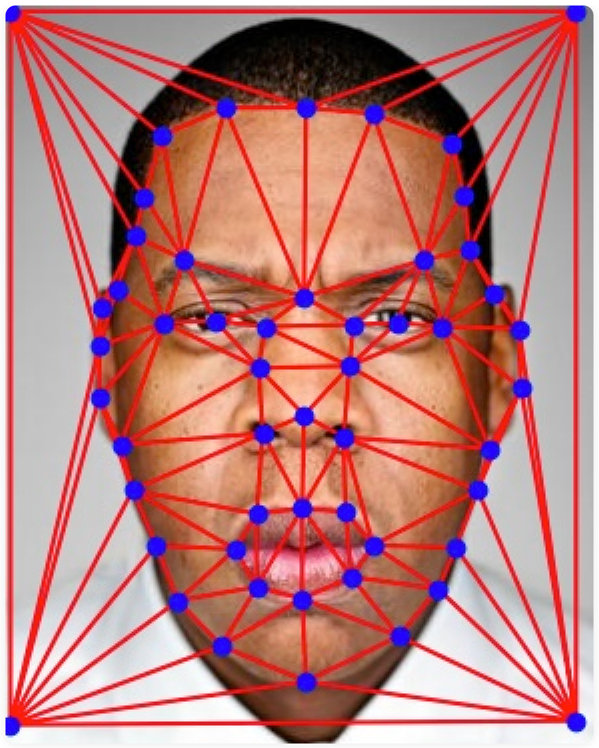
Saatvik Billa CS180 Project 3: Face Morphing

Part 1: Defining Correspondences

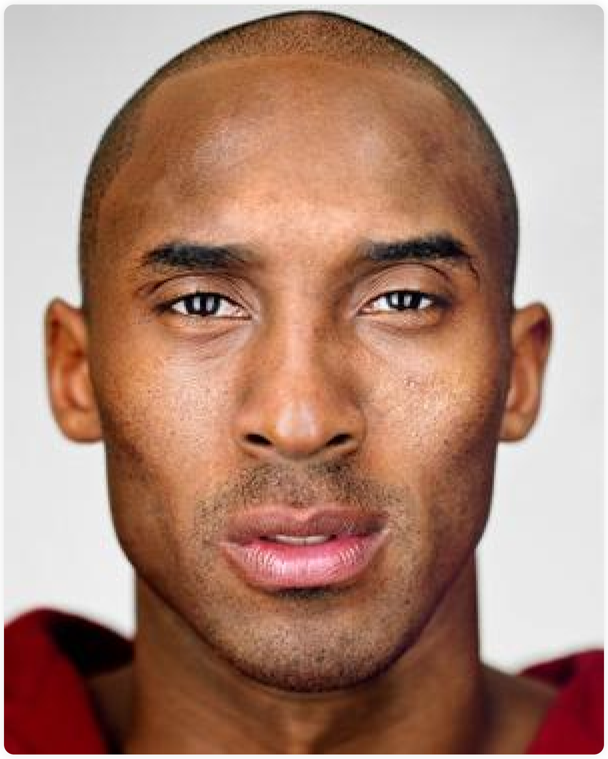
In this section, I created a triangulation that both images could work with, which consists of a set of points that are connected in the same way between both images. I used the tool created by a former student to select points that are consistent, and then I used the Delaunay class from the scipy.spatial package to actually create the set of triangles.

The images shown below are the original ones overlaid with the triangulation. It's important to mark each of the four corners so that you can grab the pixels that are not on the face. Otherwise in future steps when you're morphing, all the pixels around will turn black.

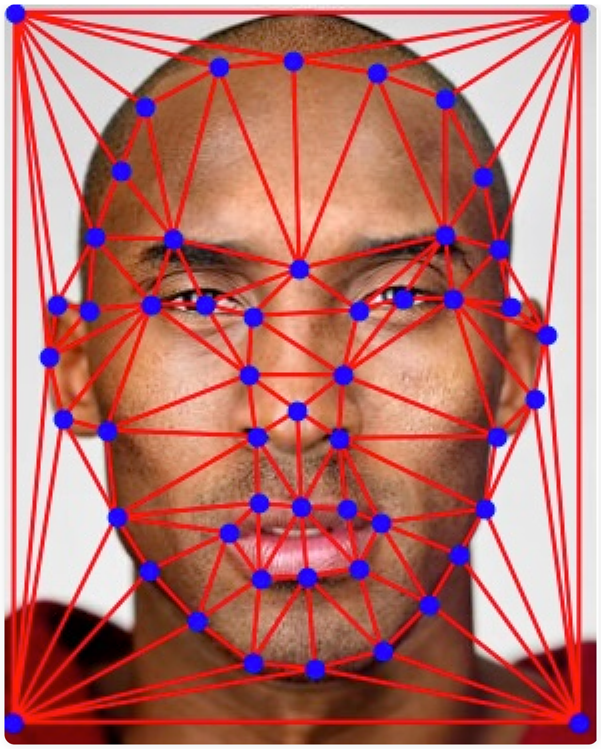
Original Jay-Z Picture



Jay-Z with Triangulation



Original Kobe Picture



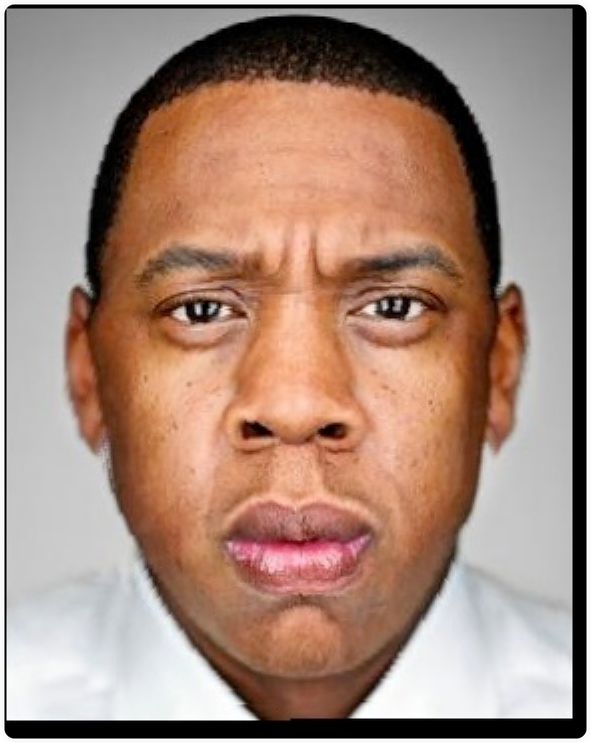
Kobe with Triangulation

Part 2: Computing the "Mid-way Face"

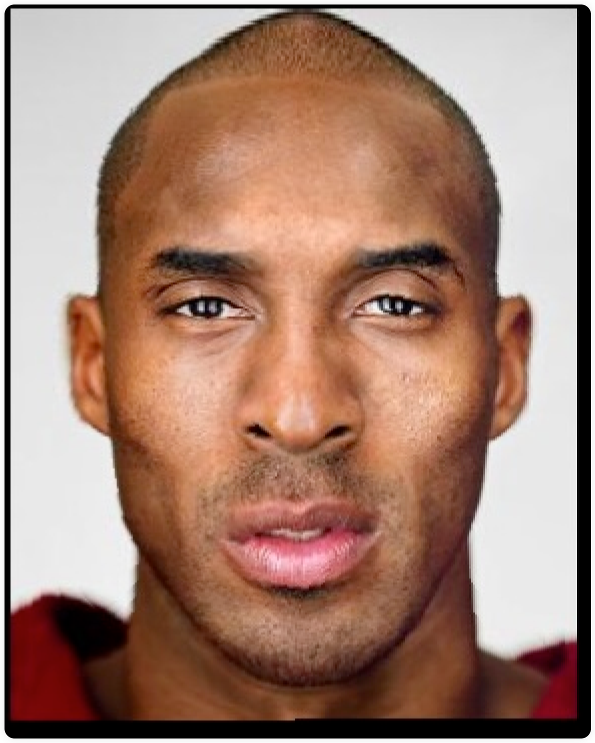
In order to compute the mid-way face, I had to basically warp each of the images such that they line up with the triangulation that we computed in the first part. To be more specific, this triangulation was calculated by taking the midpoint between each correspondence point between both images, so right out of the box, it wasn't going to line up with each individual image, hence the necessity to warp.

In order to do this, for each image, I looked at the set of triangles, which are nothing but a list of three correspondence points. I looked at the coordinates in the original image, the ones computed by the average, and I defined an affine transformation matrix that would take me from the latter to the former. After I got this matrix, I then defined a mask using the polygon function from the skimage.draw package get a list of all the coordinates in the image that were inside of the triangle. From there, I applied this affine transformation to each of the coordinates, got those pixel values, and placed them at the corresponding place in the newly warped image. That's how I got the warped image for each individual.

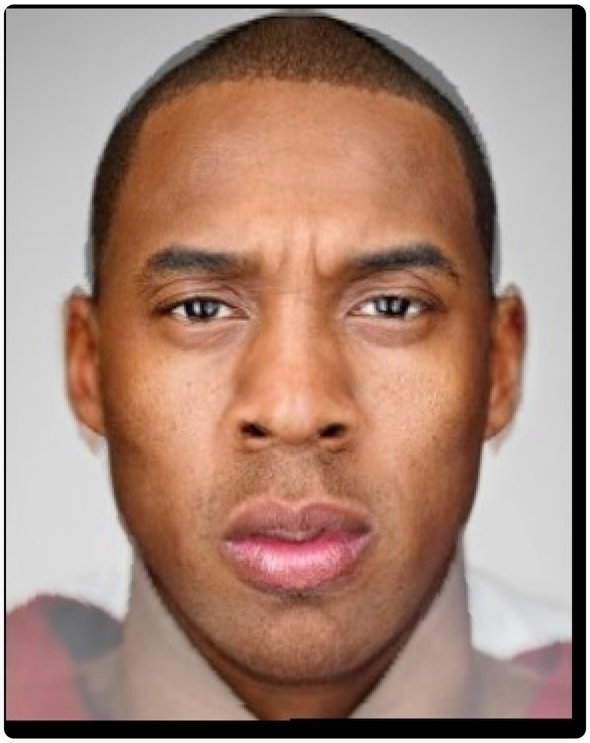
For the completely averaged midway-face, all I did was take the numpy arrays representing both of the warped images and average them out. I used nearest neighbor interpolation by casting the pixel values calculated by the matrix to ints.



Jay-Z Mid-Way Face



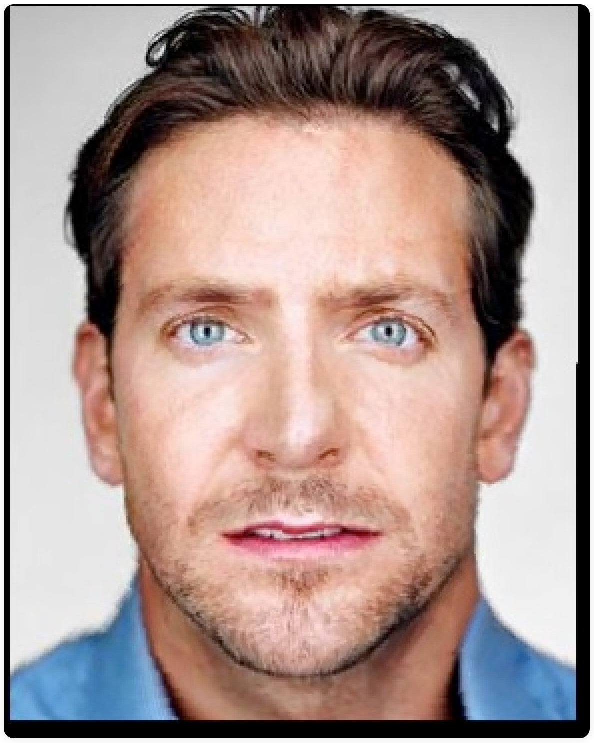
Kobe Mid-Way Face



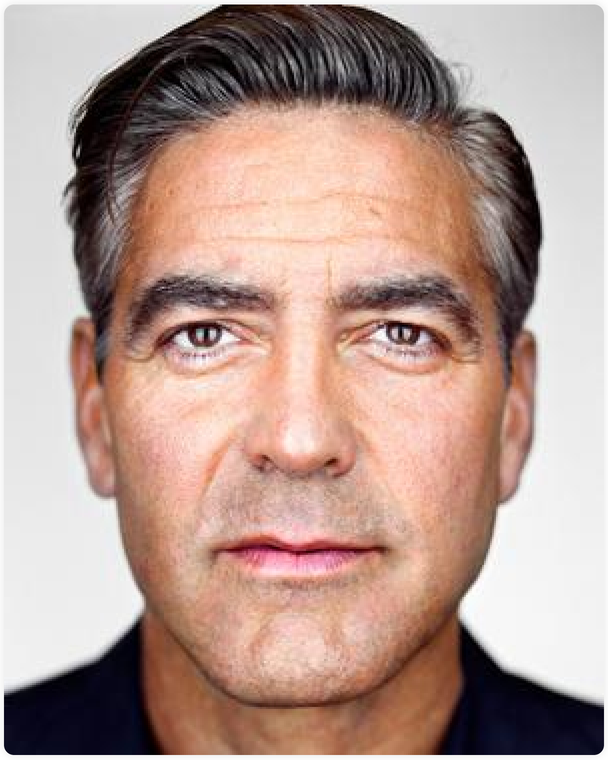
Kobe and Jay-Z Averaged Mid-Way Face

Another Example:

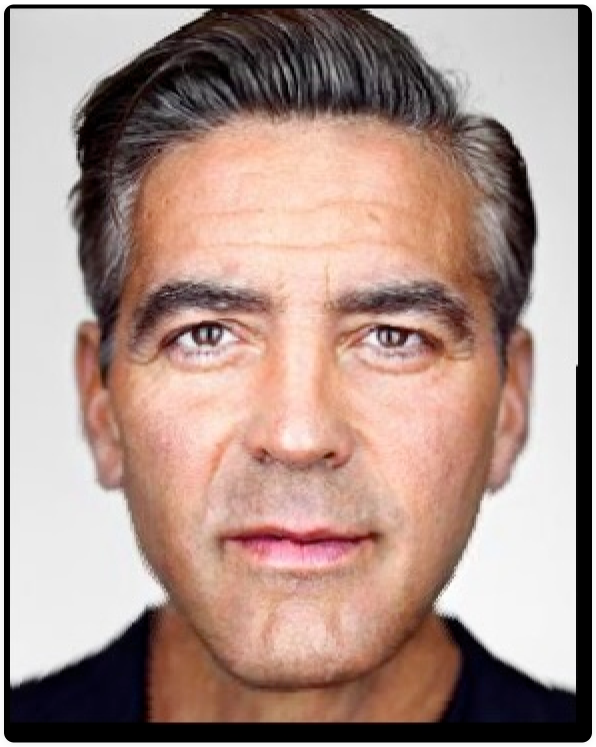
Bradley Cooper Original Face



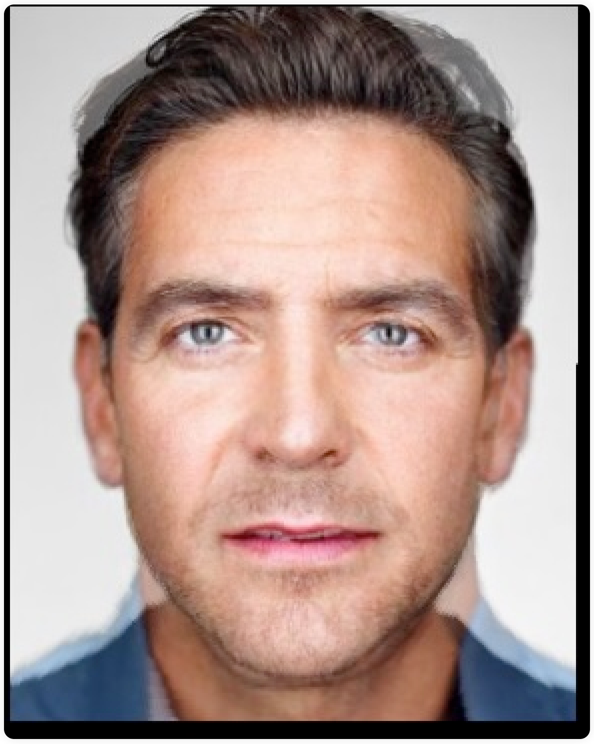
Bradley Cooper Mid-Way Face



George Clooney Original Face



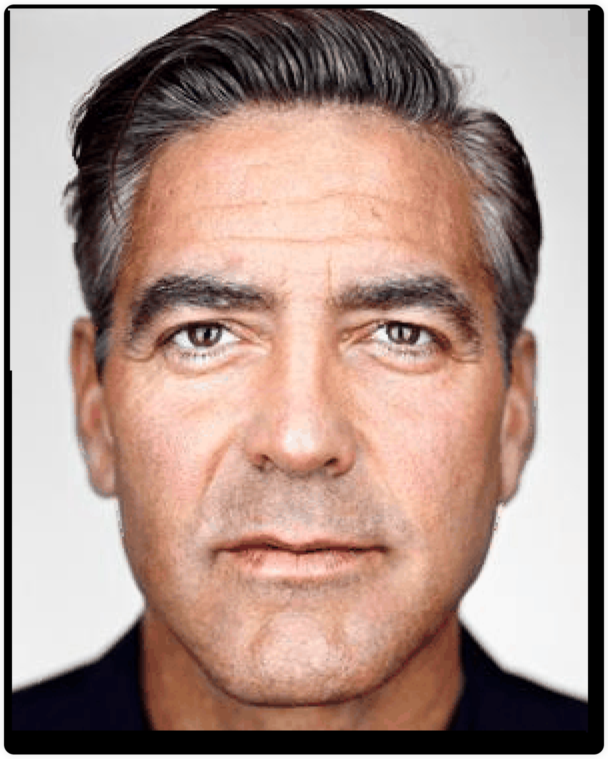
George Clooney Mid-Way Face



Clooney and Cooper Averaged Mid-Way Face

Part 3: The Morph Sequence

This part was pretty easy since I did Part 2 successfully. All I had to do was create a bunch of frames and then stitch them together. A frame is nothing but a linear combination of both of the images instead of an even average. For this specific GIF, I used 240 frames, so it started out with using only 1/240th of the first image but ending up using 1/240th of the last image. I set my frames per second to be default 30 frames per second.


GIF showing morph from George Clooney to Bradley Cooper

Reset GIF

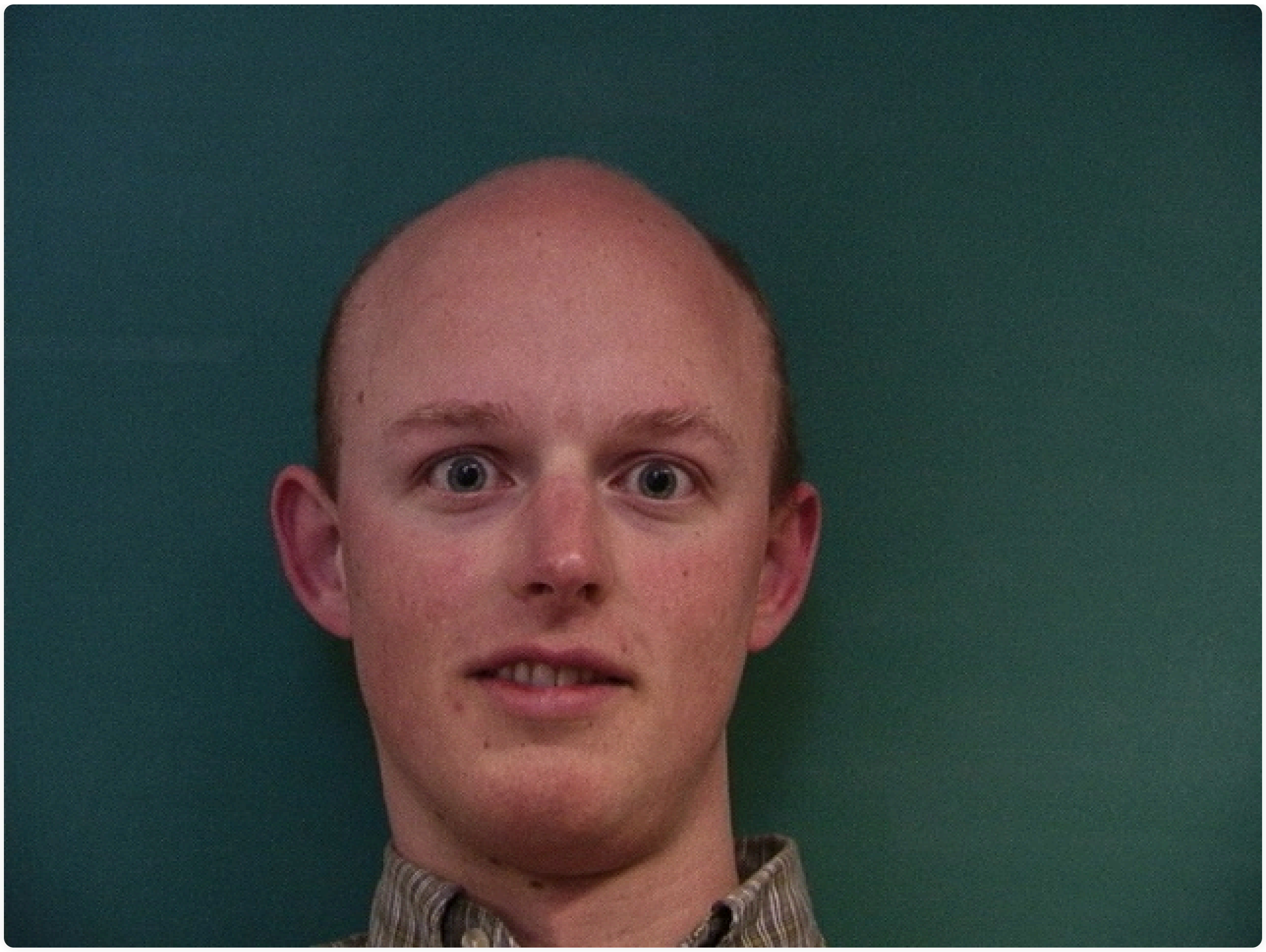
Part 4: The "Mean face" of a population

Very similar once again to Part 2. Instead of averaging over 2 people now, I averaged over 4 and created the triangulation off of that. Then using the same functions as before, I warped each individual to the population average and morphed them all together.

Original Pictures of White Men

White Man #1

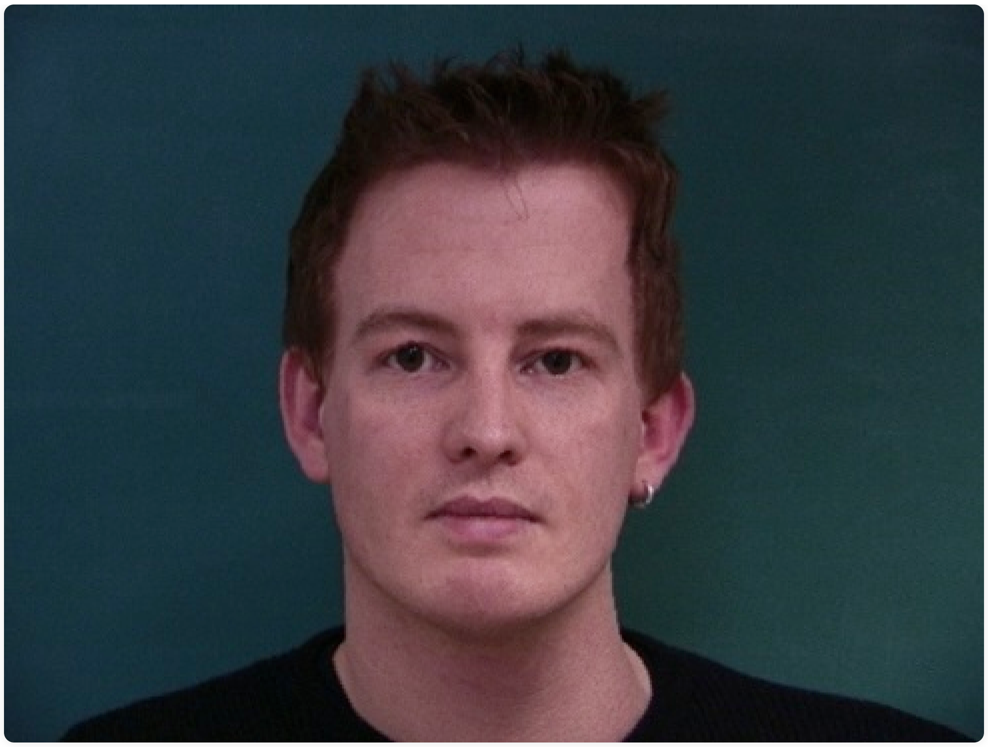
White Man #2



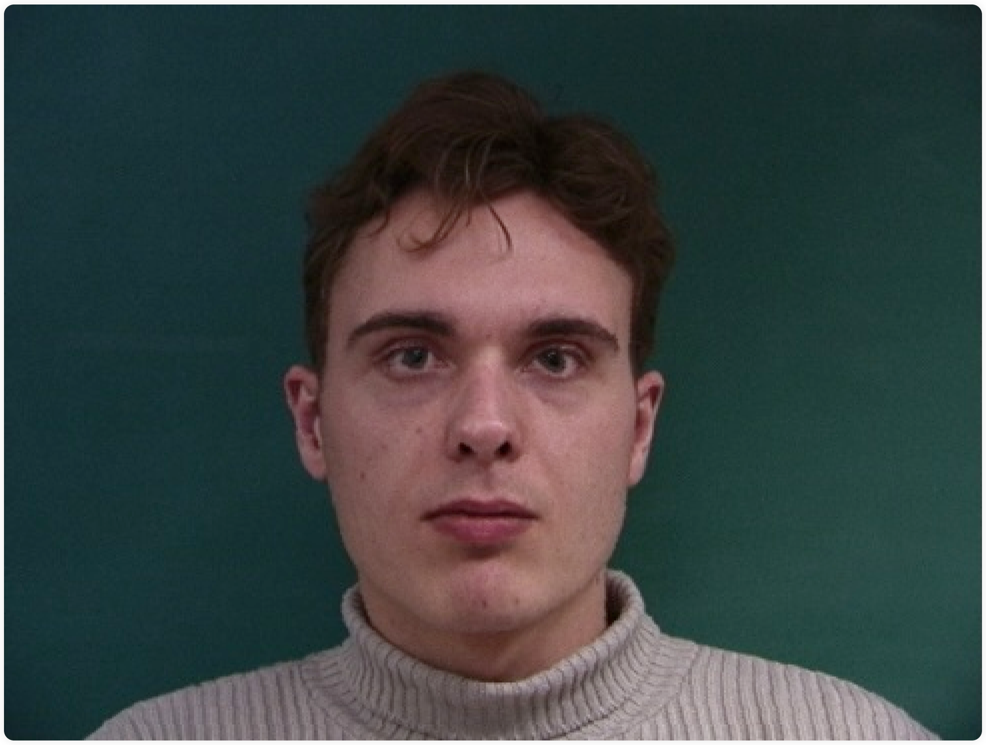
White Man #3

White Man #4

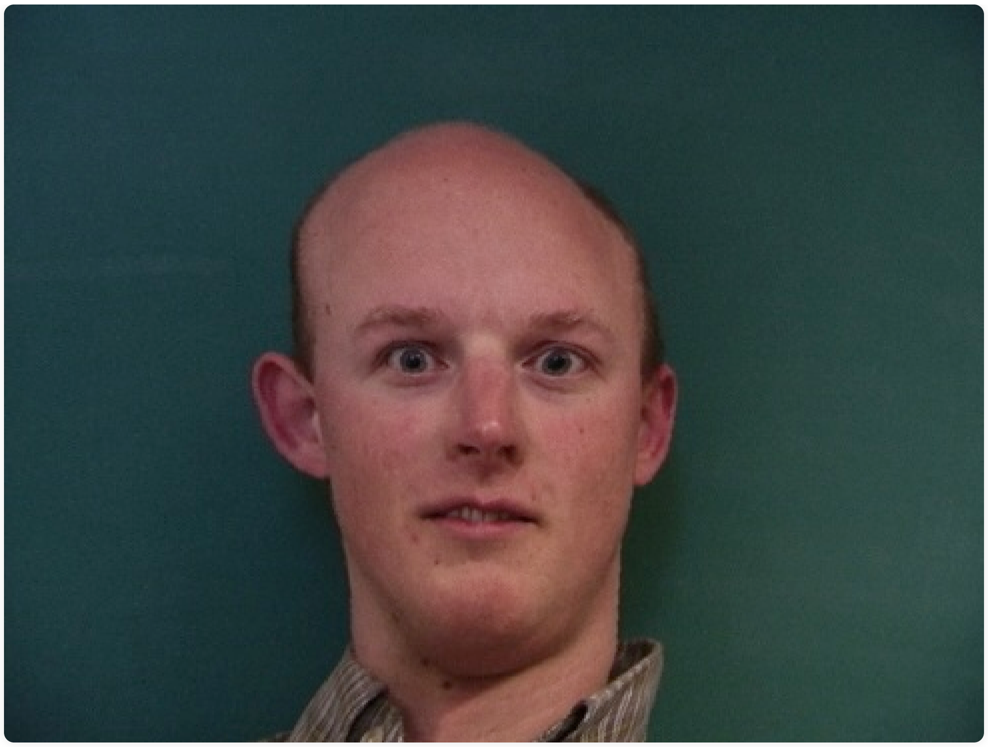
White Men Individually Morphed Into Average



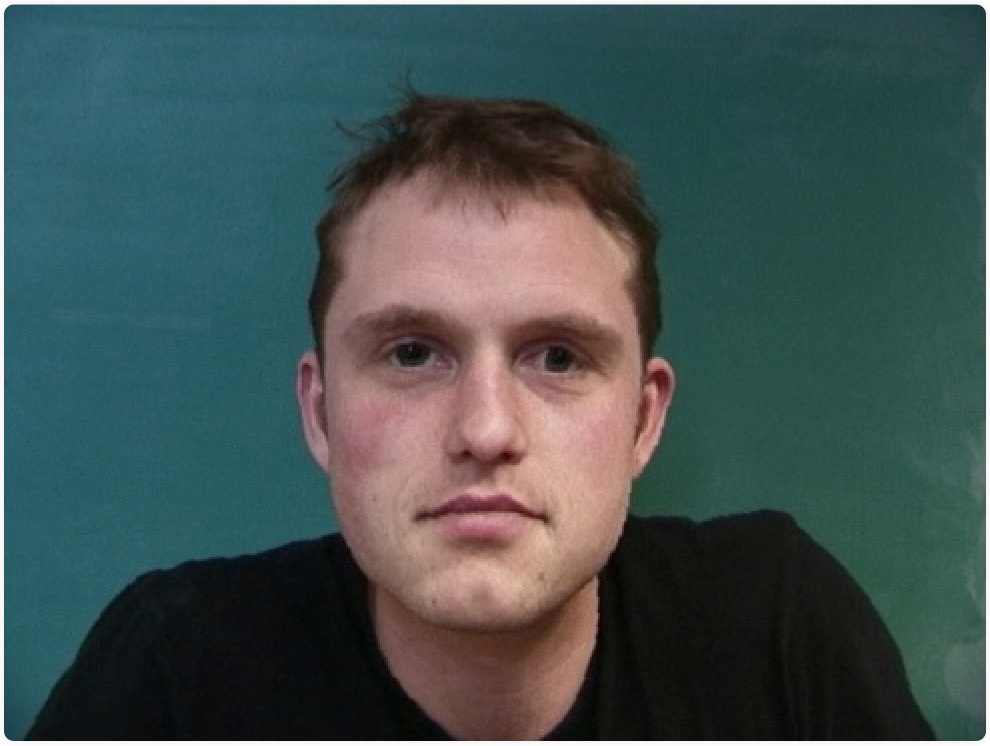
Morphed White Man #1



Morphed White Man #2

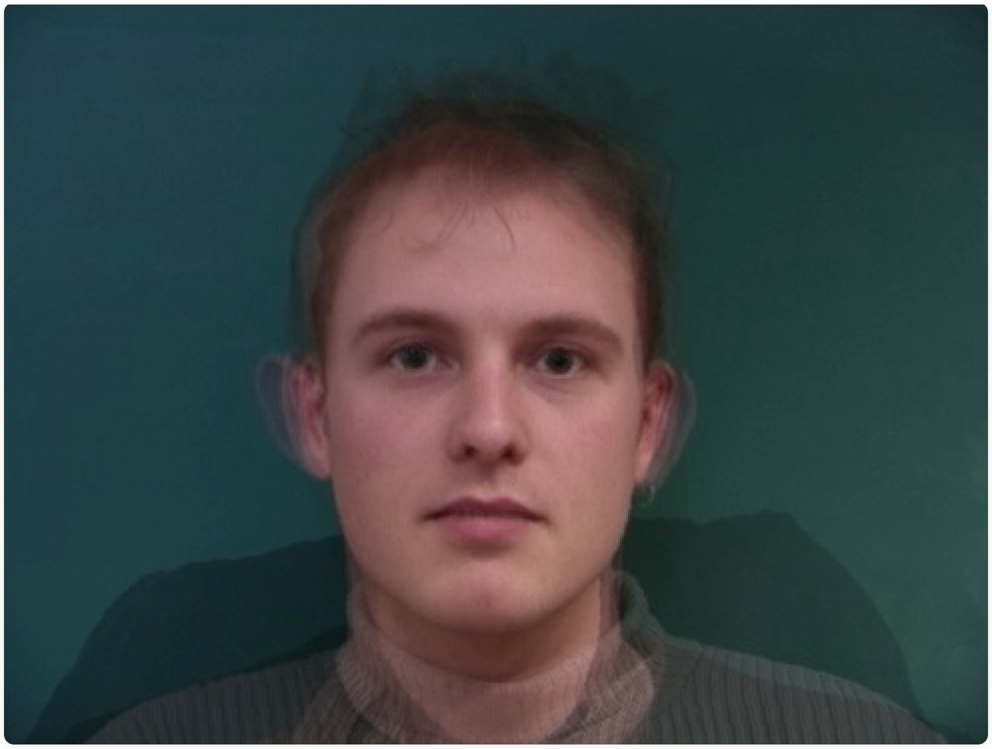


Morphed White Man #3



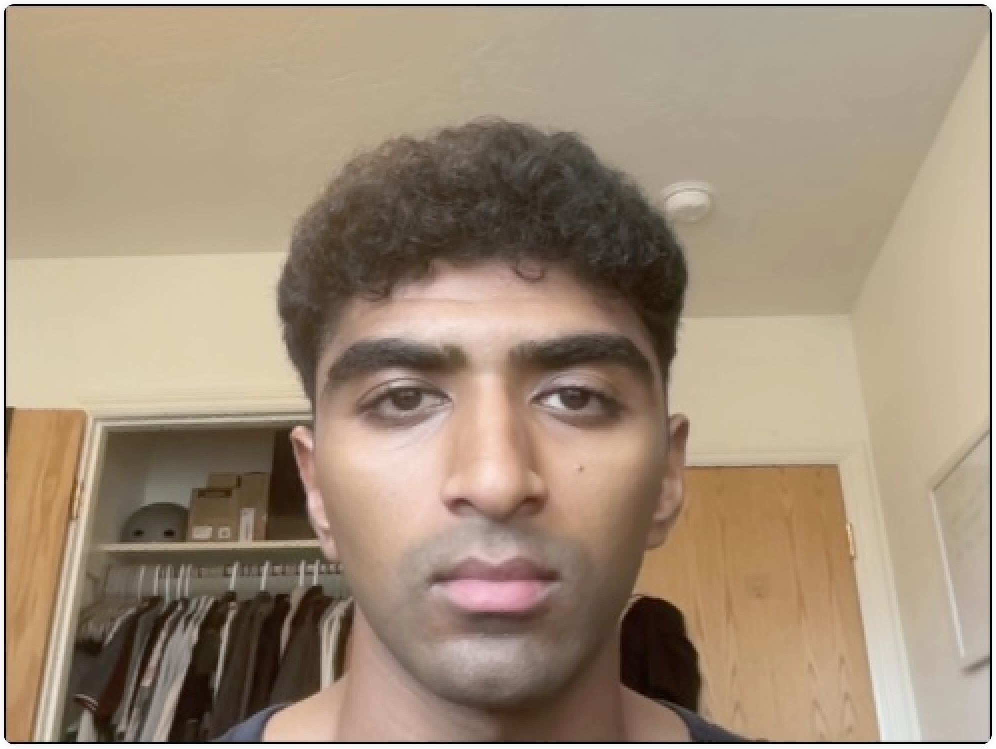
Morphed White Man #4

All White Men Morphed Together

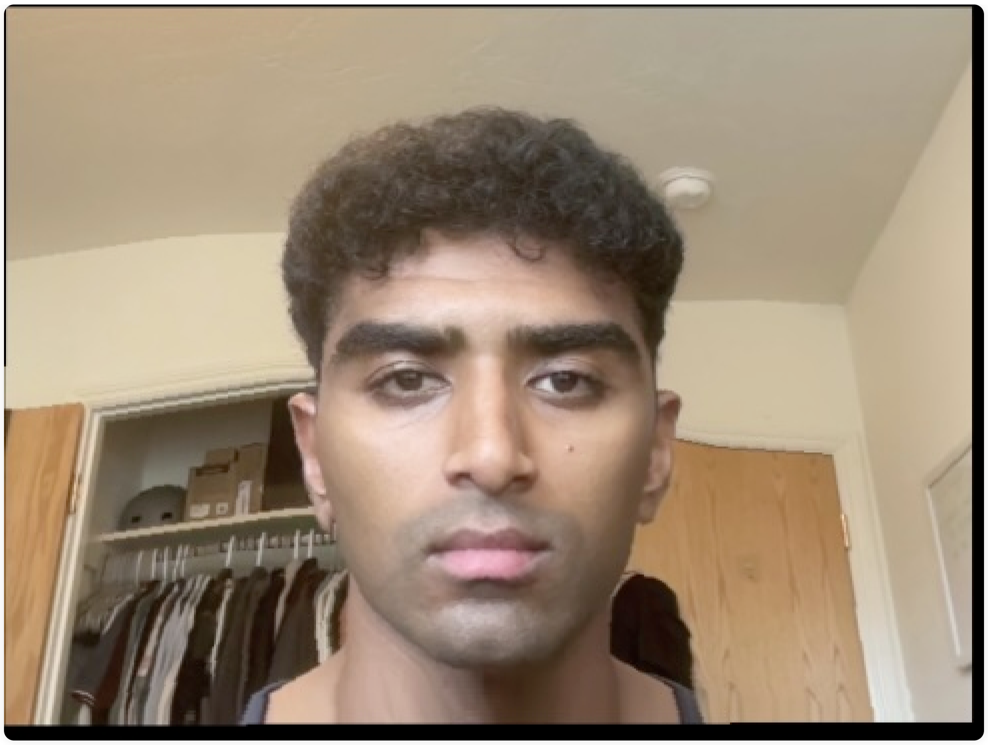


Subset of Population Average

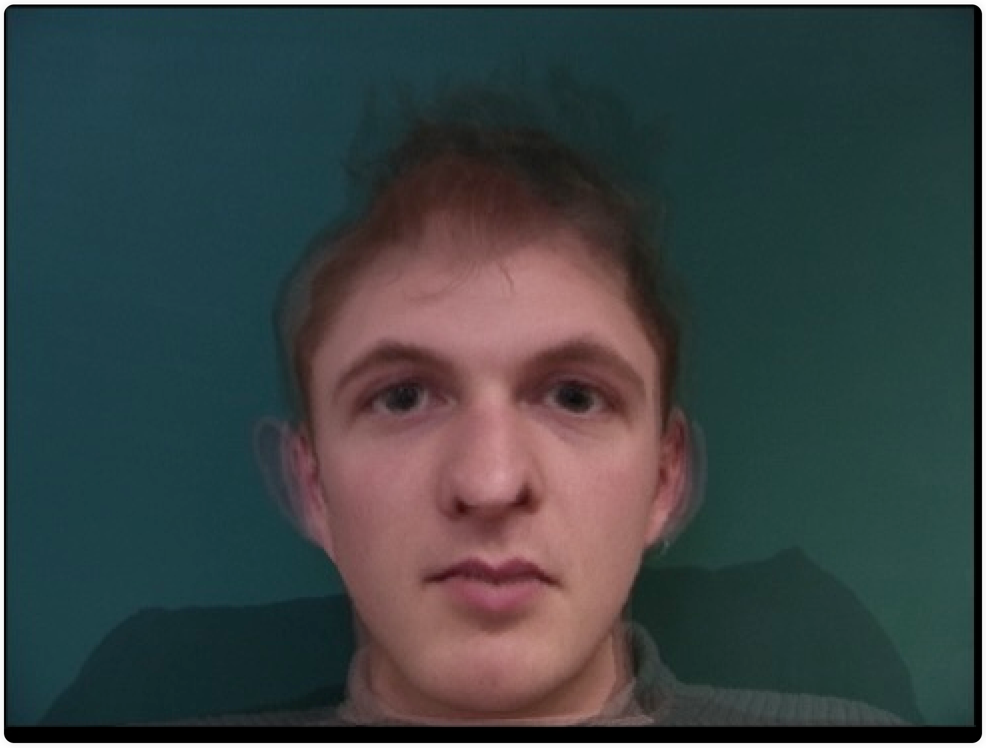
Warping My Face With White Men Average



My face



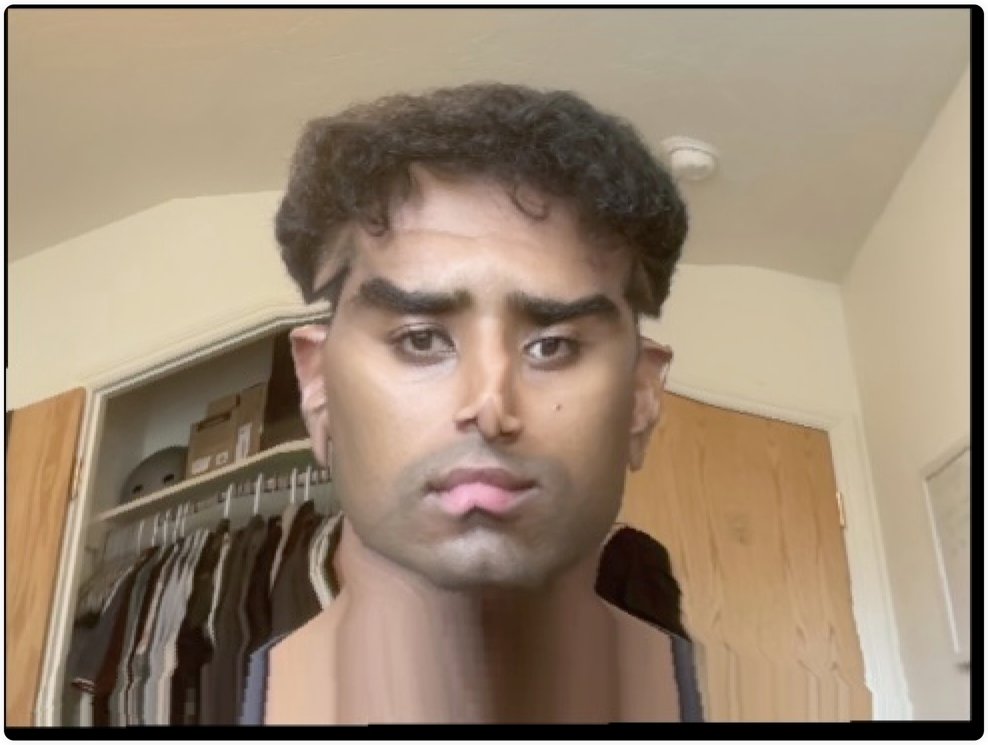
My face warped to white man average



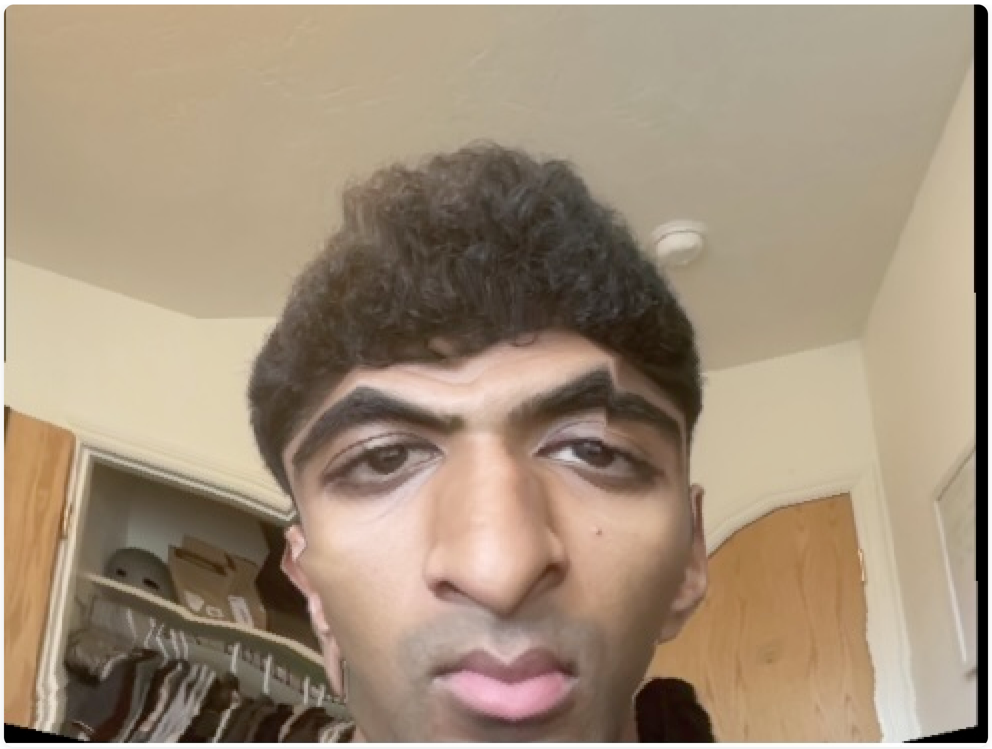
White man average warped to my face

Part 5: Caricatures: Extrapolating from the mean

Here, all I had to do was use the morph function I defined in Part 5, except play around with the warp_frac value a little bit, which basically controls how much of the difference between my face and the white man average face am I adding to my facial features. In the first picture, I am adding 1.5 times the white man features, and in the second one, I am actually subtracting 1.5 times the features.



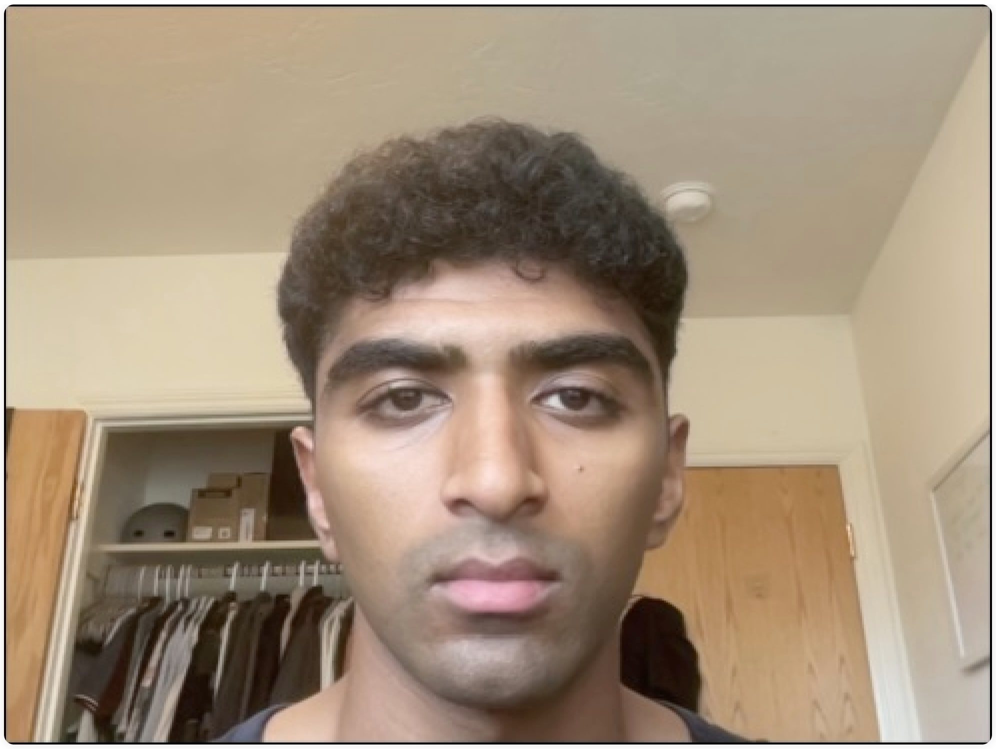
warp_frac = 1.5

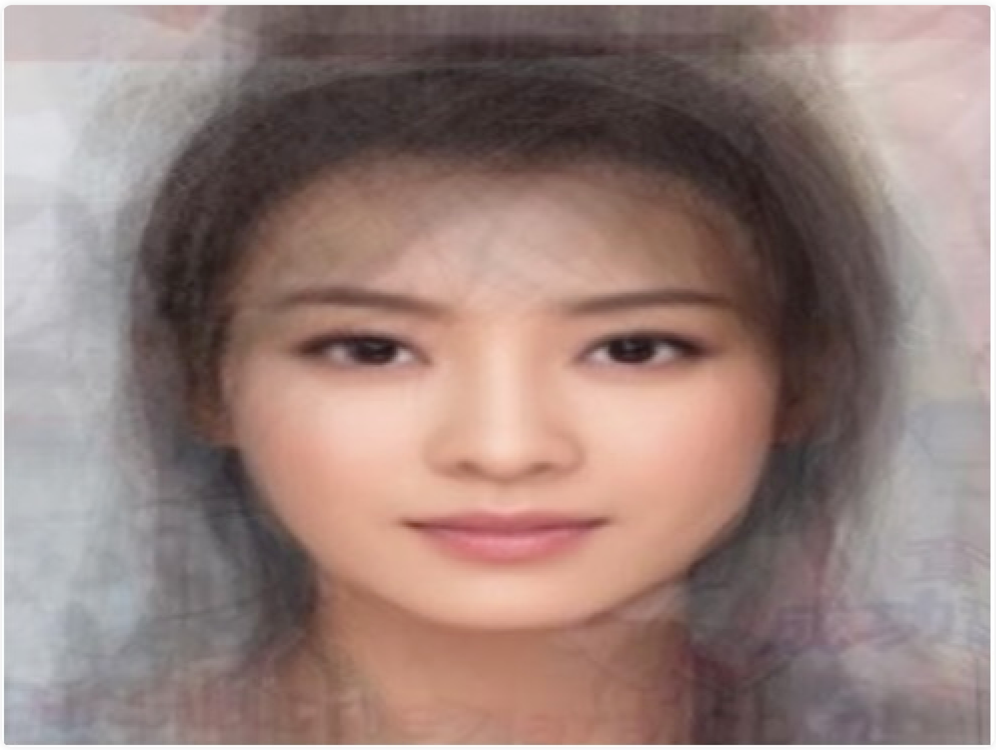


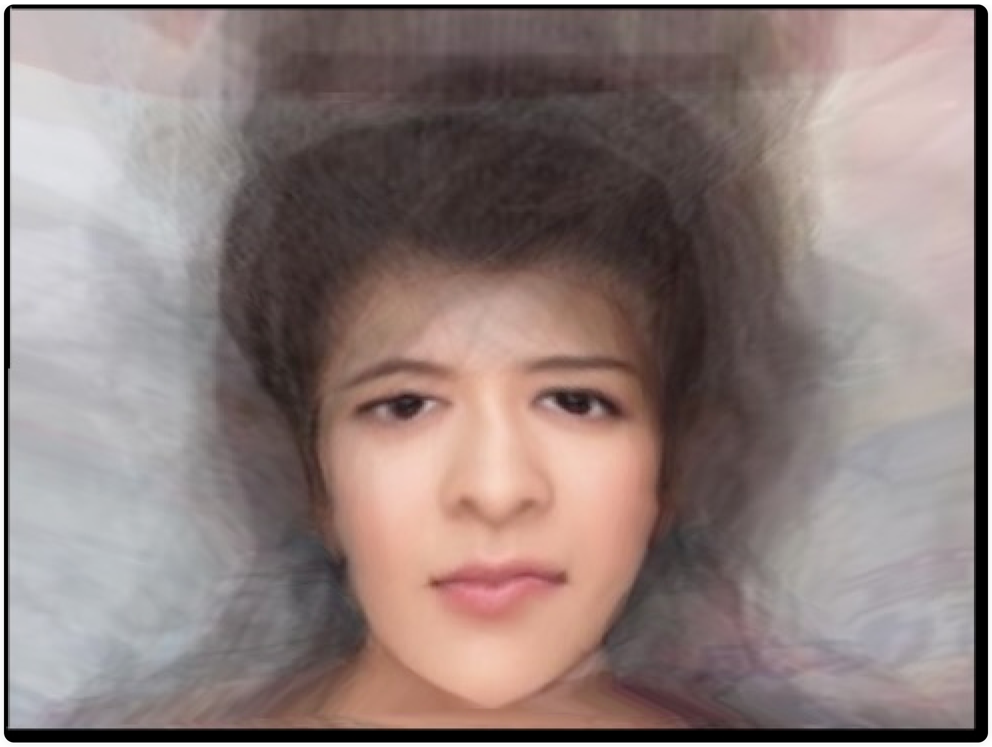
warp_frac = -1.5

Part 6: Bells and Whistles

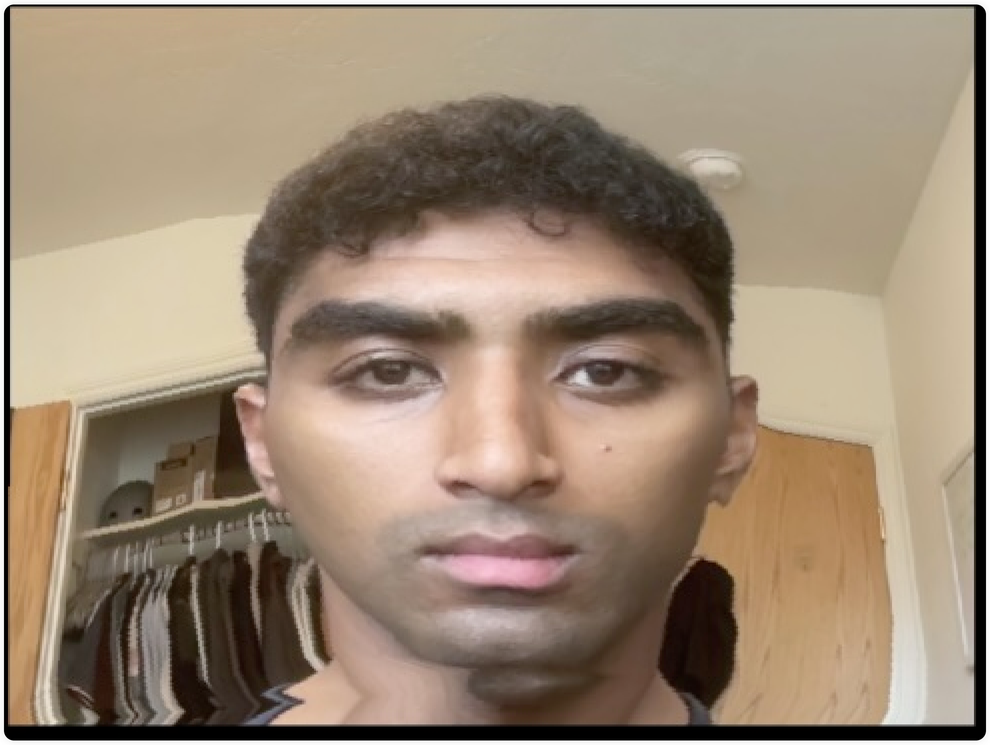
I aimed to change both the gender and ethnicity of myself, so I took the average chinese woman's face and experimented with how I looked when I cross-dissolved with only her colors, warped to her triangulation, and finally morphed with her properly. I'd say I look pretty good!


My Face

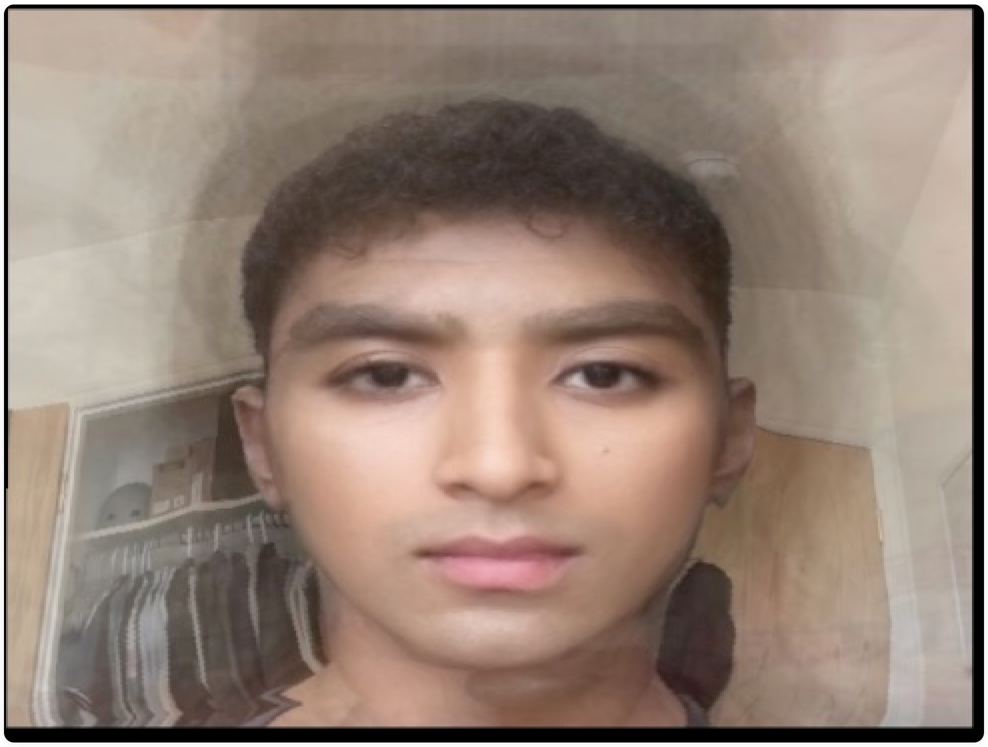

Average Chinese Woman's Face



My Face with her colors



My Face with her shape



Our faces completely morphed. Aren't I sexy?