Ryan Miller
u1067596
Professor Whitaker
Digital Image Processing

Project 4 - Image Morphing/Atlases

Note: Please refer to the "ReadMe.txt" file submitted with the project for details on running the python scripts

Introduction

In the third project of CS 6640, I experimented with image morphing and atlases formulated from geometric transformations and warping. In this report, I will present two different tasks: forming image morphs between two images and forumatling atlases between multiple images. Both tasks utilize correspondences strategically located in the input images that act as control points to define the transformation between the original images and the output. In both tasks, a radial basis function is used to carry out scattered data interpolation. In the image morphing, the output image is formulated by combining the shape and intensity of the two input images by means of a ratio to create a blended output. In the image atlas, the simple average of the input images is taken and statistical analysis is performed on the resultant to deduce information about the various samples used as inputs. For each task, I will first give a bit of background of the theory behind the analysis being carried out on the images before explaining how I implemented the algorithm in python. A variety of images, including inputs and outputs, will be presented in this report to demonstrate the ability of each algorithm. Each of the output images can be reproduced using the python scripts. After all tasks have been presented and analyzed, credit will be given to other individuals who have made this analysis possible.

Task 1 - Image Morphing

The first task in this report analyzes image morphing as a means of creating an output image based on a pair of image inputs with corresponding correspondances. The morphing routine takes the coordinates of each image pair and transforms one image to the other before morphing the two together using a parametric ratio equation, representing the overall ratio of one image compared to the other in the final image output. To carry out this operation, a transformation is formulated to map the points in one image to the other, creating a result with overlapping correspondances. The problem of finding this transformation then becomes a problem of interpolating the correspondence offsets such that the transformation is smooth. In this example, I utilize a thin plate spline function that has been shown to minimize thin plate bending energy. This function incorporates the use of a radial basis function positioned radially around the correspondence points. The problem then becomes a task of solving the linear system of

equations whose solved coefficients are used to formulate the x and y smooth transformations of the input x and y coordinates.

The input files for the algorithm I designed is a json file which specifies to the algorithm the two input images to be morphed as well as the correspondences for each image. Two scripts are used in this task: the first script is simply used to extract all the necessary information from the input json file and the second script is used to formulate the output morph. Such details that were deemed important and extracted by the formatting script include: number of correspondences, the actual correspondances in each image, and image names.

The algorithm starts by formulating the correspondences in the final canvas image to aid in formulating the transformations between both of the input images and the canvas. The canvas correspondences are set simply by taking the average in the x and y dimensions of the two input images. The next step is to find the dimensions of the canvas by transforming the input images to the canvas. To do this, I use the correspondances on the original image and the correspondances created for the canvas to define the linear systems describing the forward transform, from the input images to the canvas, and the reverse transforms from the canvas back to the input images. Various functions are defined that create the linear system of equations to solve for the coefficients needed to formulate the transformations between images. Singular Value Decomposition is used to solve the linear system of equations for these coefficients. I utilize a function definition to define the transformation such that x and y coordinates can be passed into the function and the output is the transformed coordinates based on the input coefficients for that specific coordinate transformation.

From the transformation of the input images to the canvas, the max and min dimensions of the output file (the canvas) are defined. The reverse transformation is then defined using the same functions (but with different) inputs as the forward transformations. A (so I thought) clever way to store the transformation is through the use of a matrix of lists. The original matrix is comprised on lists of the coordinates. For example, the list in position x = 0 and y = 0 is [0,0]. The matrix can then be passed into the transformation function and the function extracts the list elements and replaces them with the transformed coordinates. Although this method worked very well and is neat in storing the transformation (as will be seen with more images in the atlases), it was slightly inefficient computationally.

After the transformations from the input images back to the canvas were found, I used the scipy interpolator object to interpolate the final intensities of each input image to be morphed. THe final step was then declaring the canvas pixels as a ratio of the two images to view the results. The result of the sample example given to use using the images provided is presented in Figure 3 and the original inputs for the given morph output are presented in Figures 1 and 2.

It must be noted that to deal with the possible difference in intensities of the input images, both images undergo histogram equalization before they are used to formulate the output. SInce we want to be able to control the ratio of intensities via our parametric equation, we want to deal with the possibility of different images that have different intensity values when evaluating a variety of examples.

Figure 1: Input 1 for morphing 1 example

Figure 2: Input 2 for morphing 1 example

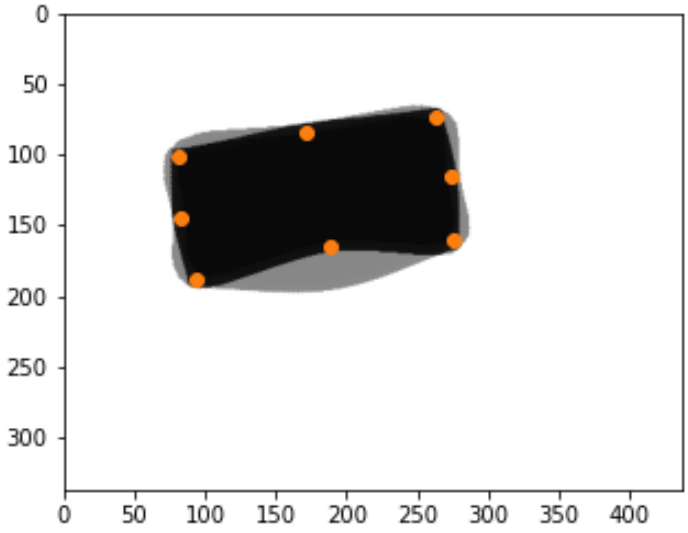

Figure 3: Output of Morphing algorithm created from Figures 1 and 2 as inputs (t=0.5)

Figure 3 presents the output of the morphing algorithm with a blending amount of 0.5, meaning each image has equal effect on the output intensity shown in Figure 3. All in all, the algorithm appears to perform fairly well. The correspondences that are shown in the output image are that of the canvas and it is clear that the correspondences in each individual image are mapped correctly to that of the canvas as intended. It must be noted that in this example, the number of correspondences used is 8. To analyze the effects of this number, I experiment with increasing and decreasing the number of correspondences.

In this first experiment, I use the same images as presented above but increase the number of correspondences by 2, adding two more along the tops of the shapes in between the middle and the top left and right corners. The output is presented in Figure 4. From its appearance, it is clear that increasing the number of correspondences drastically increases the overall performance of the algorithm as some portions of the morph presented in Figure that are not perfectly overlapping are better overlapping in Figure 4. The overall accuracy of the output is increased but the intensity itself is unaffected. To find the exact coordinates of the correspondences, a photo editor is used to find the pixels desired to be used as correspondances. Since the overall number of pixels is very large, the correspondances end up in slightly incorrect places although it does not appear to have a drastic effect on the outcome.

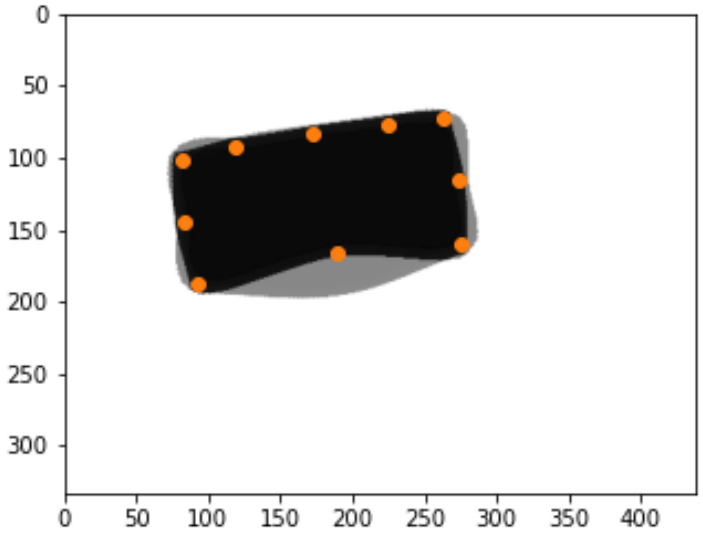

Figure 4: Increasing Number of Correspondences to 10 (t=0.5)

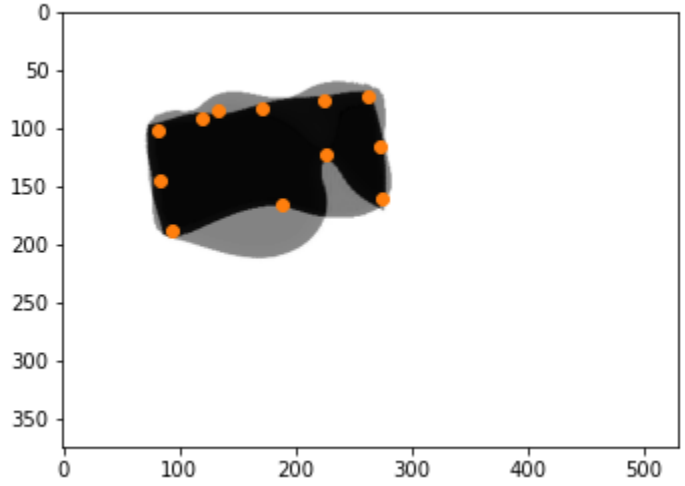

Figure 5: Number of Correspondences = 12 but with error in placing

In attempting to place 12 correspondences, I made an error in placing the correspondences in the first input image and the result is shown in Figure 5. It has a drastic effect on the overall accuracy of the image and also affects the output intensity as the amount of overlap between the input images is affected. Another example of this mistake is shown in Figure 6.

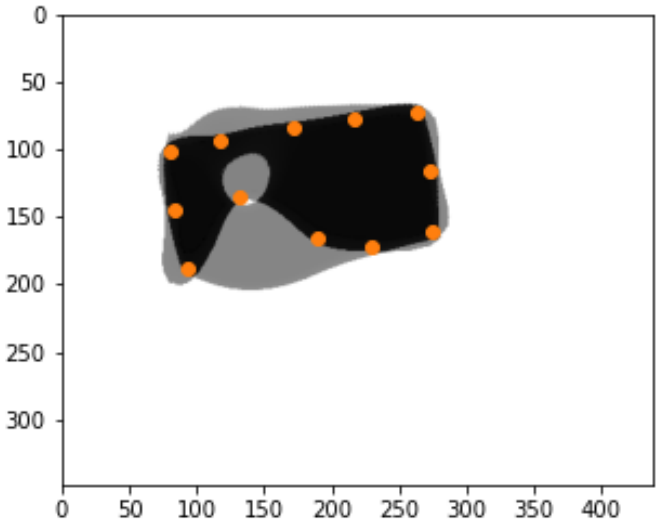

Figure 6: Correspondence that is not like the rest formulated from mistake (missing the 1 in 194 makes the correspondance at 94)

In Figure 6, when inputting correspondances, I forgot to add the 1 in 194 so the resulting correspondence is 94 instead of 194 as intended. Again the intensity is affected because the amount of overlap is affected.

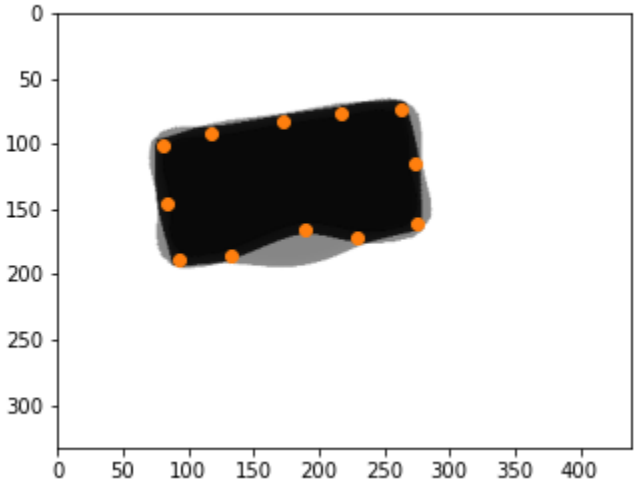

Figure 7: Morphing from 10 correspondances

From Figure 7, we can see that even one correspondence that is slightly off will pull the morph in that direction. Thus, I conclude that going forward, I must be very careful in setting the correspondences. Shapes that have hard edges should have correspondances at their edges to produce results that are desirable. To show what happens when the number of correspondences decreases, the same input images are morphed using 4 and 6 correspondences.

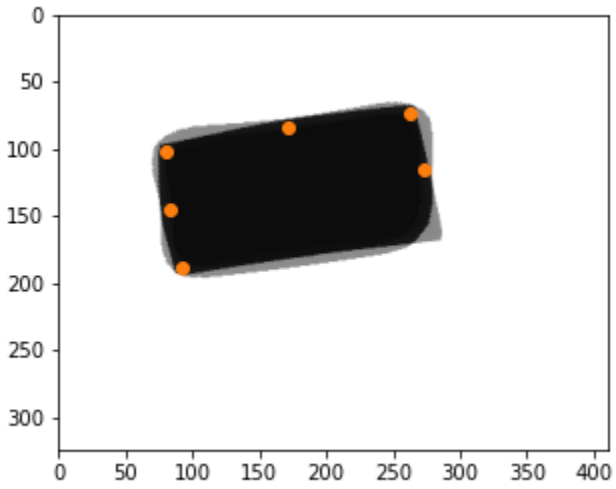

Figure 8: Number of correspondences = 6

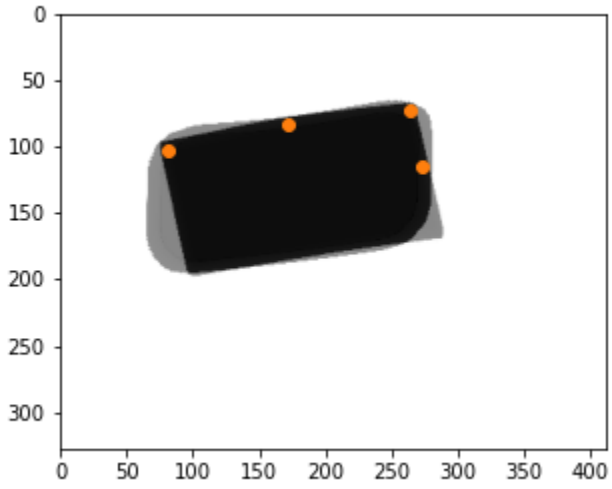

Figure 9: Number of correspondences = 4

When decreasing the number of correspondences, it is clear that the morphing is slightly less accurate especially at the correspondence points, but it performs better than expected. The

intensity is affected by the decreased overlap between the images. To present what different outputs looks like at different ratios, I present Figures 10 and 11.

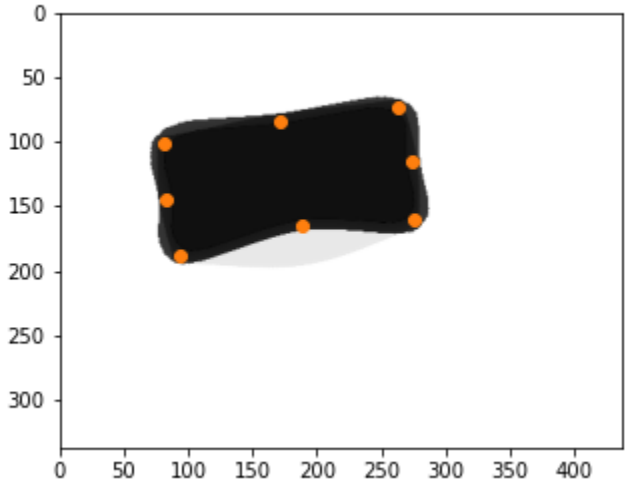

Figure 10: Morphing, 8 correspondances, t=0.1

In Figure 10, you can more easily make out both of the images and where they overlap to be the dark black. If you look around the edge, you can see the darker image taken to be 9/10 of the output and the very light image taken to be the 1/10 ratio of the output. The opposite is presented in Figure 11.

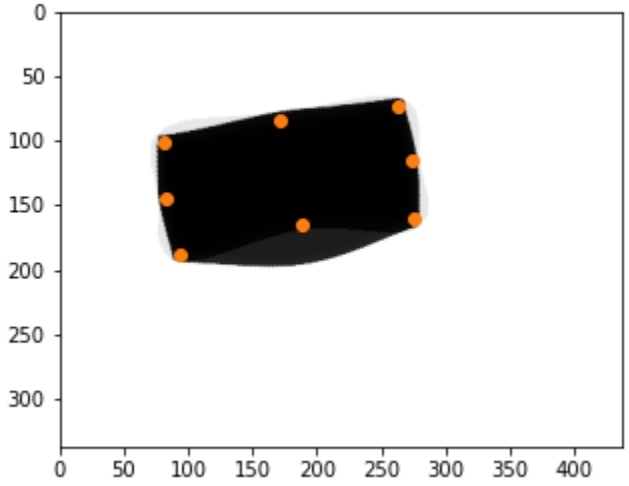

Figure 11: Morphing, 8 correspondances, t=0.9

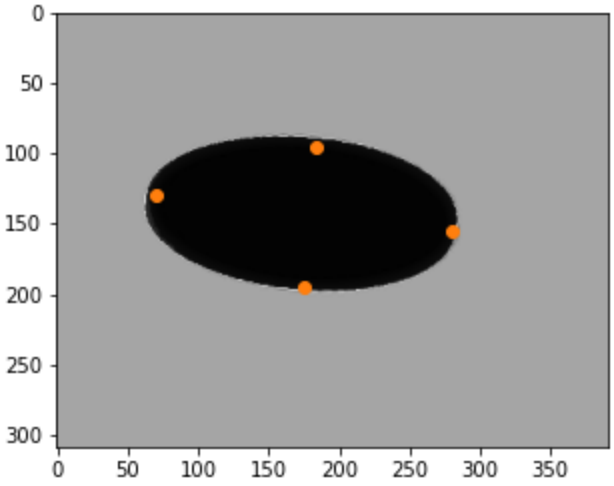

Figure 12: Morphing of ellipses

I now move on to an example not provided where I try to morph a picture of my face with that of Santa. With the holidays approaching, I thought it would be fun to try and be santa. The original images are presented in Figures 13 and 14.

Figure 13: Image input 1 for morphing (me)

Figure 14: Input image 2

In the first iteration, I set the number of correspondences to be equal to 5. I try to pick correspondances that pull out important features: the nose, both eyes, top of the head, and bottom of the chin. Since I want my face to be slightly overpowering that of santa, I set t = 0.25 so that the ratio of my face in the output is greater than that of santa. The result is shown in Figure 15.

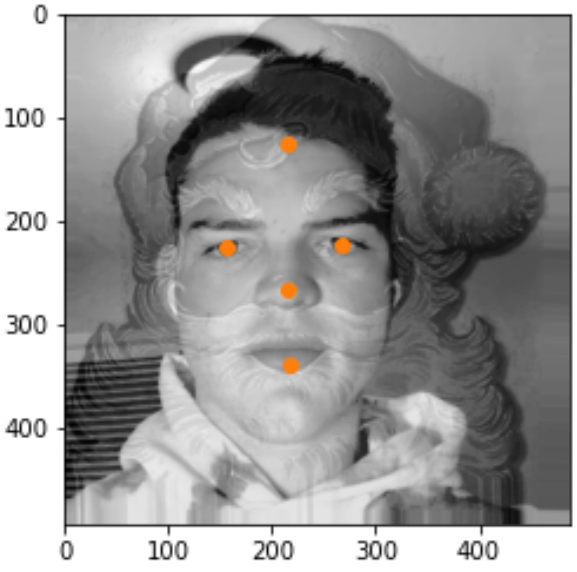

Figure 15: Iteration 1 of morphing me and santa

It appears that I am slightly overpowering santa. Looking closely, the four control points work very well to align our faces but my face is too pronounced compared to that of santa. Santa also has a fatter head and it is slightly off putting as the beard does not fit my face but the control points match perfectly. I will increase t to a value of 0.4 to allow Santa to be more pronounced in the output image.

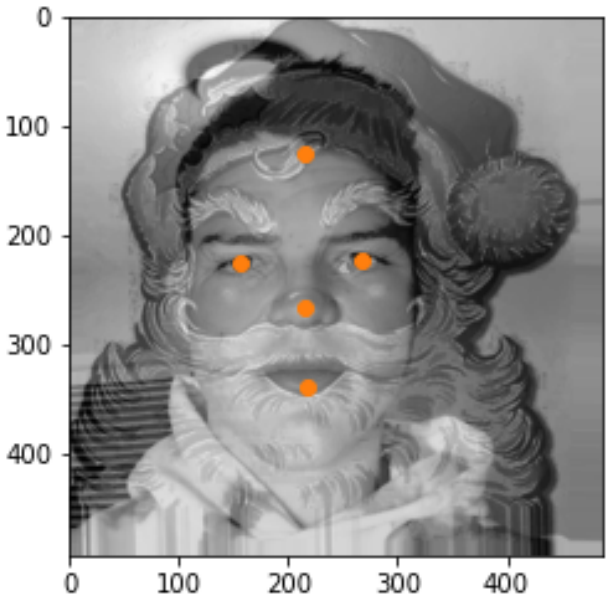

Figure 16: Morphing me and santa (t=0.4)

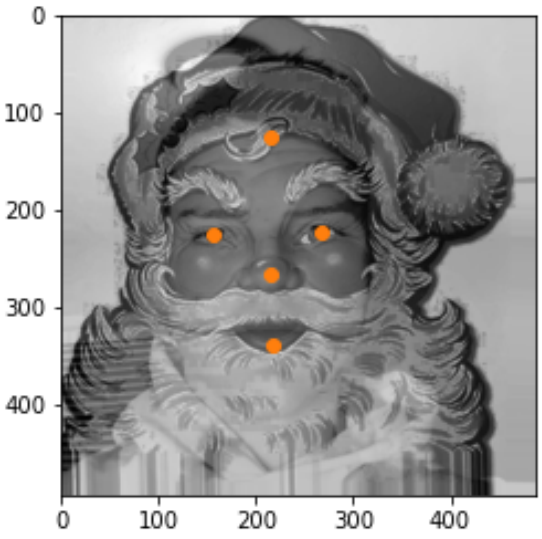

Figure 17: Morphing me and santa (t=0.6)

Since some of santa's features need to be easily distinguishable to make me look like santa, it was found that a value of t of 0.6, meaning the output is a majority of santa, gave good visually looking results. Since the edges of my face and santas are not as cut and dry as that of the shapes, the fact that the correspondences are slightly off from the exact pixel value does not affect the output as much as we saw for the previous shapes with hard edges. More control points are not needed because the ones already picked do a sufficient job of overlaying our faces. If I were to add more (as we saw previously in the case of the shapes) the local intensities would be slightly affected as the correspondances would pull and locally distort various overlaps which will have the effect of changing the local intensity of the output image at that point.

To analyze how the choice of radial basis functions affect the morphing output, I experiment with building a Gaussian kernel function to replace the current radial basis function and experiment with different sigma values for the Gaussian kernel. We expect the gaussian to act in a similar manner as the radial basis function up to some certain sigma value. The form of the Gaussian radial basis function kernel is presented in Figure 18.

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{2\sigma^2}\right)$$

Figure 18: Gaussian radial basis kernel function

This Gaussian kernel will replace the previous radial basis function used in the previous morphing experiments, presented in Figure 19.

$$\phi_i(\bar{x}) = ||\bar{x} - \bar{x}_i||^2 \lg(||\bar{x} - \bar{x}_i||),$$

Figure 19: Previously used radial basis function (thin plate spline radial basis function)

The radial basis function in our project is used to find the coefficients solved for in the linear system of equations. Knowing this, with a different radial basis function, we expect a change in the overall parameters that are solved for in the linear system of equations. The exact outcome is not completely apparent before testing, thus we wait to form conclusions until after the experiments have been carried out. When running the morphing script, the script will first ask the user if they would like to run the gaussian radial basis function or the thin plate spline radial

basis function outlined in the warp notes. If the user selects a gaussian rbf, the script will then ask the user for a sigma value to associate with the computation.

To see the effect of the gaussian radial basis function on the resulting output images, the gaussian RBF was first run on the morphing of myself and santa with a value of t being 0.6 (to compare with Figure 17). The first experiment (sigma = 0.25) is presented in Figure 20.

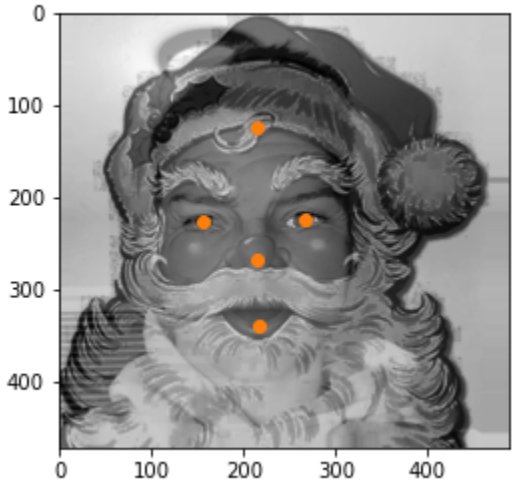

Figure 20: Morphing using t = 0.6 with Gaussian RBF (sigma = 0.25)

As presented in Figure 20, the Gaussian RBF has very subtle changes in comparison to the regular basis function. I notice that for the given sigma, the output image is slightly widened. Another very noticeable difference is that the Gaussian RBF does not produce the unwanted smearing that occurs near the bottom of the image in Figure 17. Let's analyze the effect of changing sigma to 0.75.

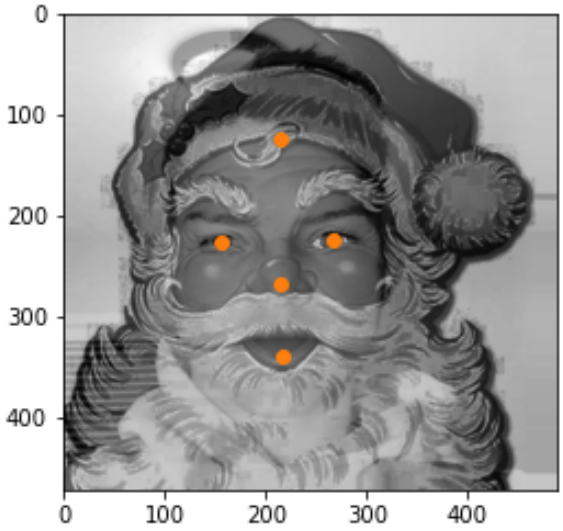

Figure 22: Morphing using t = 0.6 with Gaussian RBF (sigma = 0.75)

As seen in Figure 22, there are no noticeable changes with changing the sigma to the higher value. To analyze the extrema, let's analyze the output image formed with a sigma value of 5.0.

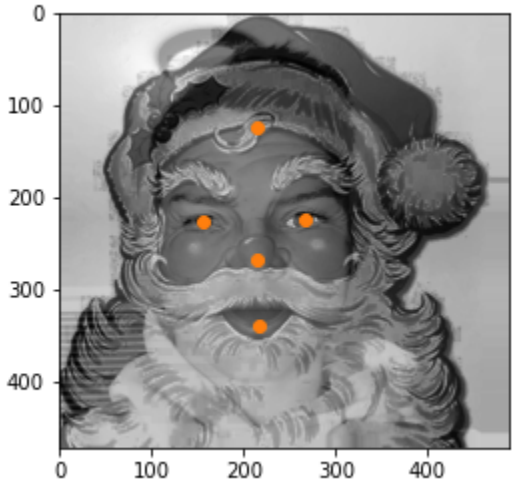

Figure 23: Morphing using t = 0.6 with Gaussian RBF (sigma = 5)

As shown in Figure 23, there is no significant change between sigma = 0.75 and sigma = 5. I conclude that we are saturated and increasing the sigma any further will not result in noticeable changes. Let’s see what happens with a smaller sigma, sigma = 0.01.

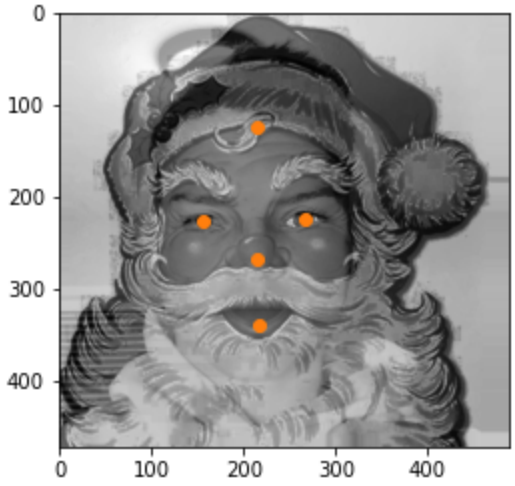

Figure 24: Morphing using t = 0.6 with Gaussian RBF (sigma = 0.01)

Based on the Figures presented above, I conclude that the change of radial basis function does not have a great effect on the output image. This may be because the thin plate spline radial basis function is in fact very similar to that of the Gaussian. We see small changes in the very fine details of smearing intensities etc with varying sigma, but the effect is not noticeable unless one is to look very closely at the detailed pixels. If I had more time on the project, it would be interesting to try another radial basis function that does not take on a similar shape to the thin plate spline radial basis function and the Gaussian RBF.

As one final example, I morph my face with that of spongebob, using t = 0.6 to formulate the output. Since the two faces have similar key points: eyes, nose, etc, the number of correspondences used is only 6. If two images are to be morphed that have more complexity, not just two faces, then it must be noted that more correspondances would be needed to increase the accuracy of the output. After looking at the output, it's clear that since spongebobs face is rotated a little bit and his face is square compared to my round face, the output did not turn out as great as anticipated.

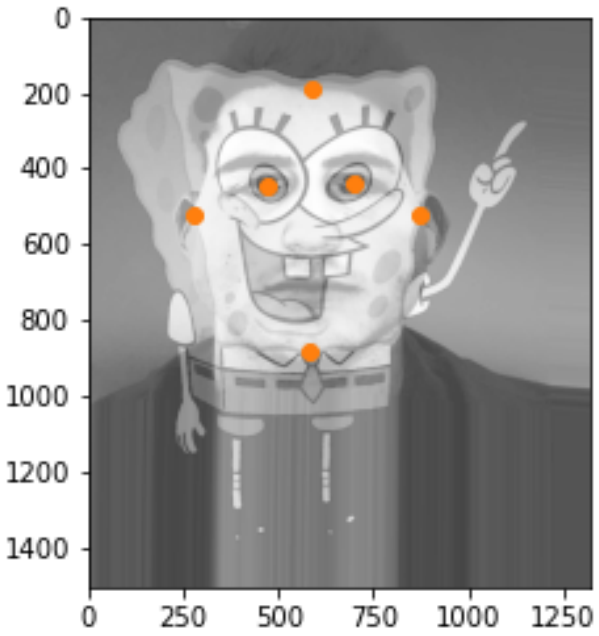

Figure 25: Morphing me and spongebob (t=0.6)

Details - Task1 :

A lot of the details have been outlined in the above discussion but just to reiterate a few key points. Two radial basis functions can be tested by calling the script and inputting the desired RBF. The output image size is dealt with by first doing the forward transform and taking max and min x and y coordinates to find the range of x and y values needed to accurately transform the entire image. Difference in intensities is dealt with using histogram equalization so that the output intensity effect by each input image is set by controlling t in the parametric equation.

Task 2 - Atlases

In the second task of the final project, I utilize the mathematical operations detailed in Task 1 to form atlas templates. In image processing, atlases refer to models formulated from a variety of images representing different instances of the same thing. Atlases are very popular in medical imaging where various samples of the same anatomy can be analyzed by forming atlases and running statistics on the output images to gain a better understanding of the input samples.

The mathematical operation to form the atlas output is very similar to that of the morph except the number of input images is increased. In morphing, we only dealt with two input images but now we deal with a greater number of input images. Coordinates of the canvas are first formed, such as in morphing, using the average x and y coordinates of the inputs. The forward transform of each image to the canvas is then formulated to find the min and max x and y coordinates to set the size of the canvas. The inverse transformations are used to fill the canvas utilizing scipy interpolate to fill in the canvas pixel intensity values (in the same manner as morphing). Instead of the parametric equation used to determine the ratio of the two input images to the output image used in morphing, the average of each pixel location is used to fill the canvas in the atlas. In other words, the canvas coordinates are filled with equal contributions from the input images. To analyze the output image, the mean and variance of each pixel coordinate was found and plotted to deduce information about the input images.

I start off by running the atlas algorithm on the given example, using the same set of shapes as presented in Task 1. To run some statistics on the output, I also calculate the standard deviation amongst the images and output the resultant as a figure, shown in Figure 27.

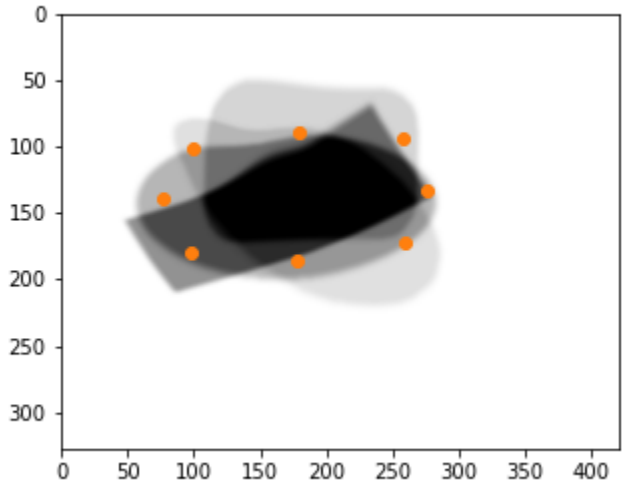

Figure 26: Output of Atlas Algorithm (8 correspondances)

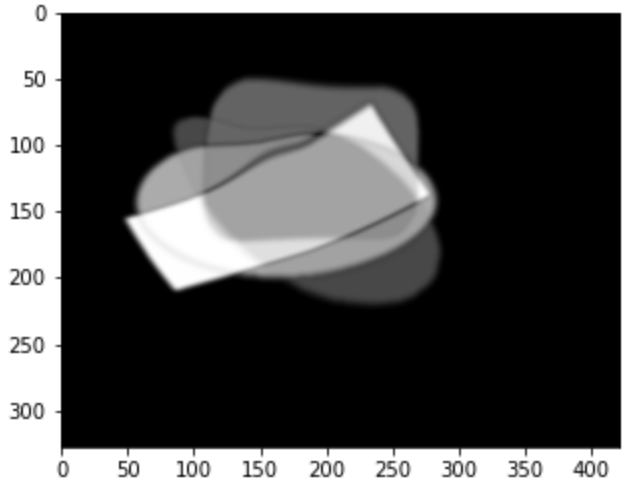

Figure 27: Standard Deviation of image set used in Figure 26

Since the images are fairly far apart with different intensities, we see that the atlasing has a little bit of a hard time trying to form the canvas. For real applications where the images are of the same image, centered in the screen and formatted as such, we would likely see the atlasing algorithm perform better. In Figure 27, we see the standard deviation is lowest by represented black and highest by represented white. It is not very easy to determine how the standard deviation is related to the overlap. On some edges with only one shape, there is no overlap with other shapes and yet the standard deviation is very low. On the other hand, on other edges with a lot of overlap, we see white signaling a high amount of standard deviation. Let's move on to more examples and try to make more sense of the results. To see how well the algorithm can perform on a set of like images, I formulated a data set of different types of shoes (6 images in total) and set correspondences in each of the images. In the first iteration, I set the number of correspondences equal to 6. The raw images are shown below:

Figure 28: First three images in shoes set

Figure 29: Second three images in shoes set

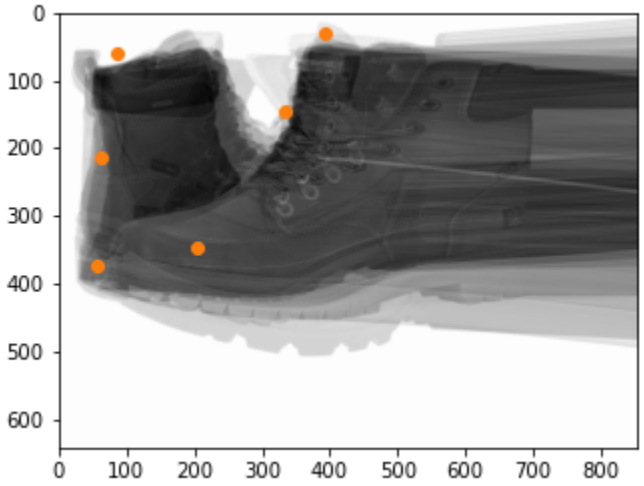

Figure 30: Output Atlas of shoe set

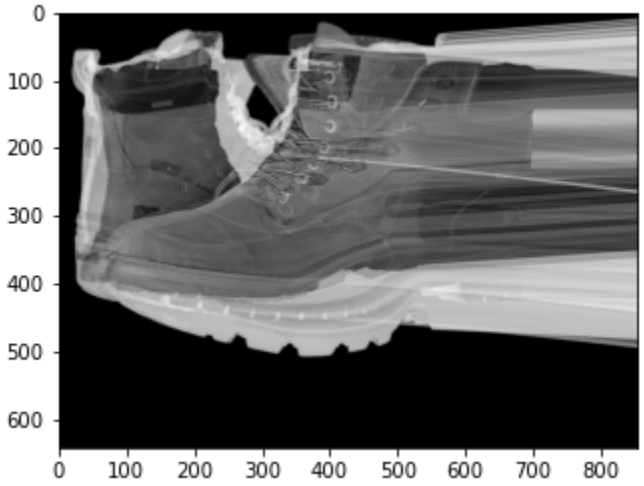

Figure 31: Output Atlas Standard Deviation of shoe set

The Atlas smears the right hand side as shown in Figures 30 and 31. Although this seems like an error in the algorithm, it is actually expected. While uploading the images, one image had part of the right hand side cut off and no correspondences were placed towards the right side of the shoes. As a result, the algorithm does not know exactly how to handle the right hand side and has to make some adjustments for the uniformity in the cutoff image, resulting in major smearing. The rest of the image appears very clean and is a good average of the raw boot images. It must be noted that if I had more time, I would fix these issues and re-run the algorithm, but with such large photos and large number of photos, the algorithm takes too long to run. This example again reiterates the importance of the correspondence placements in forming the output image.

Next, I move on to the major example of the Atlas Task: the brain imaging data. Taking an input of 14 different brain images, a brain atlas was formed and is presented in Figure 32.

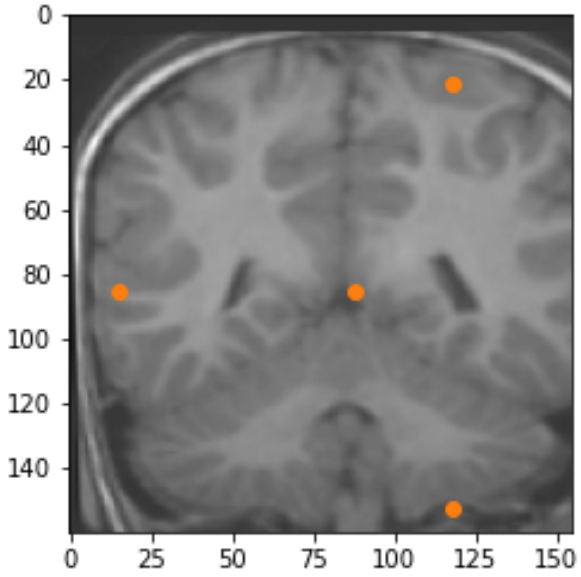

Figure 32: Brain Atlas

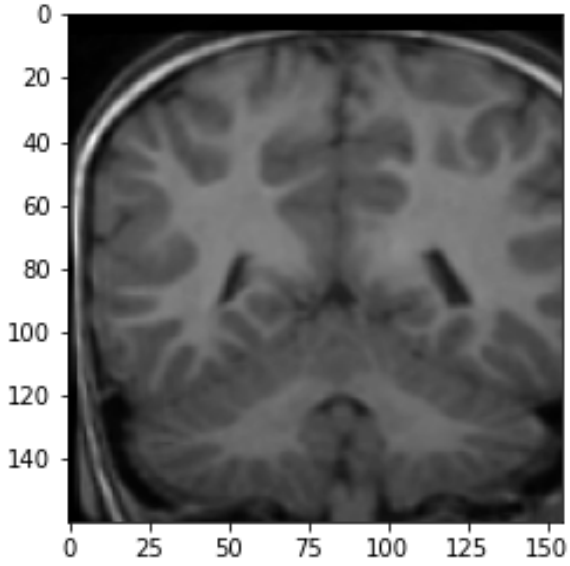

Figure 33: Brain Atlas Standard Deviation

Since the images are fairly hard to make out smaller masses and components, I decided to run flood fill on the image to see if I can make out masses that are otherwise too hard to distinguish with the human eye. This image is presented in Figure 34.

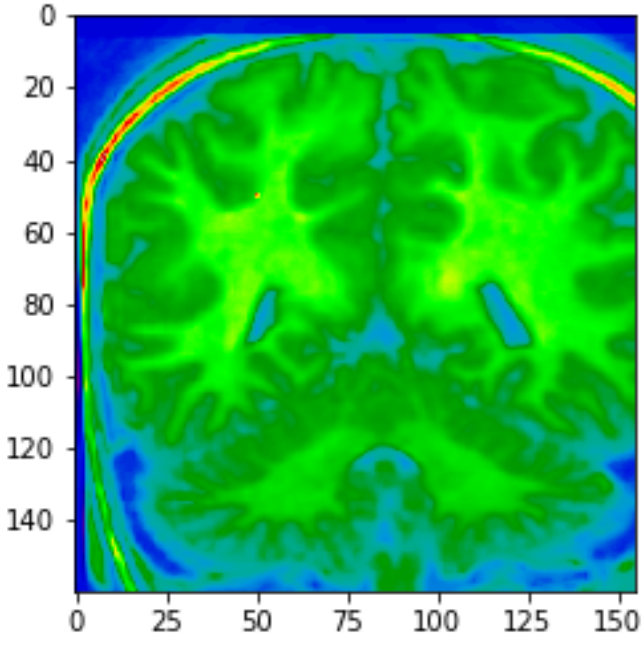

Figure 34: Flood fill on Brain atlas template

As seen in Figure 34, there are a few small yellow masses seen in the right and left part of the brain that signify a non uniform mass distribution. I am not a neurologist and cannot make any conclusions but I find this interesting and it possibly signifies that one of the brain images is not like the rest (as it is seen to have a deviation away from mean) for these mass lumps. Using this knowledge, I conclude that the points that are easily found are those that are not like the rest. For example, if there is a large circular mass in one image, then doing a template and seeing the standard deviation and running flood fill for visual purposes will pull out the area of deviation caused by the single brain image not like the rest (or possibly multiple brain images if the data set is large) and show it in a different color. This type of processing on medical images is very interesting and shows a good introduction on how image processing can be leveraged in medical settings and patient care.

Conclusion

As this is the last project for the semester, I reflect back to the very first project where we were tasked with simple input and output of images. In the final project, we find ourselves using various algorithms on medical images to try and pick out non uniformities that can be interpreted by medical specialists as possible signs of concern.

Credit

I must credit the following people and online websites for making this project possible:

- Dr. Ross Whitaker for teaching me the theory and giving me hints behind the morphing algorithm
- TA Tushar and Jadie for helping me with the details of implementing the algorithms
- **Image Resizer | Crop & Resize Image Online - RedKetchup**
- code for SVD function (not my own code) -> SVD - Matrix transformation Python - Stack Overflow
- Zappos for the pictures of shoes analyzed in the atlas task https://www.zappos.com/men-boots/
- Flood Fill — skimage v0.20.0.dev0 docs (scikit-image.org)