aim of this project is to implement an end-to-end pipeline to do swap faces in a video just like Snapchat's face swap filter https://www.snapchat.com/ or face swap website http://faceswaplive.com/. It's a fairly complicated procedure and variants of these have been used in many movies. One of the most successful examples has been using these method is replacing Paul Walker's brother's face by Paul Walkers in the Fast and Furious 7 movie after the sudden death of Paul Walker in a car crash during shooting. Now that I have conviced you that this is not just for fun but is useful too, In the next few sections, let us see how this can be done along with the specifications of the functions for each part.

1 Collect Data - 5Pts

This is step 0. Record two videos. One of yourself with just your face and the other with your face and your friend's face in all the frames. Convert them to `.avi` or `.mp4` format. Save them to the `Data` folder. Feel free to play around with more videos. In the first video, We'll replace your face with some celebrity's face or your favorite relative's face. In the second video We'll swap the two faces. If there are more than two faces in the video, swap the two largest faces.

2 Face Detection with Landmarks

The first step is to detect faces in the videos you recorded. Fortunately, you don't have to implement this part as I spent a lot of time hacking the codes available online to make it usable. Because this step is generally very slow I found some really good codes based on C++ and wrote some python/mex wrappers for them. In particular, you are provided with two libraries DLib http://dlib.net/ and X. Zhu's code https://www.ics.uci.edu/~xzhu/face/. Both are really good and robust packages. The codes can be found in the folder `Code`.

DLib and X. Zhu's code can be respectively found in `Code/DLib` and `Code/XZhu` folders. For running DLib code run the `Code/DLib/python_example/DLibRunner.m` code. For running X. Zhu's code run the `Code/XZhu/XZhuRunner.m` code. I installed the necessary packages for both the codes on the server. In case you want to install them on your own laptops/PCs they can be downloaded from their respective websites and compiled. In addition to detecting faces across various poses, these packages also give you the facial landmarks. You can use any other code for this part but please be sure to cite them.

3 Face Warping using Triangulation - 35Pts

Use the point correspondences from the last step to create the traingulation. Be sure to check that your correspondences match properly, they sould but please check this. The point on the nose on the first image should correspond to the nose on the second image and so on. Use MATLAB's `delaunay` to compute the triangulation. Note that we need to use the same traingulation for both point sets (corresponding to the two images) for the algorithm to make sense. Use the destination face (your face in the first video or any one of the faces in the second video) to create the triangulation and apply it to the source face (celebrity's face in the first video and the other face in the second video). You can also choose to do the triangulation on the midway shape (mean of two point sets) to lessen the potential traingle distortions. Let us call the two point sets as `A` and `B` denoting the landmarks on two faces. The main part of this process is to map the intensity values after the triangles have been moved/affine transformed (Remember any triangle can form a plane). We will employ inverse mapping because it is generally easier to interpolate than splatting. Follow these steps for a successful face warp using triangulation.

1. For each pixel in the target shape (`B`), determine which triangle it falls inside. Use MATLAB's `tsearchn` for this. But think how you can implement your own barycentric coordinate check.

2. Compute the barycentric coordinate for each pixel in the corresponding triangle. Recall, the computation involves solving the following equation:

$$\begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

   Note, here a, b, c are the three corners of the triangle, (x, y) is the pixel position and α, β, γ is the barycentric coordinate. Note that you need to compute the 3×3 matrix on the left hand side and it's inverse only once per triangle. **DO NOT USE `tsearchn` for computing the barycentric coordinate**.

3. Compute the cooresponding pixel position in the source image: using the barycentric equation (shown in last step), but with three corners of the same triangle in the source image $\begin{bmatrix} a_x^s & b_x^s & c_x^s \\ a_y^s & b_y^s & c_y^s \\ 1 & 1 & 1 \end{bmatrix}$ and plug in the same barycentric coordinate (α, β, γ) to compute

the pixel position (x^s, y^s, z^s). You need to convert the homogeneous coordinate to pixel coordinate by normalizing with z^s, i.e., $x^s = x^s/z^s$ and $y^s = y^s/z^s$.

4. Copy back the pixel value at (x^s, y^s) the original (source) image back to the target image. Use `interp2` command in MATLAB to do the interpolation between pixels.

You have successfully warped the face using triangulation! Congrats!

4 Face Warping using Thin Plate Spline - 35Pts

As we discussed in class, triangulation assumes that we are doing affine transformation on each triangle. This might not be the best way to do warping considering that the human face is a very complex and smooth shape. A better way to do the transformation is by using Thin Plate Splines which can model arbitrarily complex shapes (1D splines). Now, we want to compute thin-plate-spline that maps from the feature points `B` to the corresponding feature points `A`. Recall you need two of them, one for the x coordinate and one for the y. A thin plate spline has the following form:

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

Here, $U(r) = r^2 \log(r^2)$

We know there is some thin-plate spline (TPS) transform that can map the corresponding feature points in image `B` back to image `A`, using the same TPS transform we will transform all of the pixels of image `B` into image `A`, and copy back their pixel value. Warping a face using TPS is done in two steps.

In the first step, we will estimate the parameters of the TPS. Recall the solution of the TPS model requires solving the following equation:

$$\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where

$$K_{ij} = U\left(\|(x_i, y_i) - (x_j, y_j)\|\right)$$

$v_i = f(x_i, y_i)$ and the i^th row of P is $(x_i, y_i, 1)$. K is a matrix of size size $p \times p$, and P is a matrix of size $pby \times 3$. In order to have a stable solution you need to compute the solution by:

$$\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_p \\ a_x \\ a_y \\ a_1 \end{bmatrix} = \left(\begin{bmatrix} K & P \\ P^T & 0 \end{bmatrix} + \lambda I(p+3, p+3) \right)^{-1} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_p \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

where $I(p+3, p+3)$ is a $p+3 \times p+3$ identity matrix. $\lambda \geq 0$ but is generally very close to zero. Think about why we need this. Note that you need to do this step twice, once for x co-ordinates and once for y co-ordinates.

In the second step, use the estimated parameters of the TPS models (both directions), transform all pixels in image `B` by the TPS model. Now, read back the pixel value from image `A` directly. The position of the pixels in image `A` is generated by the TPS equation (first equation in this section).

You have successfully warped the face using TPS! Congrats!

Note that, both warping methods solve the same problem but with different formulations, you are required to implement both and compare the results.

5 Blending - 10Pts

We will follow a method called Poisson Blending to blend the warped face onto the target face. More details about this algorithm can be found in Ref. [1]. Note that, you do not need to program this part, you can use the code from the link in Ref. [2]. Feel free to use any other code but don't forget to cite them. Your task in this part is to blend the face as seamlessly as possible.

6 Motion Filtering - 15Pts

After you have detected, warped and blended the face your algorithm works really well for individual frames. But when you want to do this for a video, you'll see flickering. Come up with your own solution to reduce the amount of flickering. You can use a low-pass filter or a fancy Kalman Filter to do this. Feel free to use any third party or built-in MATLAB code to do this. If you use third party code, please do not forget to cite them.

7 Extra Credit

- Implement fancy filters like bunny ears, doggy nose and so on. Look at SNOW `https://goo.gl/eijFvF` for some ideas or take ideas from Google Hangouts or Snapchat. **(Upto 5Pts)**.

- Make this work on Live Video in class! Will be cool. You might need methods to make this work fast (sort of real-time). **(Upto 10Pts)**.
- Artificially make everyone smile in the video - if they are not already. **(Upto 5Pts)**.
- Identify the expression and display an emoji (can be hard-coded per expression) for the current expression. **(Upto 20Pts)**.
- Capture the frame if a person blinks or smiles. **(Upto 10Pts)**.
- Present in class. **(Upto 10Pts)**.
- Any other cool thing you can think of. **(Upto 10Pts)**.

8 Starter Code

There is no starter code given for this project but face detection codes are given in the Folder `Code/FaceDetectorCodes`. For running the `DLib` code run the `DLibRunner.m` in `Code/FaceDetectorCodes/DLib/python_examples`. For running the `XZhu` code run the `XZhuRunner.m` in `Code/FaceDetectorCodes/DLib/XZhu`.

9 Submission Guidelines

We have not given any videos for this project as the first part is to collect data (this is the training set). Your output will be a `.mp4` or `.avi` video with swapped faces. We'll also release 2 test sets on the day the project is due (If we forget, remind us - if you don't all of you will lose 20 points). Run your code on all the training set videos and the test set videos. Your report should contain all these outputs (good or bad - or you lose 20 points).

Describe all the steps (anything that is not obvious) and any other observations in your report. Your report should be in the IEEE double column format and should be typeset in LaTeXand **NOT** contain any code, feel free to include algorithms and math. Use the format provided in the `Draft` folder. Include outputs after each stage and talk about failure cases - we are interested in this. You should also include a detailed `README` file explaining how to run your code.

Submit your own videos (in `Videos/Video1, Videos/Video2`) and codes (`.m` files) with the naming convention `YourDirectoryID_P2.zip` onto ELMS/Canvas (**Please compress it to .zip and no other format**). If your e-mail ID is `ABCD@terpmail.umd.edu` or `ABCD@umd.edu` your Directory ID will be `ABCD`.
To summarize, you need to submit these things and in the following strcture: A zip file with the name `YourDirectoryID_proj2.zip` onto ELMS/Canvas. A main folder with the name `YourDirectoryID_P2` and the following sub-folders:

- `Code` with all your code. Make a function called `Wrapper.m` which takes no input arguments and saves the video output into the current path with the output convention as follows. If your input video has the name `Video1.mp4`, your output should be called `Video1Out.mp4`. Please include instructions on how to run your code in the `README` file. This assignment might be auto-graded so please follow the guidelines very carefully. All the file and folder names are case-sensitive.

- `Report` with your report in pdf format (typeset in LaTeXdouble-column IEEE format given to you in `Draft` folder). Present a comparison between triangulation and TPS methods too.
- `Videos` with two sub-folders `Video1` and `Video2` which has your training set videos.

If your submission does not comply with the above guidelines, you'll be given **ZERO** credit.

Also, please mention about all the extra credit you have done in your report.

Also make a presentation (need not be slides, you can use your report or the whiteboard or just talk) if you want to present it in the class **Note: Every student should present AT LEAST once in the class (you can volunteer to present for more than one project) and can choose to do for any of the Project except the last one. Good presentations will receive upto 10 bonus points.**

10 Disallowed Matlab functions

`tsearchn` to compute barycentric co-ordinates and any other function which completes the project.

11 Project Hardness

This project is at a difficulty level of graduate level final projects in many other universities, so please start early.

12 Collaboration Policy

You are restricted to discuss the ideas with at most two other people. But the code you turn-in should be your own and if you **DO USE** (try not it and it is not permitted) other external codes/codes from other students - do cite them. For other honor code refer to the CMSC733 Fall 2016 website.

DON'T FORGET TO HAVE FUN AND PLAY AROUND WITH IMAGES!.

Acknowledgements

This fun project was inspired from 'Computer Vision & Computational Photography' (CIS 581) course of University of Pennsylvania (https://alliance.seas.upenn.edu/~cis581/wiki/index.php?title=CIS_581:_Computer_Vision_%26_Computational_Photography).

References

[1] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *ACM Transactions on Graphics (TOG)*, volume 22, pages 313–318. ACM, 2003.

[2] Masayuki Tanaka. Poisson Image Editing. https://www.mathworks.com/matlabcentral/fileexchange/37224-poisson-image-editing/content/PoissonEdiitng20151105/imgrad.m, 2012.