CS 194-26: Image Manipulation and Computational Photography, Fall 2022

Project 3: Face Morphing

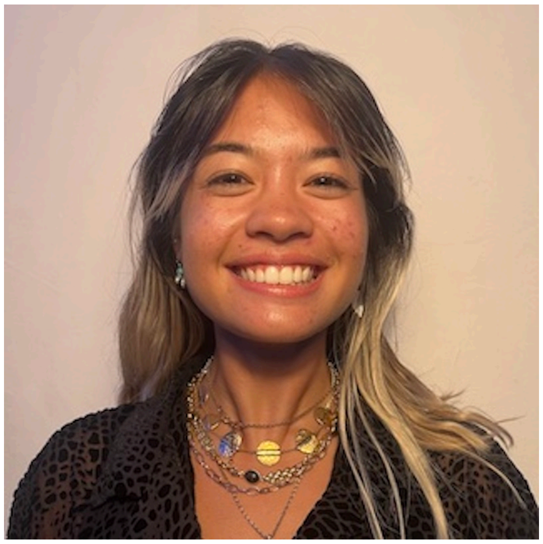
Ethan Gnibus

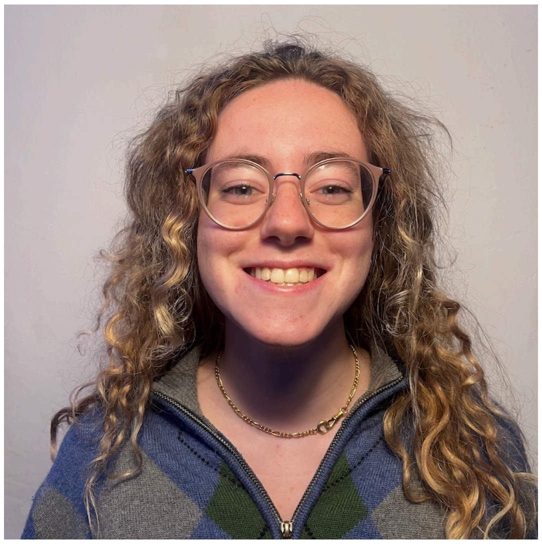
Overview

In this project, I will morph Image1 into Image2 by simultaneously warping Image1 into the shape of Image2 while cross-dissolving the colors from Image1 into Image2. I will use this technique to compute the "Mid-way" face, or the average shape and colors of two faces. Additionally, I will make a video of a gradual warp from one face to another. I will also find the average face shape for an entire population use it to show more possibilities of warping. I will then use this population mean to make a caricature of my face. I've also decided to make a music video at the end.

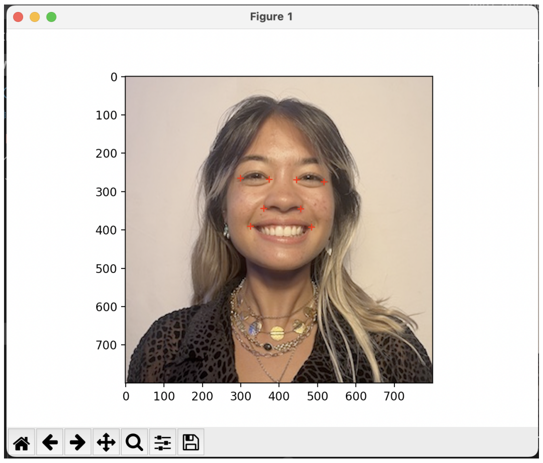
Part 1. Defining Correspondences

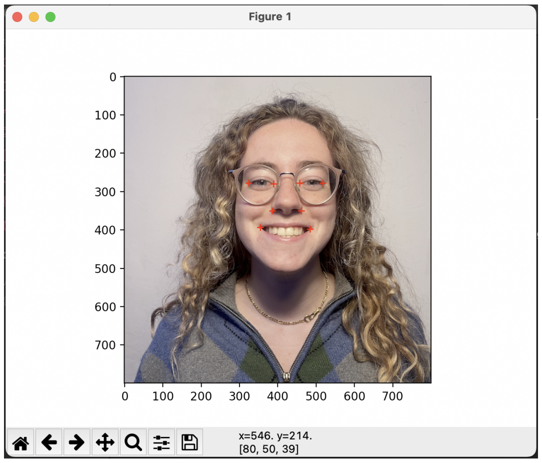
To morph faces, we must first warp them into an average shape. Doing this requires us to find corresponding points in both images, averaging the location of those corresponding points, then warping our images so that their corresponding points overlap with the average location.
The first step in this process involves defining these correspondences, which entails creating a GUI to pick points on both images, then making a triangular mesh between points.
Below are the two input images that I used to find an average face


Emma's face


Juliette's face

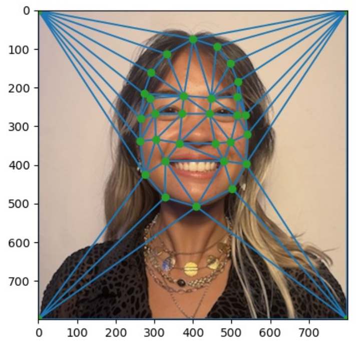
I made a GUI using matplotlib in Python to choose corresponding points

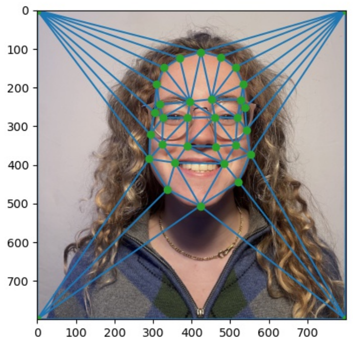

GUI to pick points on Emma's face



GUI to pick points on Juliette's face

After defining sufficient correspondences, I made a triangle mesh over them using Delaunay triangulation. This will later be used to morph the shape of each image into the shape of the average between both images. We will go over this process in the next part.


Delaunay triangulation over Emma's correspondences

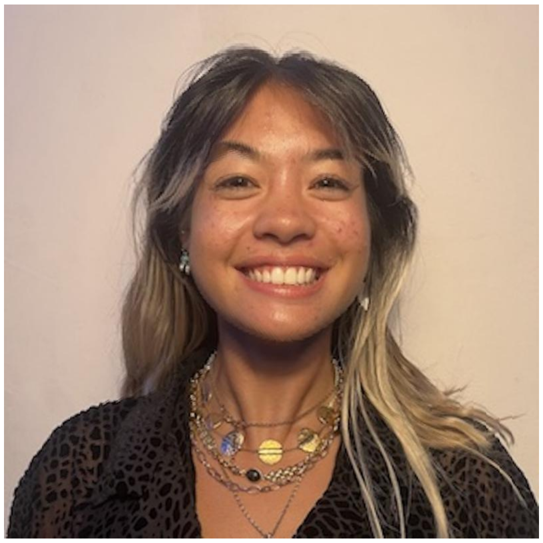
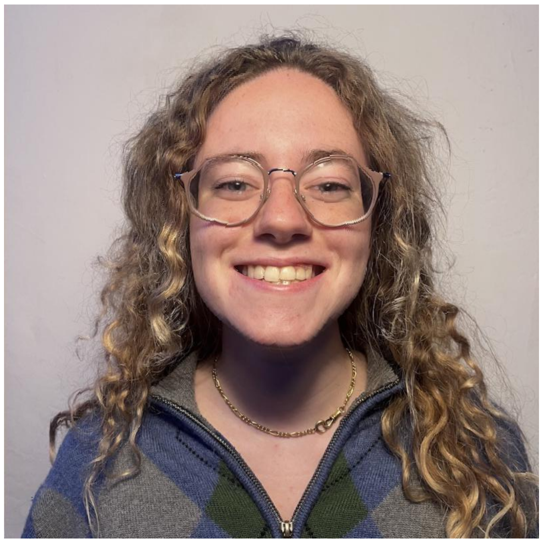

Delaunay triangulation over Juliette's correspondences

Part 2. Computing the "Mid-way Face"

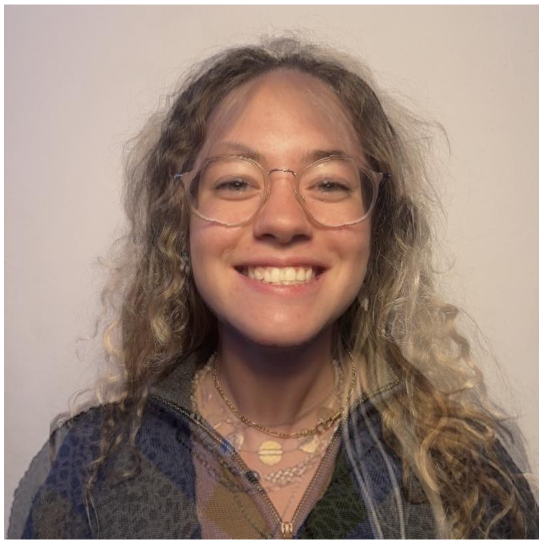
Next I used each input image's triangle mesh to warp each face into the shape of the average face.

Emma's face before warp

Juliette's face before warp



Emma's face warped into the average shape of both faces Juliette's face warped into the average shape of both faces

After this, I averaged the colors in the warped images so that the output image is the mean of the input images, both in shape and color.


Mean face

Part 3. The Morph Sequence

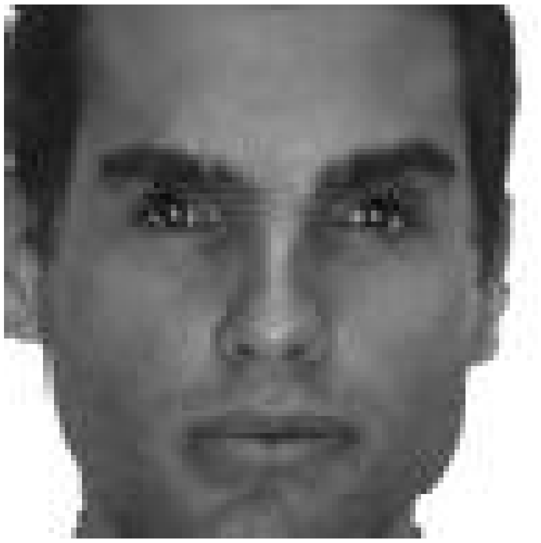
By interpolating the fraction by which we warp correspondences and mix colors from the first image to the other, we can create a smooth transition between images! We will refer to this transition as the Morph Sequence.

This Morph Sequence can be found here.

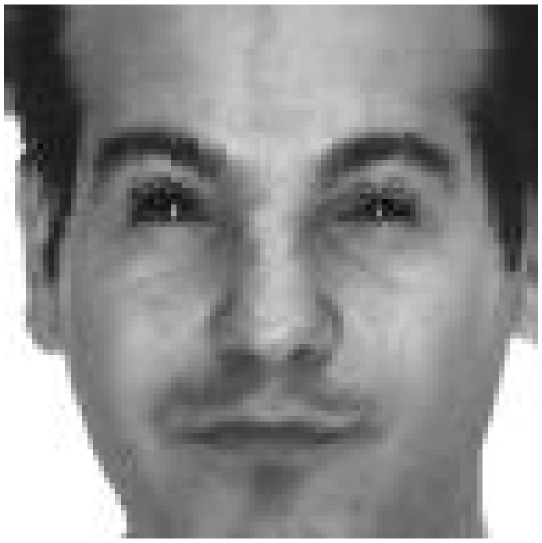
Part 4. The "Mean face" of a population

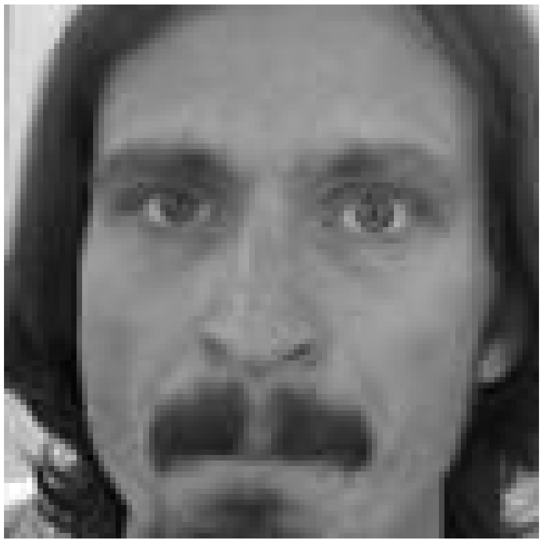
I got my data from https://www.kaggle.com/datasets/drgilermo/face-images-with-marked-landmark-points?resource=download

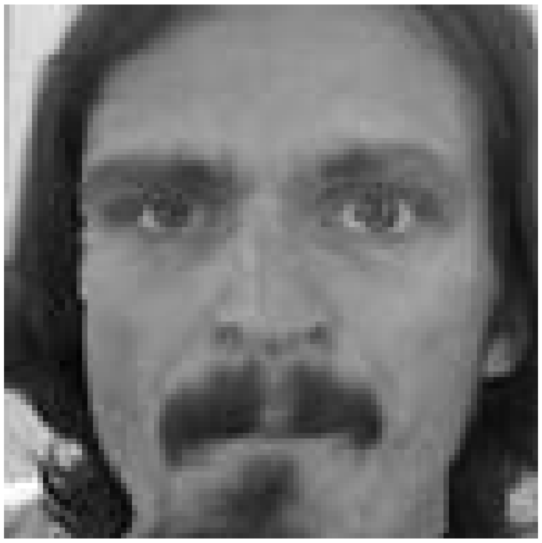
Face 1


Face 1 morphed into the average shape

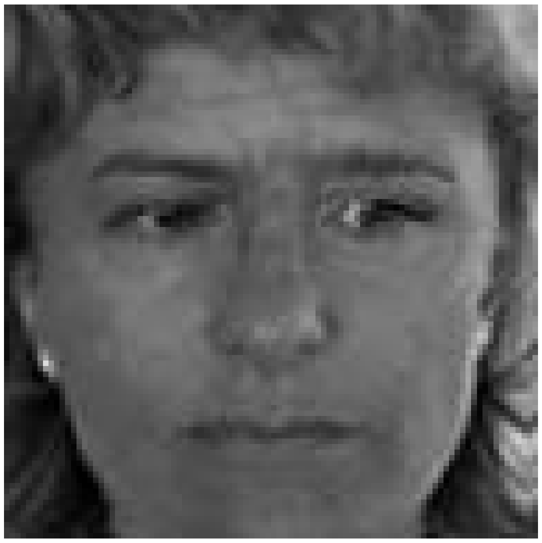
Face 1

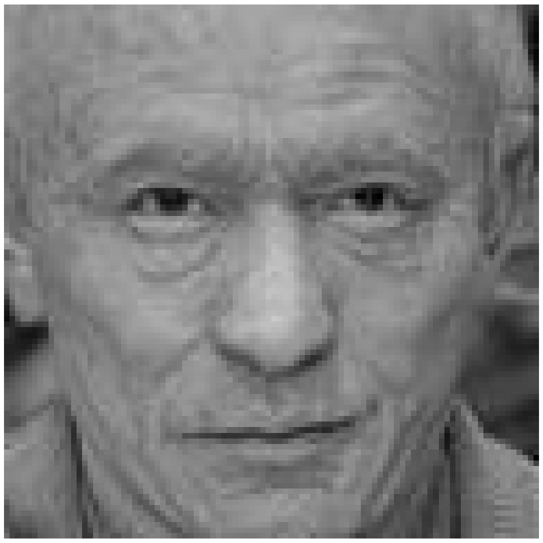

Face 1 morphed into the average shape

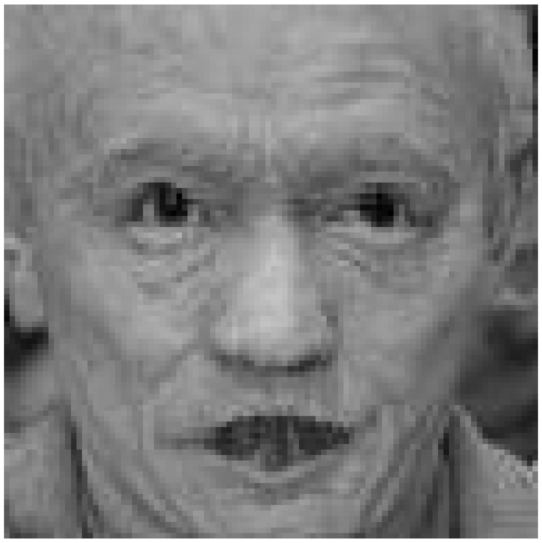

Face 1


Face 1 morphed into the average shape

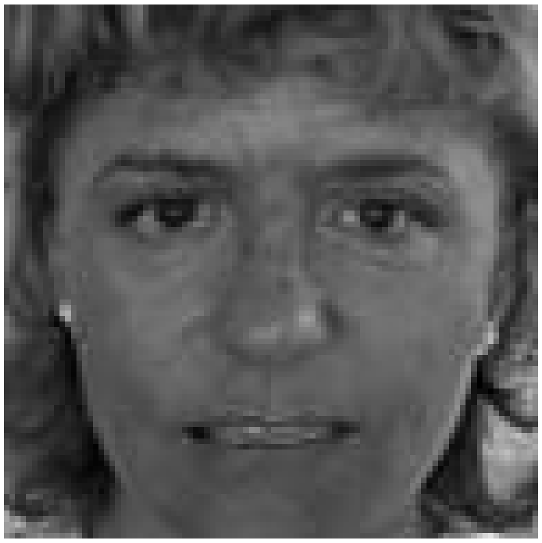
Face 1

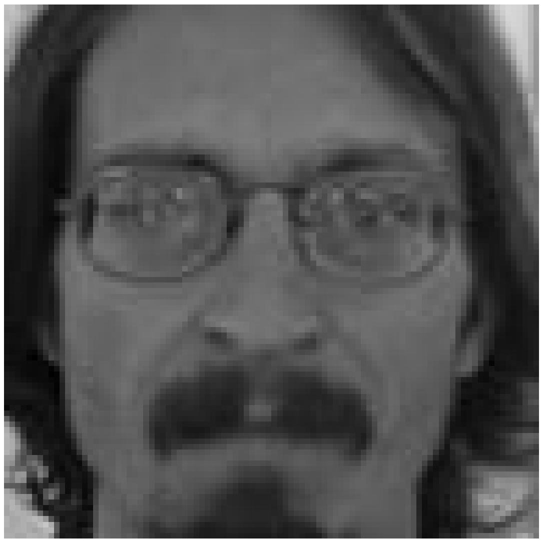

Face 1 morphed into the average shape

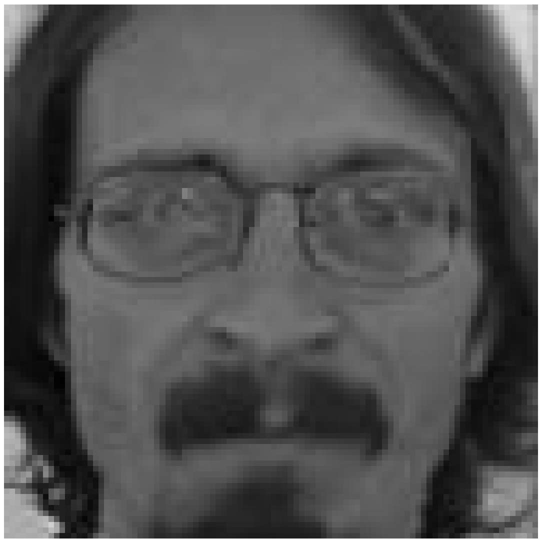

Face 1

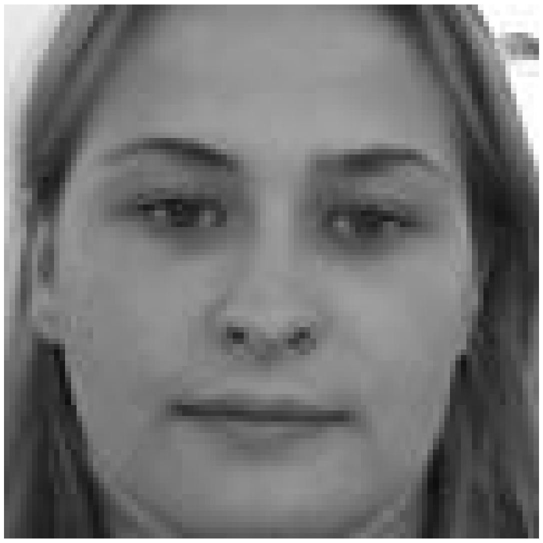

Face 1 morphed into the average shape

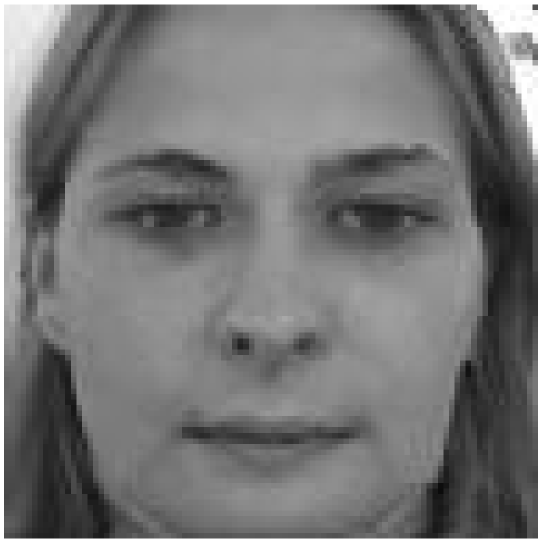
Face 1

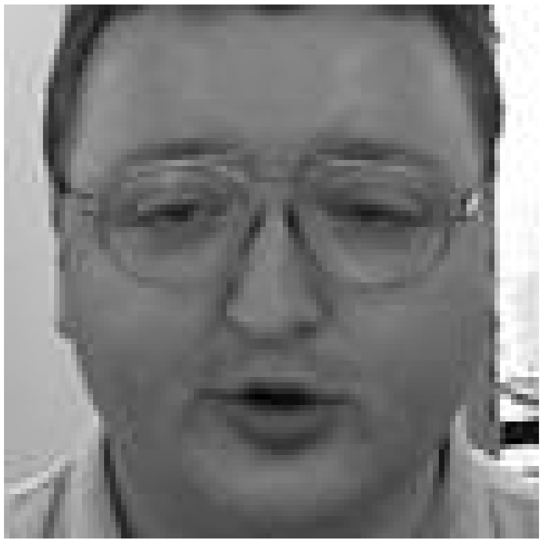

Face 1 morphed into the average shape

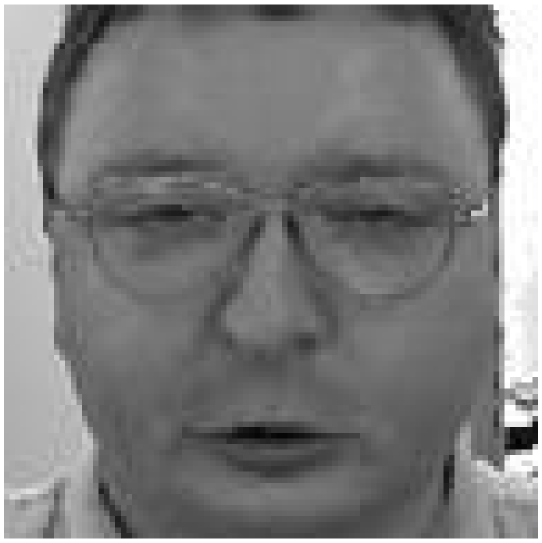

Face 1


Face 1 morphed into the average shape

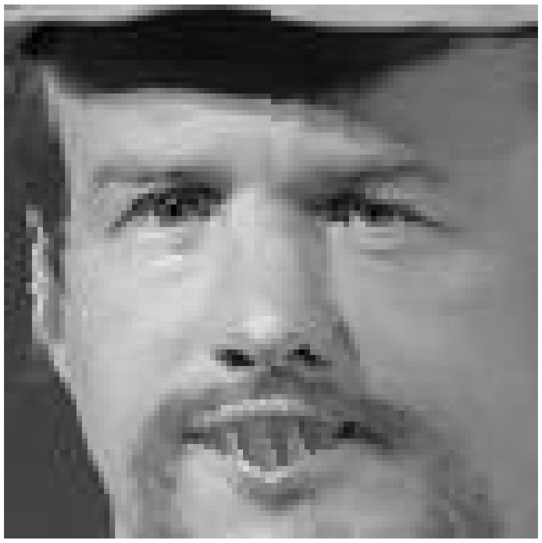

Face 1


Face 1 morphed into the average shape

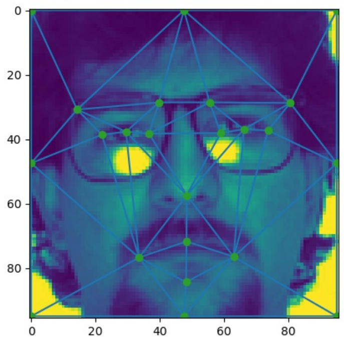

Face 1


Face 1 morphed into the average shape

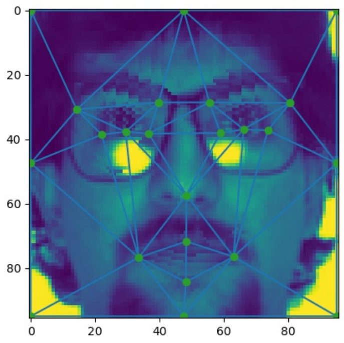
Face 1


Face 1 morphed into the average shape

When warping my examples, I noticed that bilinear sampling left weird artifacts when working with low-resolution images:

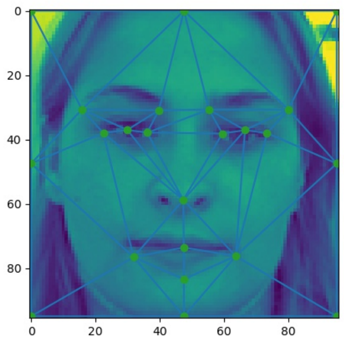


Before bilinear sampling on low-res image

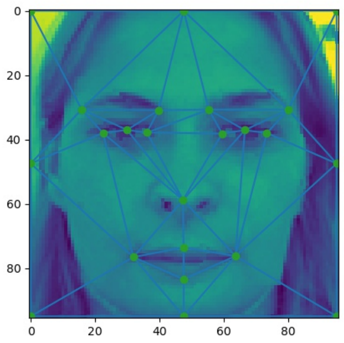


After bilinear sampling on low-res image

Because of this, I chose to also implement nearest neighbors sampling so that we could get good results on low-resolution images too!

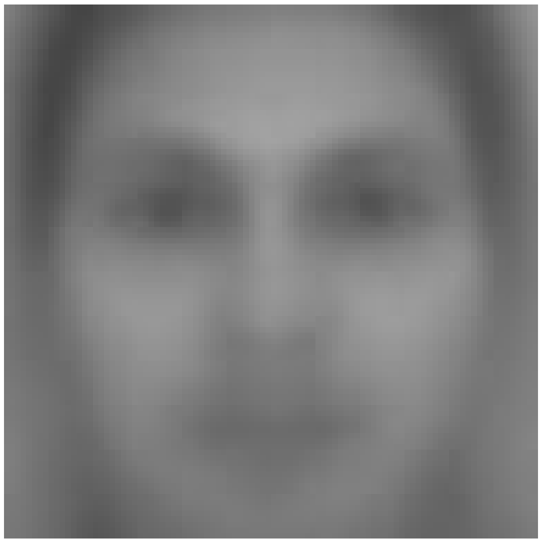


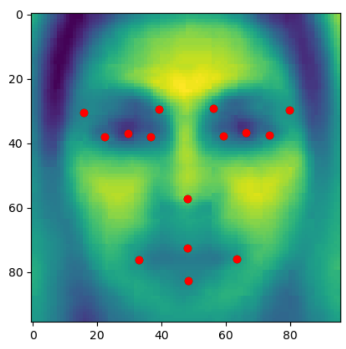
Before nearest neighbors on low-res image



After nearest neighbors on low-res image

I morphed all the faces in the dataset together to compute the average celebrity:

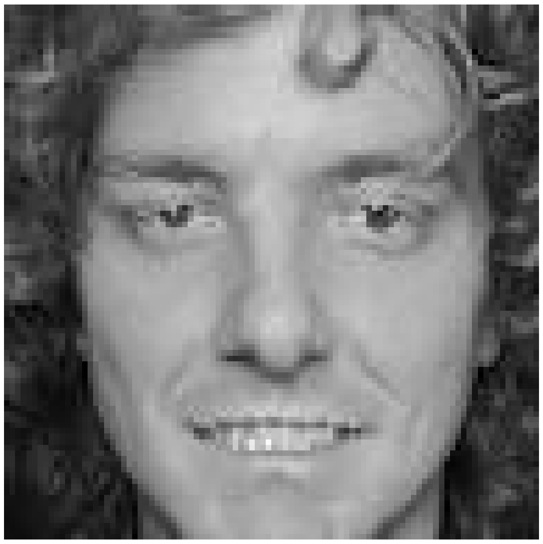

Average Celebrity



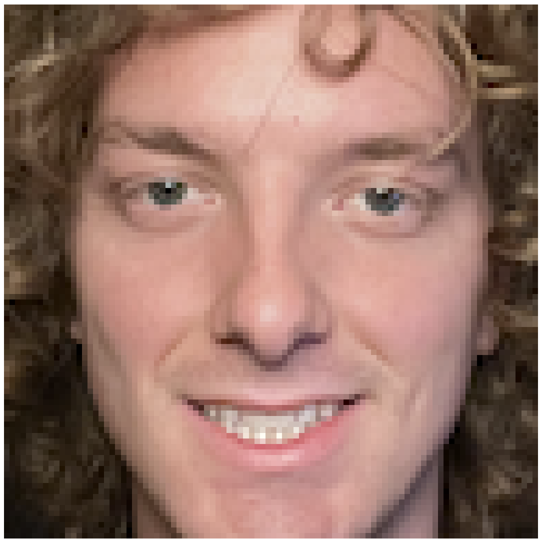
Average Celebrity with average geometry shown

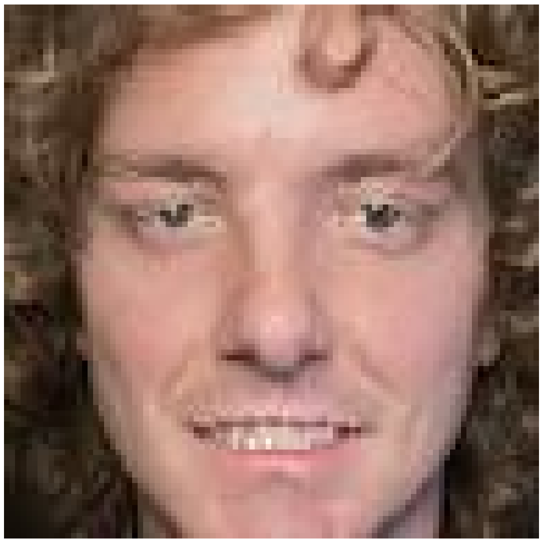
Here I warped my face into the geometry of the average celebrity:

My Face

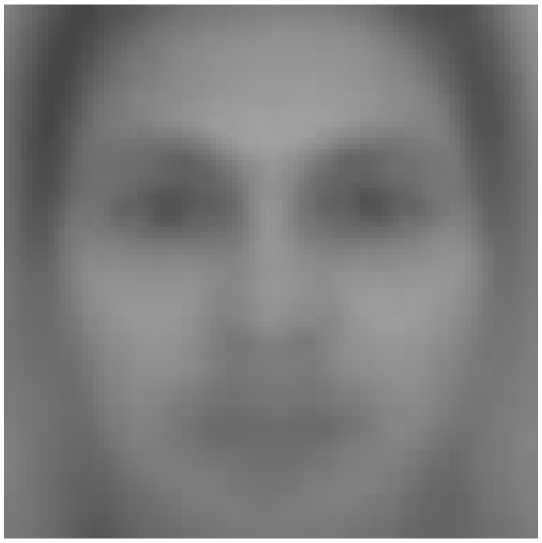

My face warped into the average celebrity's geometry

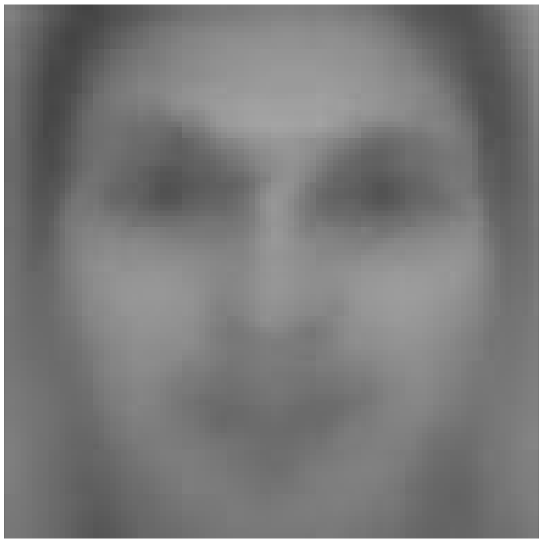
The dataset only had greyscale images, but I wrote my algorithm to work for full color images. As proof, here's my face warped into the average celebrity's geometry in full color:


My Face in color


My face warped into the average celebrity's geometry

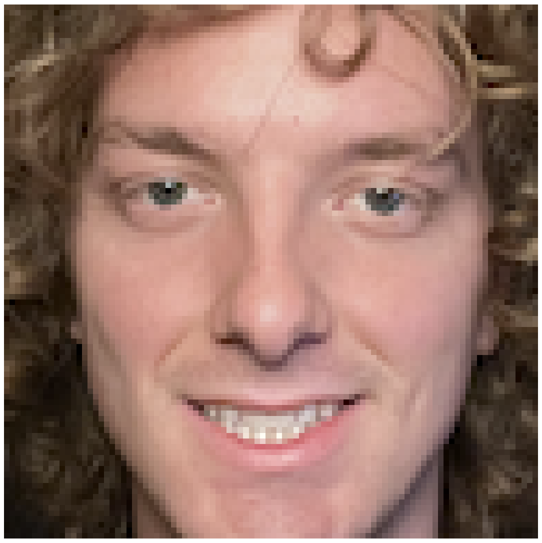
Here I warped the average celebrity into the geometry of my face:

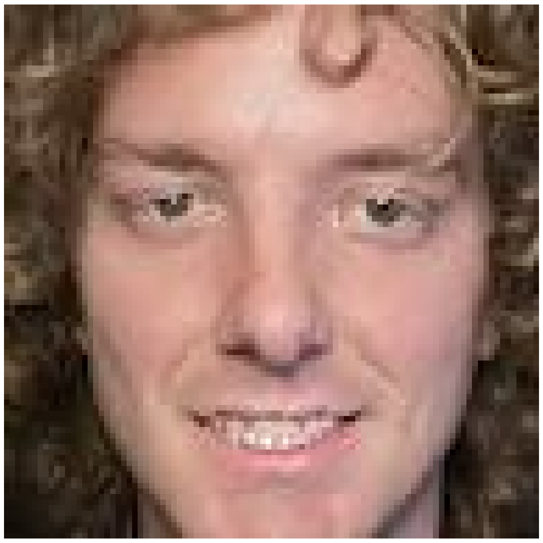

Average Celebrity

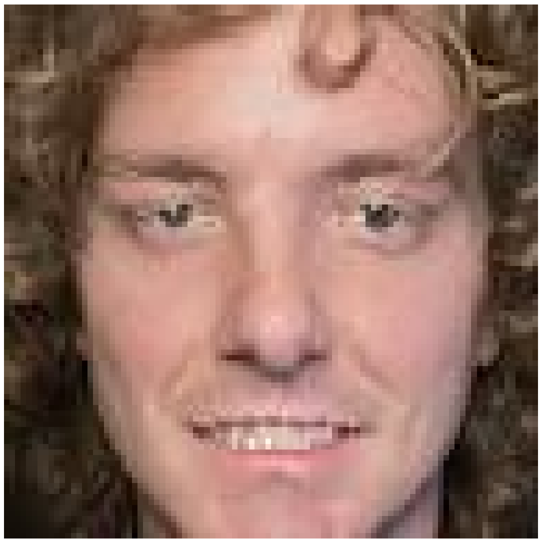

Average celebrity warped into my face's geometry

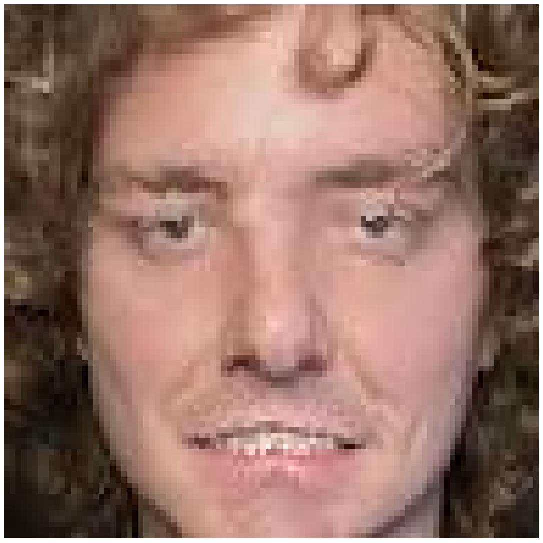
Part 5. Extrapolating from the mean

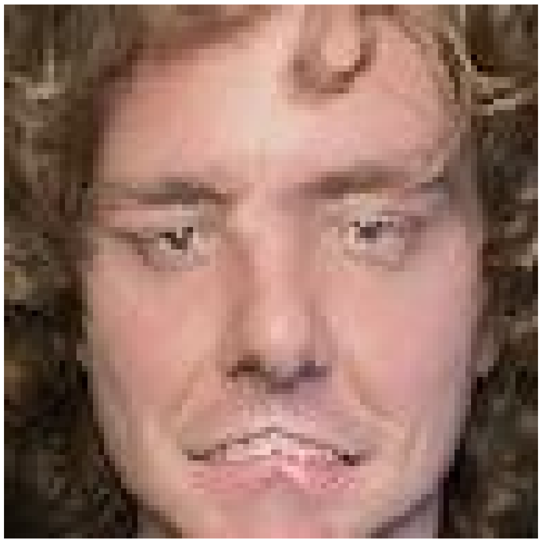
Using the difference between my face and the average celebrity's face, I can overexaggerate the difference between my features and that of a celebrity. This will result in a "larger than life" image that will intensify characteristics of celebrities that I don't have. Here's an example below.

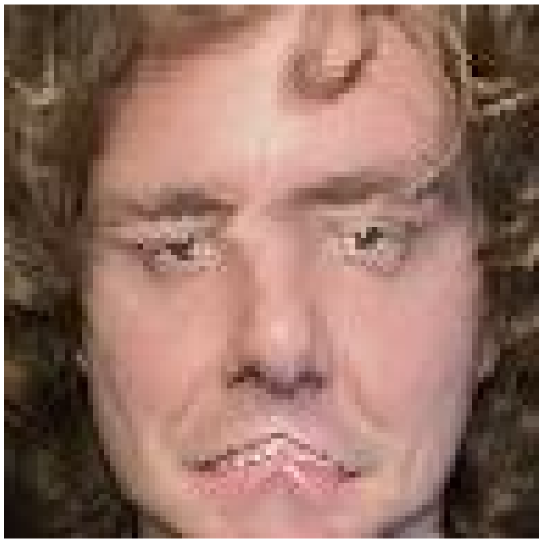

alpha = 0

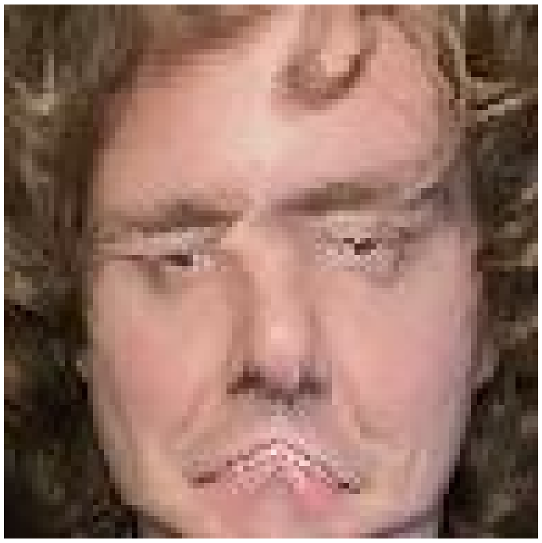

alpha = 0.5

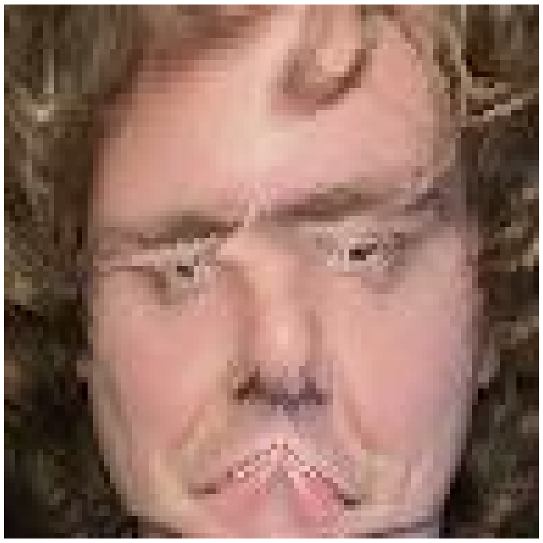

alpha = 1.0


alpha = 1.5


alpha = 2.0

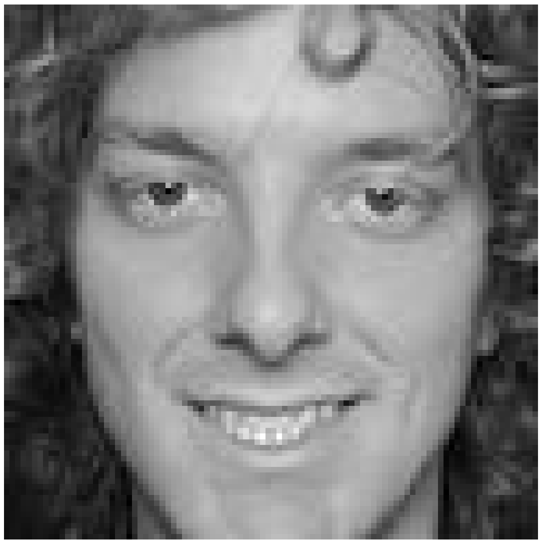

alpha = 2.5

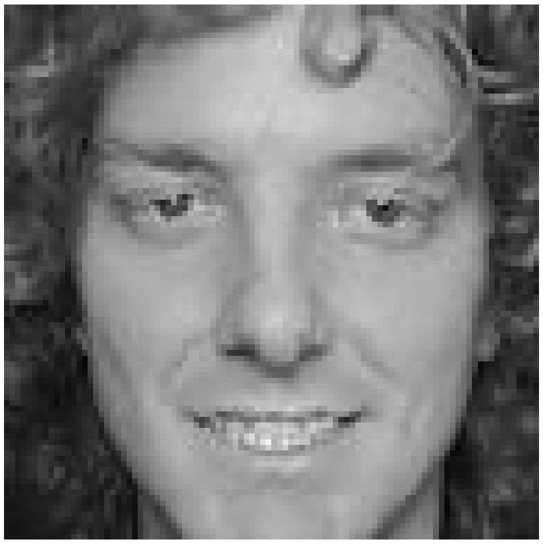

alpha = 3.0

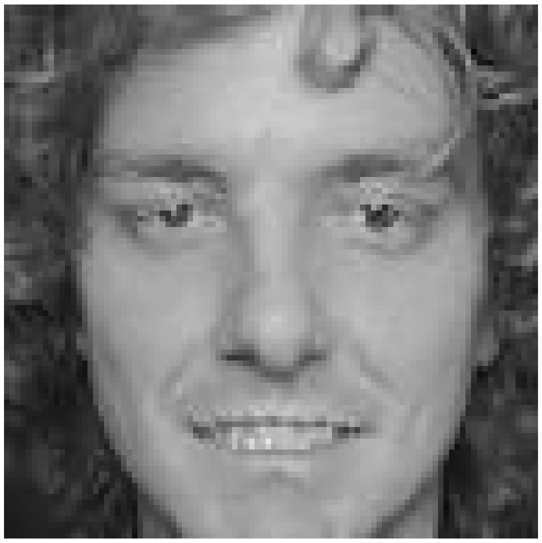

alpha = 3.5

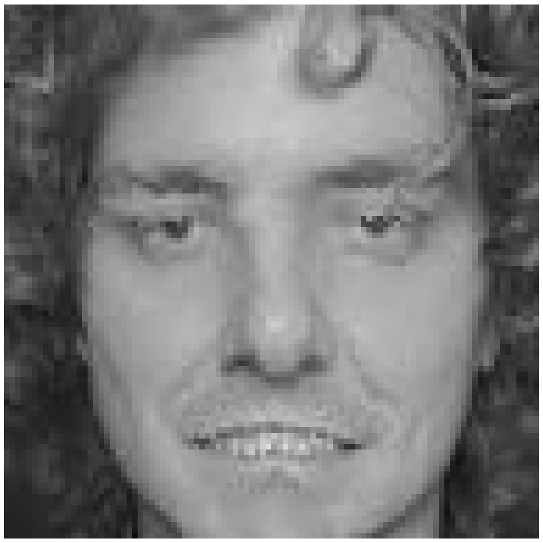
As you can see, it seems like I'm happier than the average celebrity!

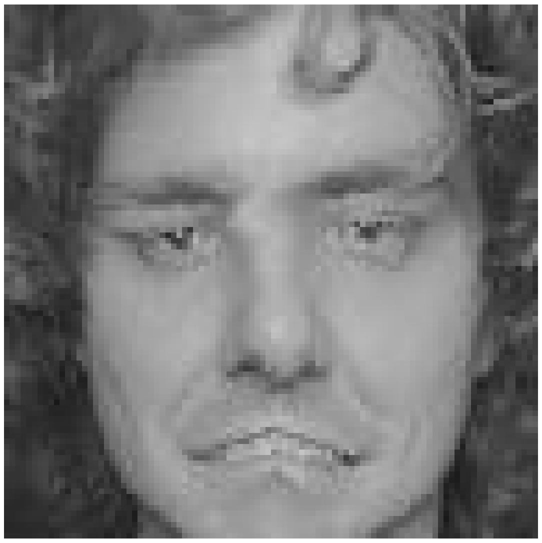
The dataset I used was in greyscale, so I have to convert my face to greyscale if I also want to extrapolate by color. Here I mix colors by a fraction proportional to alpha. This makes it so my face is exaggerated in both shape and color!

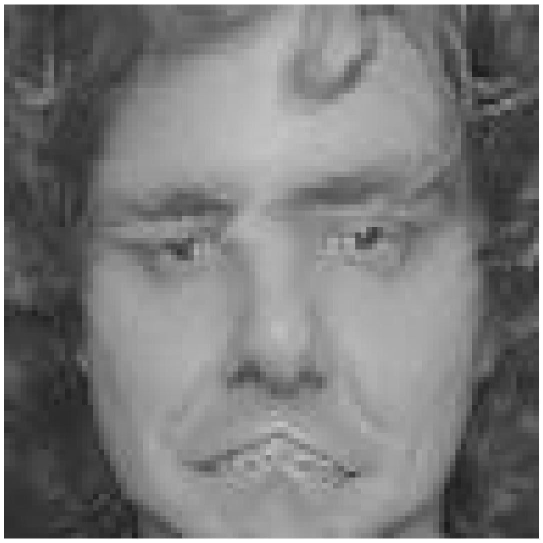

alpha = 0

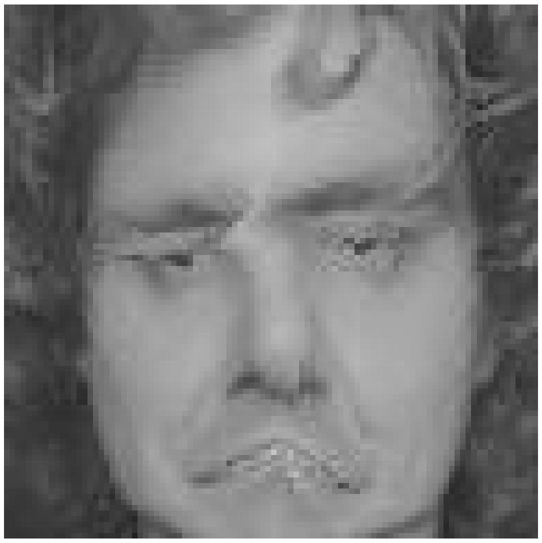

alpha = 0.5

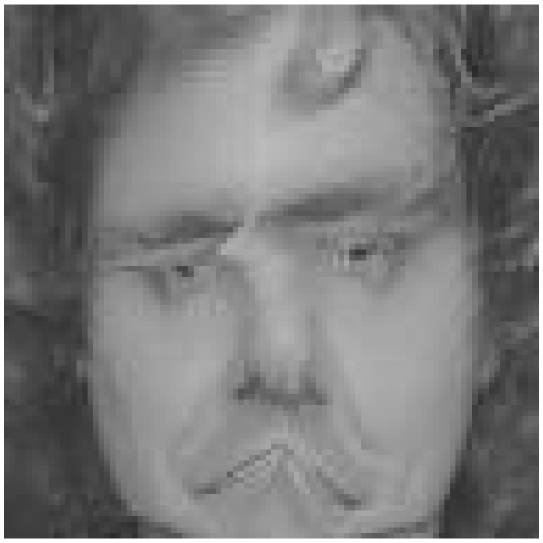

alpha = 1.0


alpha = 1.5

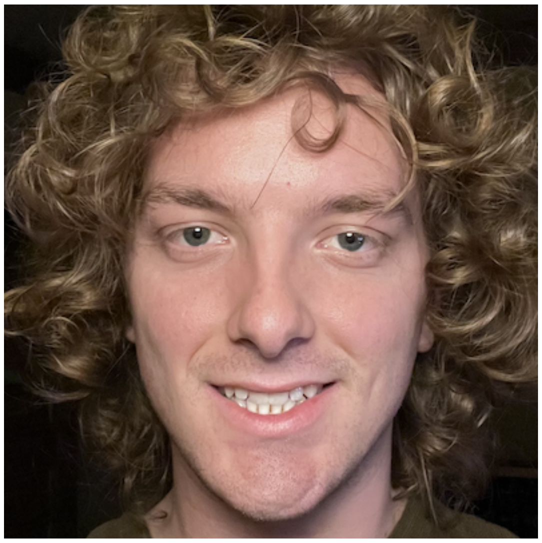

alpha = 2.0

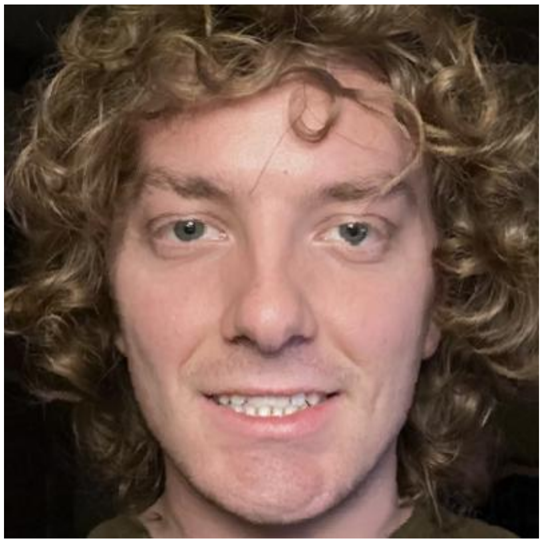

alpha = 2.5

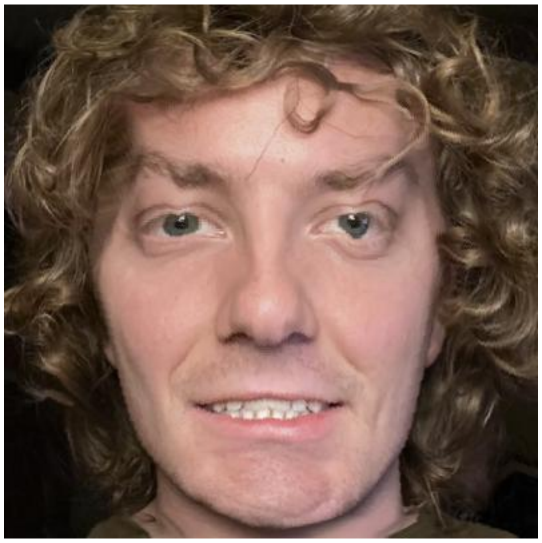

alpha = 3.0

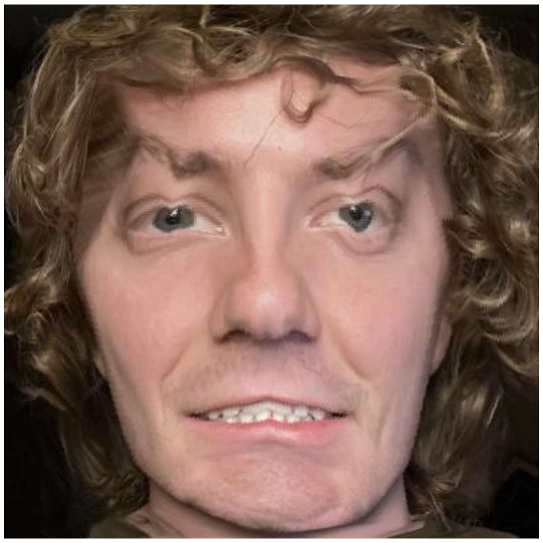

alpha = 3.5

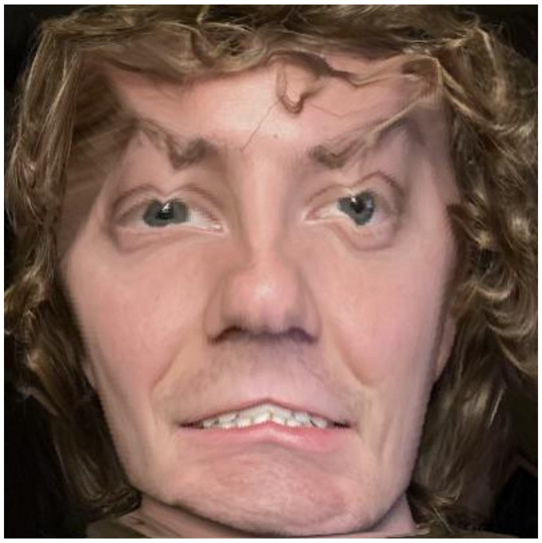
Unfortunately, the dataset I chose was very low resolution and lacked a high number of correspondences. This made it so my results were underwhelming even though my algorithms can produce beautiful morphed images as seen in the GIF above. Fortunately, the average celebrity picture gives me lots of data to use to predict average features of the dataset. I decided to upscale the image of my face and use estimated average features to show how my face would looked warped into the shape of the average celebrity with more correspondences. Here are my results:

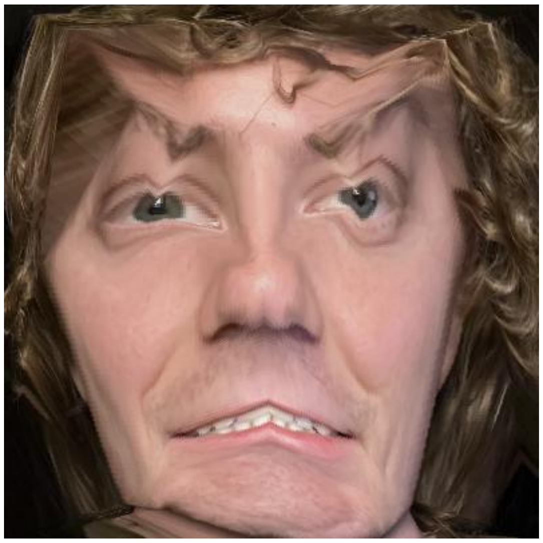

alpha = 0

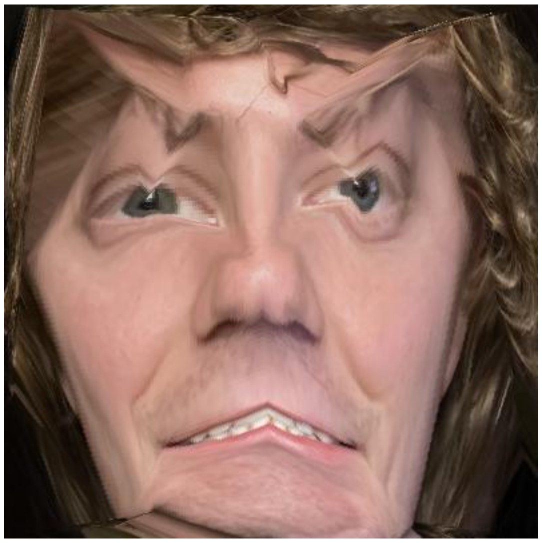

alpha = 0.5

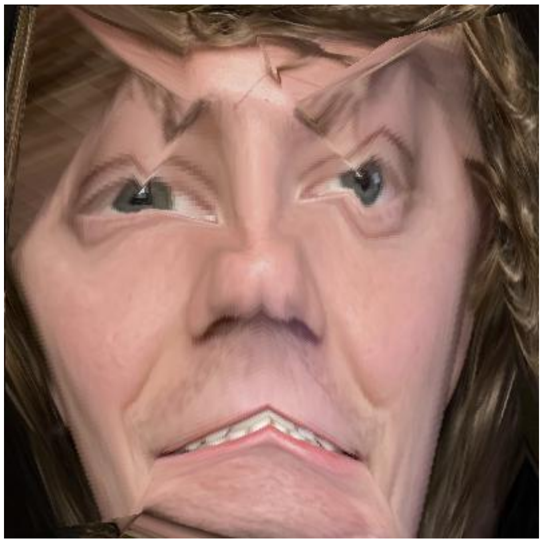

alpha = 1.0


alpha = 1.5


alpha = 2.0


alpha = 2.5


alpha = 3.0


alpha = 3.5

Part 6. Bells and Whistles

I made a music video that can be found here

https://inst.eecs.berkeley.edu/~cs194-26/fa22/upload/files/proj3/cs194-26-ahn/