CS 180 PROJECT 3

Xavier Plourde

Project Overview

This project involved computing and display several facial transforms, by using Delaunay triangulation and affine transformations between different faces for each triangle. It was a very interesting project and I had a lot of fun applying the techniques taught in the lectures to images of my face and others!

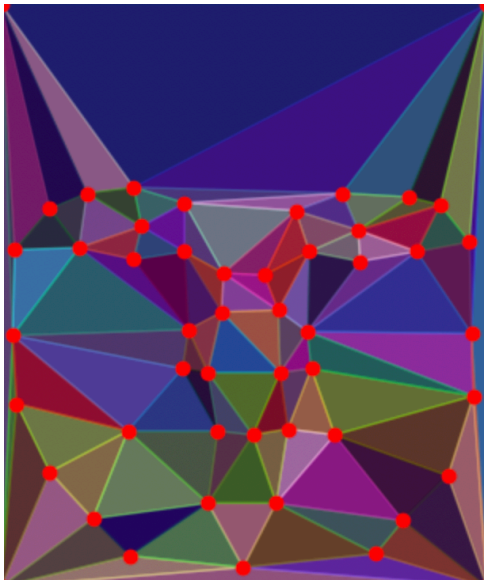
Part 1 - Defining Correspondences

Overview

For this part, I used the labeling tool linked in the website (https://cal-cs180.github.io/fa23/hw/proj3/tool.html) to generate correspondences for my face and for the image of George. After generating correspondences for my face, I computed the Delaunay triangulation of the points for each image, using the scipy.spatial.Delaunay library function.

Results

Image of my face



Delaunay triangulation of my face, with correspondences marked in red

Part 2 - Computing the Mid-Way Face

Overview

Next, after computing the Delaunay triangulation for each image, I used the algorithm taught in class to compute the mid-way face. Specifically, for each triangle in the image, I used an affine transformation to map it halfway to its corresponding triangle in the other image, making the pixels an average of the two images' values from each triangle.

Results

Image of my face

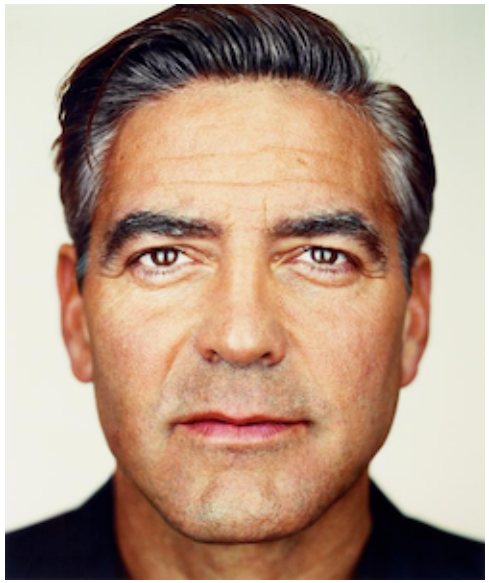


Image of George from website

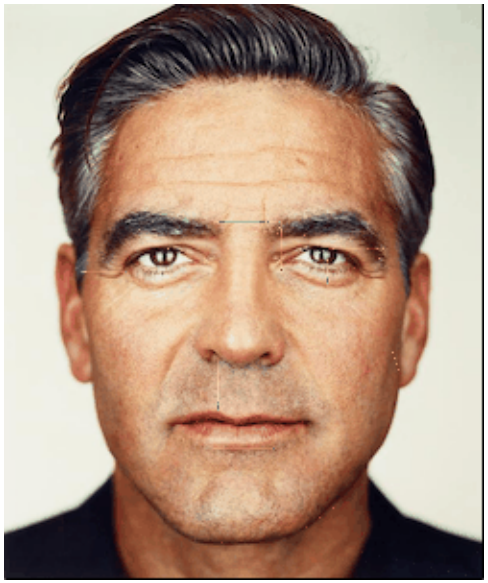
Midway face of morph sequence

Part 3 - The Morph Sequence

Overview

After computing the mid-way face, I extended this to compute the full morph sequence. I accomplished this by using a parameter "t" corresponding to what percentage of the image is George's face, and which percentage is my face. For example, t=0 is George's face, t=1 is my face, and t=0.5 is the mid-way face. I generated a gif of 30 successive values of t ranging from 0 to 1, inclusive (followed by the 30 images in reverse to provide a smooth, continuous sequence).

Results



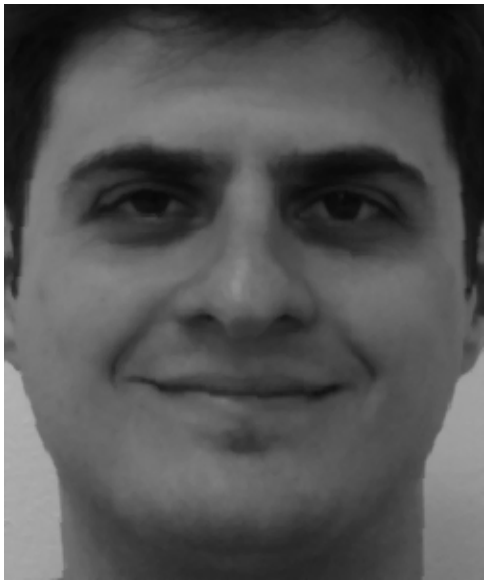
Full morph sequence

Part 4 – The Mean Face of a Population

Overview

For this part, I used the FEI face database, with 200 faces with keypoints already defined. I used the keypoints to align each face to the "average" shape, and compute the overall average. Then, I used the same correspondence tool from part 1 to map my keypoints in the same way as the dataset images, and transformed my image to the average geometry, as well as the average image to my geometry.

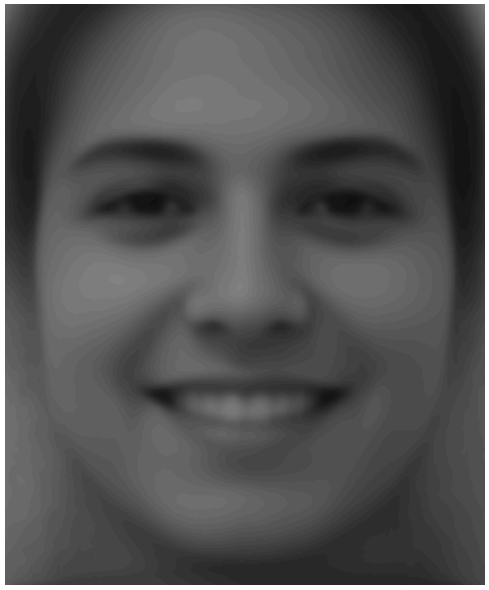
Results



Dataset image example

Dataset image example

Dataset image example



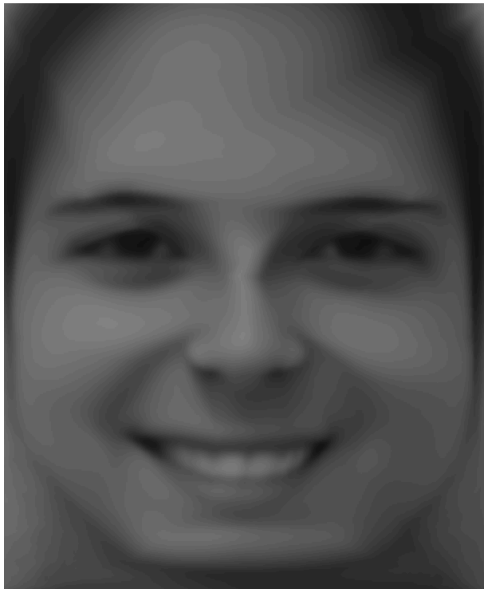
Dataset average image

Image 1 warped to average

Image 2 warped to average

Image 3 warped to average

My face warped to average geometry



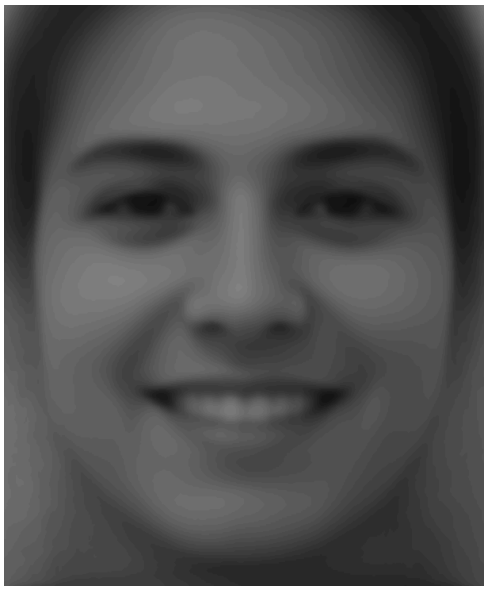
Average geometry warped to my face

Part 5 – Caricatures

Overview

For this part, I took the image of my face (converted to black-and-white to match the dataset) and the dataset average image from the previous part. I then applied the same process from parts 2/3 to the dataset average (t=0) and my face (t=1), but set t outside the 0-1 range; specifically, I found that t=1.3 gave the best result. This resulted in a caricature image - exaggerating the features of my face that stand out from the dataset average image.

Results



Dataset average image

My face (black-and-white)

Caricature of my face with t=1.3

Bells and Whistles

Overview

For this part, I chose to use the existing dataset images I had in order to change the expression of my face in a morph sequence. More specifically, the dataset I used contained both 200 images of people smiling, and 200 images of people with a neutral expression (two images per person). Using this data, I computed both an average image of smiling people, and an average image of non-smiling people. Then, I applied a similar technique from part 4 to morph my geometry to both the smiling average and the non-smiling average, and generated a morph sequence between these two altered images of my face.

Results

Average smiling dataset image

Average non-smiling dataset image

Full morph sequence