Programming Project #3 (`proj3`)

CS180/280A: Intro to Computer Vision and Computational Photography

Face Morphing

Project Due Date: 11:59 pm on Tuesday, October 8, 2024

Overview

In this assignment you will produce a "morph" animation of your face into someone else's face, compute the mean of a population of faces and extrapolate from a population mean to create a caricature of yourself.

A morph is a simultaneous warp of the image shape and a cross-dissolve of the image colors. The cross-dissolve is the easy part; controlling and doing the warp is the hard part. The warp is controlled by defining a correspondence between the two pictures. The correspondence should map eyes to eyes, mouth to mouth, chin to chin, ears to ears, etc., to get the smoothest transformations possible.

To start with, you should take a pictures of yourself on a uniform background (for instance, white). Your image should be the same size and aspect ratio as your target face (for instance, this beautiful portrait of George, taken by Martin Schoeller). Treat your target face as a passport photo template -- your face in the picture should be about where their face is. You can use photo editing software to resize/crop your image such that this is the case. This will make your morphing result more pleasing to the eye.

Use your photo as Picture A and your target person's photo as Picture B. You'll morph still picture A into still picture B and produce 45 frames of animation numbered 0-45, where frame 0 must be identical to picture A and frame 45 must be identical to picture B. In the video, each frame will be displayed for 1/30 of a second (ie. 30 fps). Create a video from your sequence of frames, either a YouTube or an animated gif.

Part 1. Defining Correspondences

First, you will need to define pairs of corresponding points on the two images by hand (the more points, the better the morph, generally). The simplest way is probably to use the `cpselect` (matlab) tool or write your own little tool using `ginput` (matlab or python) and `plot` commands (with `hold on` and `hold off`). In order for the morph to work you will need a consistent labeling of the two faces. So label your faces A and B in a consistent manner using the same ordering of keypoints in the two faces. It's strongly recommended that you save the points once you obtain something you are happy with so that you don't have to do all that clicking more than once!

Now, you need to provide a triangulation of these points that will be used for morphing. You can compute a triangulation any way you like, or even define it by hand. A Delaunay triangulation (see `delaunay` and related functions) is a good choice since it does not produce overly skinny triangles. You can compute the Delaunay triangulation on either of the point sets (but not both -- the triangulation has to be the same throughout the morph!). But the best approach would probably be to compute the

triangulation at a midway shape (i.e. mean of the two point sets) to lessen the potential triangle deformations.

Note if you don't want to implement your own labeling tool, you can use this one from a last year's student.

Part 2. Computing the "Mid-way Face"

Before computing the whole morph sequence, compute the mid-way face of your images A and B. This would involve: 1) computing the average shape (a.k.a the average of each keypoint location in the two faces), 2) warping both faces into that shape, and 3) averaging the colors together. The main task in warping the faces into the average shape is implementing an affine warp for each triangle in the triangulation from the original images into this new shape. This will involve computing an affine transformation matrix A between two triangles:

```
A = computeAffine(tri1_pts,tri2_pts)
```

(You will write this function.)

A set of these transformation matrices will then need to be used to implement an inverse warp (as discussed in class) of all pixels. One way to do so is to generate a mask directly using `polygon` in python or `roipoly` in MATLAB. Interpolation functions may be useful for this implementation, but you are not permitted to use any built-in functions for computing transforms.

Note that you only need to write one loop here, looping over all of the triangles. Don't loop over the pixels!!!

Show us the original A and B images as well as the image of the mid-way face that you got.

Part 3. The Morph Sequence

You need to write a function:

```
morphed_im = morph(im1, im2, im1_pts, im2_pts, tri, warp_frac, dissolve_frac);
```

that produces a warp between `im1` and `im2` using point correspondences defined in `im1_pts` and `im2_pts` (which are both n-by-2 matrices of (x,y) locations) and the triangulation structure `tri` . The parameters `warp_frac` and `dissolve_frac` control shape warping and cross-dissolve, respectively. In particular, images `im1` and `im2` are first warped into an intermediate shape configuration controlled by `warp_frac`, and then cross-dissolved according to `dissolve_frac`. For interpolation, both parameters lie in the range [0,1]. They are the only parameters that will vary from frame to frame in the animation. For your starting frame, they will both equal 0, and for your ending frame, they will both equal 1.

The output of this part should be a video sequence of a morph from your image A to image B. Please do NOT put a video file on your website! Either include a link to a YouTube video, or create an animated gif.

Part 4. The "Mean face" of a population

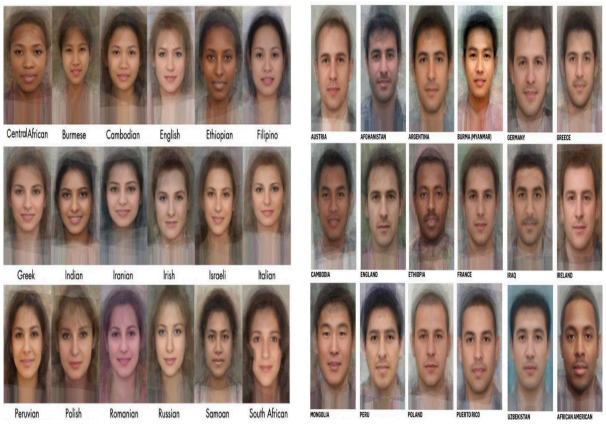
Pick a freely available dataset of annotated faces (for instance the Danes or this one or something from here or ask for permission to use this one). Using the keypoints already annotated on the data:

1. Compute the average face shape of the whole population or some subset of the population - say, all the old/young/white/asian/men/women etc. However, if you pick a subpopulation - make sure it contains enough faces for this to be interesting.
2. Morph each of the faces in the dataset into the average shape. Show us some examples.
3. Compute the average face of the population and display it.

Show the mean image that you got, as well as 1) your face warped into the average geometry, and 2) the average face warped into your geometry.

Part 5. Caricatures: Extrapolating from the mean

Produce a caricature of your face by extrapolating from the population mean you calculated in the last step. This might work better on a gender specific mean or some other characteristics specific mean.



Bells and Whistles - (you will have to do at least one in order to get full credit)

- (up to **0.08 cookie points**) Change age/gender/ethnicity/smile/etc of your (or your friend's) face. You can use average images off the web for this, no need to recompute the averages yourself (unless you want to). Show morphing just the shape, just the appearance, and both.
- (up to **0.08 cookie points**) Make a morphing music video on a theme. For instance, you can pick a photo of yourself from different ages and make a movie that shows how your face changed over time. Or you can morph between your friends/dorm mates etc.
- (up to **0.08 cookie points**) Produce a face-morphing music video of the students in the class! Something like this but waaaayyyy cooler. In order to do this successfully you will have to gang up with other students in the class and organize yourselves in one big chain of students. Each student in the chain will have to create a morph video sequence from their face to the next students face - just like you did for the main part of the homework. In order to end up with something visually appealing it is advisable that all the backgrounds for everyone's photographs are the same -- for instance white. Another trick is to get everyone's face roughly in the same place. For this you can use George as a passport photo template -- your image should be the same size and aspect ratio as George's and your face in the picture should be about where his face is. Finally, everyone should label their keypoints in a consistent manner with one another. For instance, you can use the point labeles in the following two images as a guide for everyone's labelling: points, point_labels. At the end, pick a volunteer to put everyone's videos together in order and add some music. Here are a coule of good tunes, but obviously you can come up with your own!: Kate Nash, Smash Mouth, The Flaming Lips, Steelers Wheel, Asian historical period drama etc. Everyone in the movie gets 0.02 cookie points for each student participating.
- (up to **0.08 cookie points**) Use one of the datasets to compute a PCA basis for the face space. Try performing caricatures and other transformations in the new basis. Compare your results to doing this in normal basis. Are the results better?
- (up to **0.08 cookie points**) Do something else that is fun in this space, e.g. a different morphing algorithm, automatic morphing, "visual lip-syncing", whatever else your imagination comes up with.
- (up to **0.08 cookie points**) Create caricatures by implementing a non-linear face-morphing transformation on pixels (instead of keypoints), similar to the liquify filter in photoshop. Come up with non-linear functions for bloating, shrinking, or twirling (etc) over an area of pixels, and apply them to a face (Hint: inverse mapping and scipy.interpolate.griddata might be helpful here).
- (up to **0.08 cookie points**) Make an interactive tool that allows you to use keypoints as handles to manually drag and alter facial expression or proportions.

Deliverables and Scoring

In summary, this project will require you to perform the following and describe any implementation or design details on your project webpage:

- Implement code for selecting facial keypoints and creating a triangulation mesh of these points (ginput and scipy's delaunay function can be used). Show facial keypoints and triangulation you generated on the website.
- Implement code for an affine warp to create a midway face between your two images. Display this midway face on your website alongside both of the original images.
- Implement code to create a morph sequence and display a gif of this morph sequence on your website.
- Compute the mean face of a particular population using images and keypoints given in their data. Display several images from the dataset with their face geometry warped into the average shape. Now warp your face into the average face's geometry and the average face into your face's geometry and display the results on your website.
- Generate and display a caricature by extrapolating from a population mean.
- Complete **at least one bell's and whistles**. Any additional bells and whistles will earn you extra points towards quiz drops over the semester. You may earn up to **0.50 cookie points** for completing all bells and whistles in this project.

Submission

For this project you must turn in both your code and a project webpage as described here.