CS 192-26 Final Project: 3D Doggos

Jerry Lai

December 11, 2021

Contents

1 Introduction

There are many examples of interesting photo manipulations of human faces. For example, in Project 3, we implemented numerous different ways to analyze and augment our own faces and those of faces and those of Danish computer scientists. Every social media app has a huge library of filters that are specifically made to alter the face of the user, and many video games give you dozens of sliders to modify every aspect of your character's (usually human) face.

However, there is little in the way of face image manipulation for dogs, despite the prevalence of dog pictures. There are a few possible reasons for this. One is the great variety between dogs of different breeds. The other is the difficulty of getting dogs to take high-quality, well aligned headshots the way humans can.

This project attempts to manipulate the faces of dogs using a 3D approach, and uses the Columbia Dog Dataset [3][1].

We perform the following steps:

1. Use an SFM algorithm to recover a mean 3D shape for each breed of dog.
2. Find the camera angle and scale for each individual dog image.
3. Use the angles, shapes, and dog images in the dataset to selectively reconstruct mean faces for each breed.
4. Demonstrate a workflow that uses these methods to change the breed of a dog.

The rest of this paper is structured according to these steps.

None of the methods described or used in this paper involve deep learning, except a preprocessing step where a pre-trained segmentation model (FCN Resnet 101 from Pytorch) is used to remove the background of the images.

2 SFM

This step uses math that is heavily based on the preprocessing step of both [4] and [2]. It uses an iterative algorithm to minimize:

$$\sum_{n=1}^{N} \left\| W_n - \begin{bmatrix} M_n & T_n \end{bmatrix} \begin{bmatrix} S \\ \mathbf{1}_{1\times K} \end{bmatrix} \right\|_F^2$$

Where W_n is the list of keypoints in the image;

N is the number of images for that bread;

$K = 13$ is the number of keypoints;

And the variables being optimized, $S, M_n, and T_n$, are the mean 3D shape, top 2 rows of rotation matrix, and translation matrix respectively.

After the mean shape S is recovered, it is aligned by finding M_0 and T_0, for just the mean keypoints for all of the images of a breed in the dataset and their flipped counterpart.

This is done with the assumption that, on average, the dogs are looking at the camera.

Next, to demonstrate the the 3D shapes, 2D mean faces for each breed are created then pasted onto an isometric view of each breed's mean 3D shape.

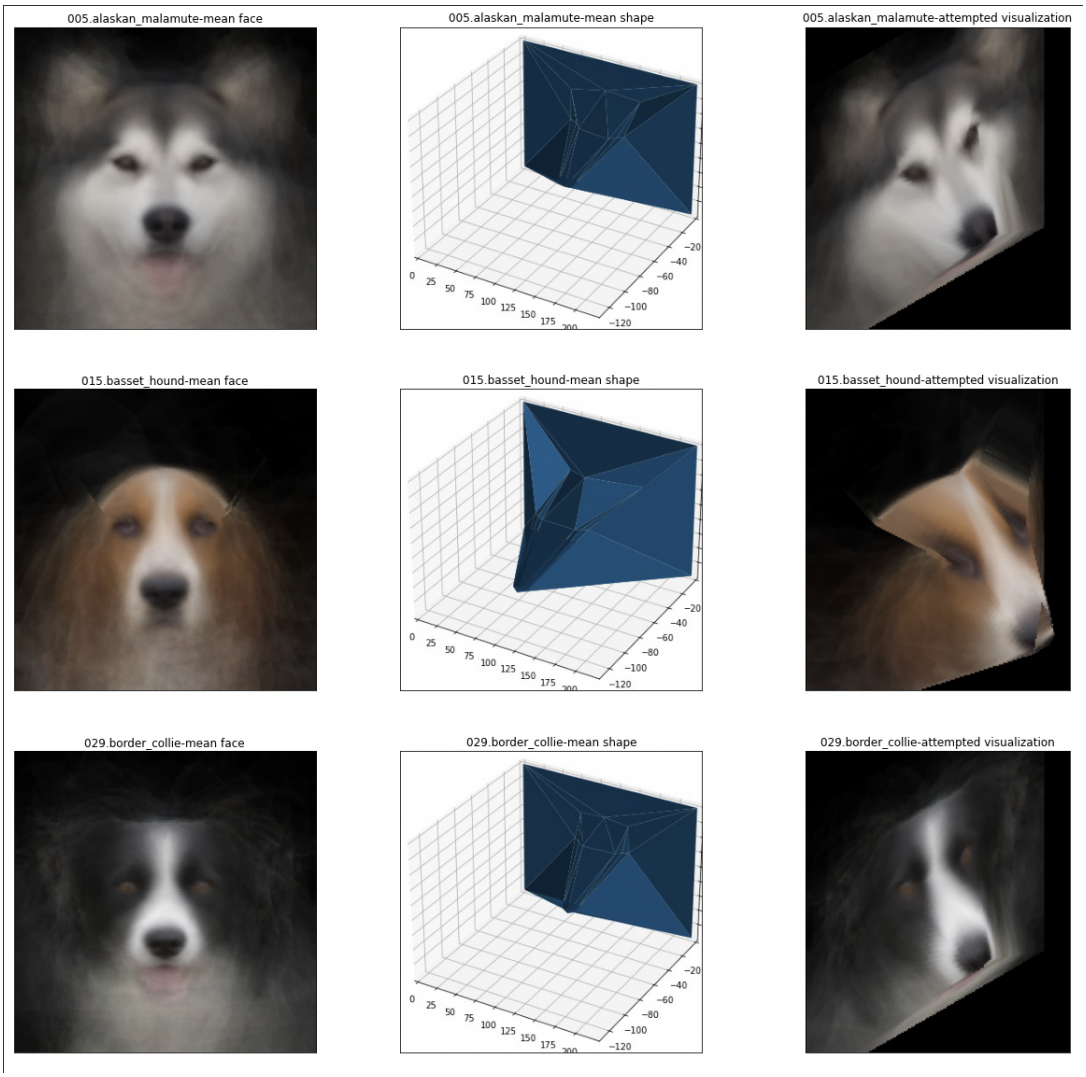

Figure 1: Left: 2D mean face. Middle: 3D mean shape. Right: Face made 3D. More images are on the website submission.

3 Recovering Camera Parameters for each image

Although the SFM method described above also finds camera parameters for each image in the form of motion and translation matrices, simple nonlinear least squares turned out to be more accurate and consistent in recovering the desired variables for this section, which are the 3 rotation angles, 1 scaling factor, and 2 translation distances.

These variables are named as follows:

- α - Tilt head left or right; rotation around z (depth) axis.
- β - Look left or right; rotation around y (vertical) axis
- γ - Look up or down; rotation around x (horizontal) axis
- s - scale
- T_x - horizontal translation
- T_y - vertical translation

For each image, the values of these variables are optimized using the Trust Region Reflective algorithm, which is included in the `scipy.optimize.least_squares` package.

During optimization, each rotation angle is initialized at 0 and bounded by 1 radian in each direction. s is initialized at 1 and bounded between [0.25, 4], but in practice it never exceeded [0.5, 2]. T_x and T_y are initialized at 0 and bounded by the size of the image.

The function to minimize is as follows:

Input: Image keypoints k_i, 3D shape keypoints k_s, $\alpha, \beta, \gamma, s, T_x, T_y$

1. $R \leftarrow$ rotation matrix from α, β, γ
2. $k_r \leftarrow s \cdot R \cdot k_s$
3. $k_r \leftarrow k_r[x, y]$
4. $L \leftarrow k_i - (k_r + [T_x, T_y]^T)$
5. Return L^2

In other words, the squared distance between the image keypoints and the mean breed shape after you transform it and ignore the depth.

4 Constructing selective mean face by angle

The main goal of this step is, given a set of rotation angels, to generate a face that is the weighted average of the k faces from the dataset that are already closest to that angle.

This is easily done with the output of the previous step. With all of that information, all that is required is the following steps:

1. Rotate the mean 3D shape of the breed by the specified angles and flatten it
2. Predefine k weights to apply to each image
3. Take the k images with the closest distance between its rotation and the desired rotation
4. Warp each of these images to the shape from step 1
5. Multiply each image by its weight, and sum.

In the below example, $k = 6$ and the weights are specified to be $[0.21, 0.19, 0.17, 0.15, 0.14, 0.14]$

I made a cool animation with these images. Sadly, I can't show it here, but it is on my website submission.

5 Demonstration: Breed Shift

Hypothetically:

Jerry bought a dog and named it Beag. He's not sure what breed it was, but he assumed it's a beagle (It was actually a Dachshund, but it doesn't matter). He's pleased with the performance of his dog. However, in the back of his mind he wondered if he should have gotten an Alaskan Malamute instead.

He knows what Alaskan Malamutes look like. But Beag is unique and Jerry wanted to see what Beag, specifically, would look like as an Alaskan Malamute.

Jerry takes a photo of Beag and uses Photoshop to remove the background and add keypoints. (He can also use machine learning for this, but he doesn't have to).

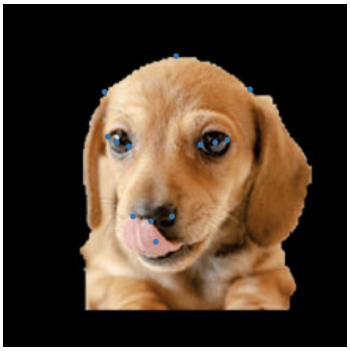

Figure 2: Beag [5]

He uses the optimization method outlined in Section 3 to recover the rotation, scale and translation of his beagle pic. Then, armed with these parameters, he uses the method in Section 4 to construct the mean image (for this specific angle) of both Beagles and Alaskan Malamutes.

Figure 3: Mean Beagle and Malamute, facing slightly down and camera left

He then morphs Beag to the mean Beagle, then subtracts the mean Beagle from Beag. He saves difference between the mean Beagle and Beag, both in terms of pixels (`my_pixel_diff`) and the differences between the keypoints (`my_shape_diff` ← Beag's keypoints - mean beagle keypoints).

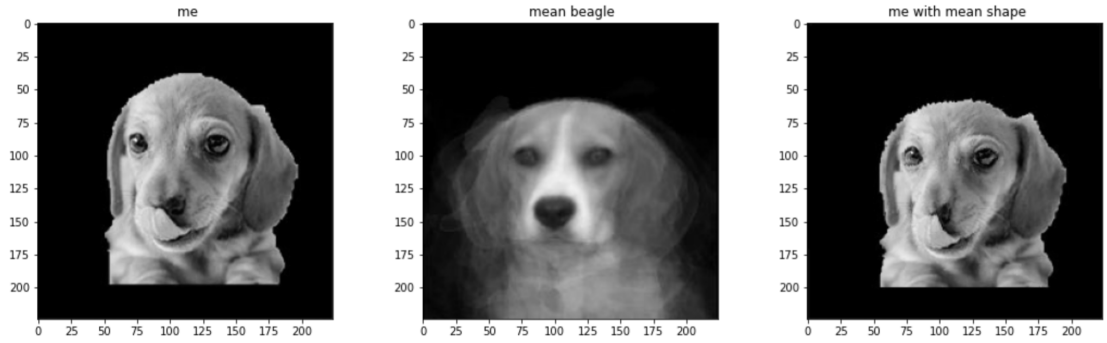

Figure 4: These images are grayscale for more consistent results.

Then, he takes differences between Beag and the average Beagle (`my_pixel_diff`) and morphs it to the shape of the average Malamute.

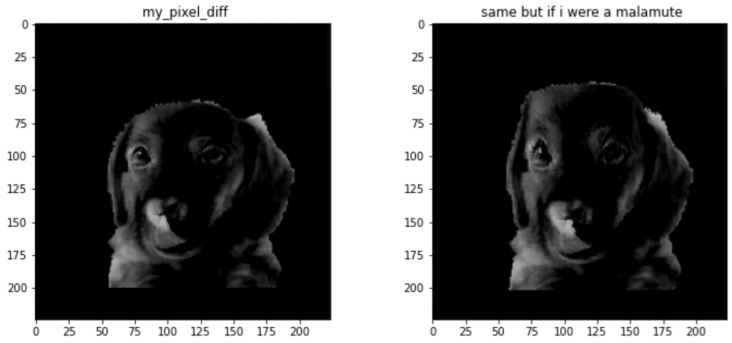

Figure 5: The uniqueness of Beag's colors, eyes, and tongue

Finally, he ads the morphed version of `my_pixel_diff`) to the average malamute before morphing the result by an additional `my_shape_diff` to show the uniqueness of Beag's shape.

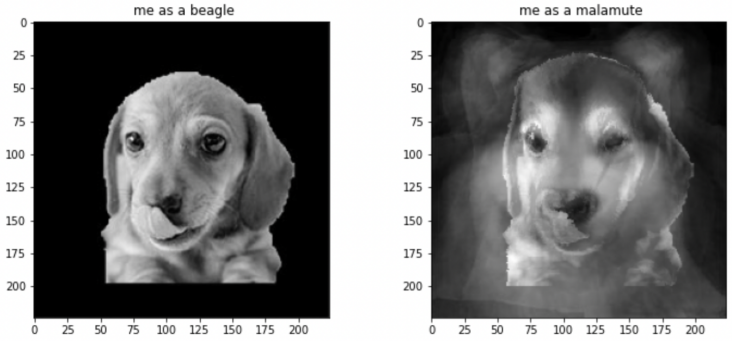

Figure 6: Before and after.

Jerry is very convinced that Beag looks better as is.

References

[1] Columbia dogs with parts dataset. `https://people.eecs.berkeley.edu/~kanazawa/`.

[2] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections, 2018.

[3] Jiongxin Liu, Angjoo Kanazawa, David Jacobs, and Peter Belhumeur. Dog breed classification using part localization. In *European conference on computer vision*, pages 172–185. Springer, 2012.

[4] Manuel Marques and Joao Costeira. Estimating 3d shape from degenerate sequences with missing data. *Computer Vision and Image Understanding*, 113:261–272, 02 2009.

[5] Steffi Trott. Some dog. https://spiritdogtraining.com/dachshund-cost/, Jul 2021.