Computer Vision Coursework

Candidate number 167083

June 5, 2020

1 Introduction

In this paper I present a simple solution to locating facial landmarks in images, the task being referred to as *face alignment* in the field of Computer Vision.

I will start by discussing some potential approaches to designing a system for face alignment, motivate the choice of using linear regression, talk about how I went about implementing it and, finally, discuss its performance.

At the end of the paper I will present simple solutions for the skin segmentation problem and the application of graphical effects on faces in images.

2 Face alignment

2.1 Design

There are many approaches to solving face alignment: Linear regression, cascaded regression[3][5][2][4], neural networks, etc. Being my first attempt at solving a vision problem I started with the simplest approach: linear regression.

The pipeline has three parts: **1.** Loading, augmenting, pre-processing, and splitting images into a training and a test set; **2.** Extracting features from images; and **3.** Fitting a linear regression model using said features as independent variables and the coordinates of face landmarks as dependent variables.

2.2 Architecture pipeline

2.2.1 Data preparation

The data 2811 training images (with landmarks), 554 test and 6 examples.

Data augmentation (Mirroring) Doubles the available training data.

Pre-processing

Vignette To reduce the weight of the information at the edges of images (hair, background objects, other faces, etc.) (Figure 1).

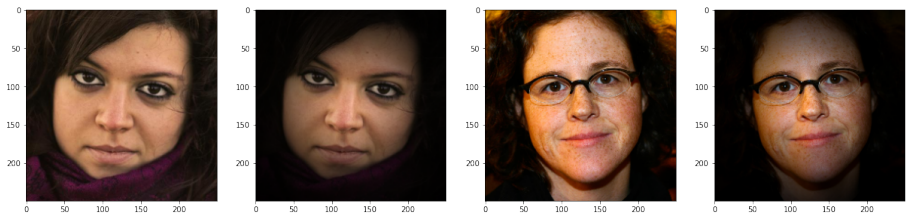

Figure 1: Vignettes using Gaussian kernels with $\sigma = 70$

Cropping First, I explored the given data set to see where most landmarks are. Then, I thought that some useful information might sit immediately outside of the outer landmarks that might be useful in deciding where the face is (like shadows). Therefore, I cropped the images to remove redundant information (Figure 2).

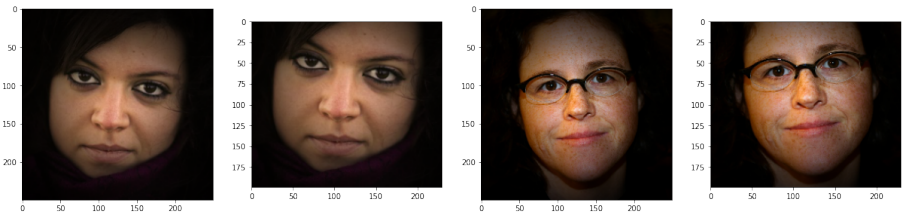

Figure 2: Cropping 40 off the top, 10 off bottom and each of the sides

Rescaling Image features useful for face alignment (e.g. shadows or corners) are still human-discernible in downscaled images. This would reduce training time (Figure 3).

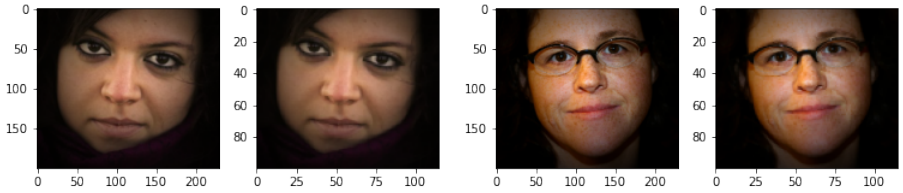

Figure 3: Rescaling by 50%

Gaussian Blur To further reduce the amount of information carried by images, I applied a Gaussian Blur with kernel size $size_{kernel} = (5, 5)$ and $\sigma = 1$.

Training/Test split The training data was randomly split into 80% training and 20% test data sets. For the reader to reproduce the results presented in the remainder of the paper, this was done with the *sklearn.model_selection* method, with *random_seed* set to 111.

2.2.2 Feature extraction

Training a face alignment model using multivariate linear regression requires the same number of features (independent variables) to be extracted from each image. Feature descriptor methods like SIFT and SURF return varying numbers of keypoints between images. HOG, on the other hand, returns the same number of descriptors for each image (given by HOG's parameters like number of orientations and grid size and the image size).

HOG descriptors Histogram of Oriented Gradients feature descriptors are usually used in object detection, but here I tried to use it for face alignment. This technique extracts shape and appearance information from pre-set sub-windows of the image in the form of a collection of histograms of intensity gradients or edge directions.

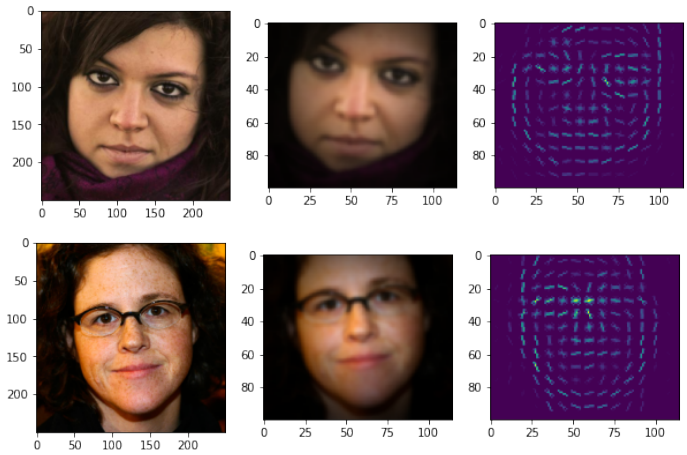

Figure 4: Example HOG feature descriptors (8 orientations, 8x8 pixels per grid cell) from pre-processed images

2.2.3 Linear regression

Multivariate linear regression takes a set of observations (or independent variables) X, in this case the **flattened vector of 68 coordinates of face landmarks for each image in the training set**. The model also needs a set of dependent variables Y, in this case the **HOG descriptors** (visualised in Figure 4), that the model will learn to predict (linearly fit) given the training landmarks X.

To test the accuracy of the model, I extracted the HOG features from the test set images (split in the previous step) and predicted their landmarks. I reshaped the (flat) predictions back into 68 pairs of landmark coordinates.

Thus, the trained model can predict the landmarks of any image given its HOG descriptors, as long as the image has the same size as the training images.

2.3 Model iterations

2.3.1 Average face model

The simplest model is averaging all the training landmarks into one archetype. This is not very accurate however, given that the current data set has most faces in similar locations and of similar scales, the model *seems* to work.

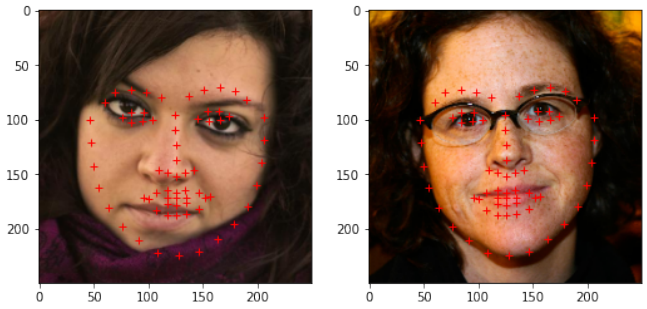

Figure 5: ***Average face model*** predictions

2.3.2 Model A

Data augmentation (mirroring), **pre-processing** (vignette, crop, downscale, Gaussian blur), **HOG** (8 orientations, 8x8 pixels per cell).

Initially I thought that by augmenting (mirroring all training data) and pre-processing the data the model would yield great results, as it would have more data to learn from and would generalise away from unnecessary information (like freckles, stray hairs, etc). However, these led to a very poor performance, especially by predicting stretched or squished face landmarks (see Figure 6).

The model seems to have learned something about rotation and where certain features are (e.g. eyes, eyebrows). I noticed the major irregularities to be on the x-axis and, therefore, assumed mirroring to have been the cause. I investigated this hypothesis next.

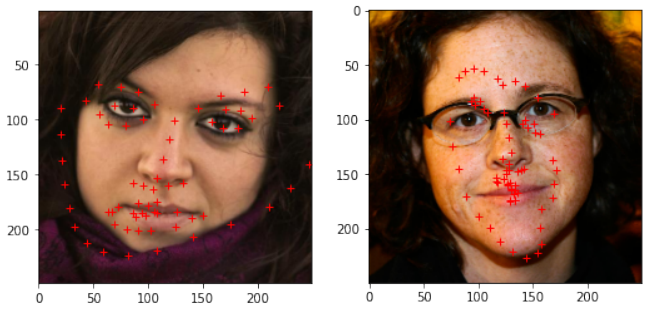

Figure 6: ***Model A*** predictions

2.3.3 Model B

Pre-processing (vignette, crop, downscale, Gaussian blur), **HOG** (8 orientations, 8x8 pixels per cell).

As a result of removing augmentation the model gave better results, learning to predict the rotation and outer face landmarks somewhat well.

Model B's weaknesses are picking up fine details (e.g. around the lips) and being more precise about the outer face limit. At this point I thought that I may have gotten rid of too much information such that the HOG descriptors could not pick up fine details that are, in fact, needed for a good prediction.

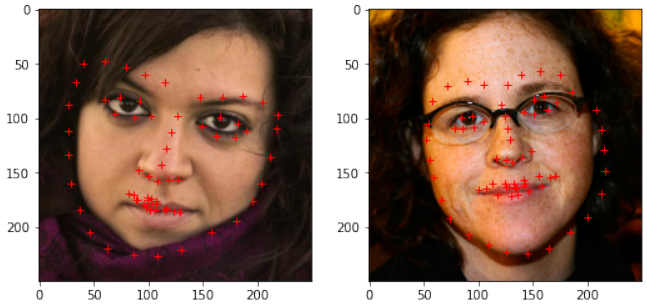

Figure 7: ***Model B*** predictions

2.3.4 Model C

No data augmentation, **pre-processing** (vignette, crop, downscale, Gaussian blur), **HOG** (8 orientations, 4x4 pixels per cell).

Without reducing any of the pre-processing, I just made the HOG descriptors pick up finer information by making the grid cells smaller, from 8x8 to 4x4. Indeed, this taught the model to better predict details around the lips and face edges. However, I was still not satisfied with the accuracy.

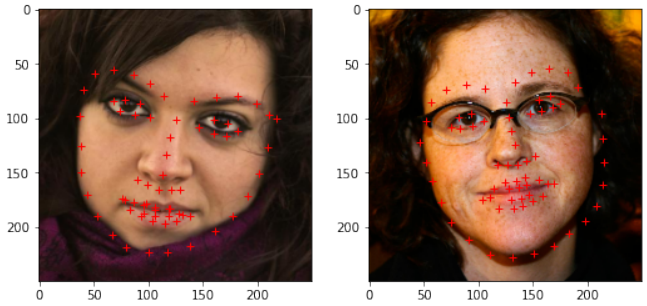

Figure 8: ***Model C*** predictions

2.3.5 Model D

No pre-processing, **no pre-processing HOG** (8 orientations, 8x8 pixels/cell).

At this point, after making the HOG descriptors account for finer details, I was curious whether my pre-processing was detrimental to the model learning

useful information. I decided to test this hypothesis by fitting the model using HOG descriptors taken from unprocessed images.

I found the results of this model to be satisfactory. Looking at the plotted predictions I felt like they are not too far off from where I would place them. This accuracy would probably work well for simple applications (e.g. simple face filters), while it may not work that well for applications that need very accurate predictions like make-up try-on (e.g. in Figure 9, image on the right, the left eyebrow is predicted to be above where it actually is, potentially resulting in very poor make-up application).

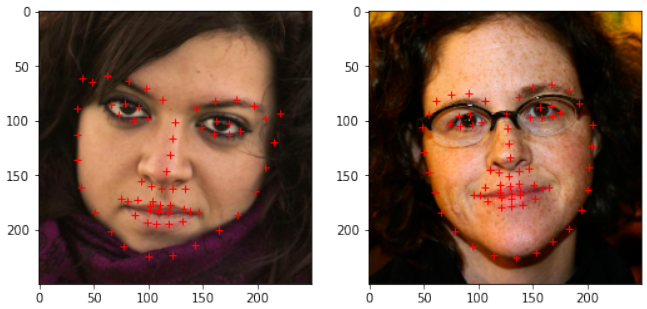

Figure 9: ***Model D*** predictions

2.4 Model analysis

The accuracy of a model is calculated as the average Euclidean distance from the predicted landmarks to the true landmarks of the test set (split before training).

The cumulative density measure shows a steady increase in accuracy performance over the model iterations (Figure 10). ***Model A***, lower in performance than ***Model B*** but higher than the ***Average face model*** was left out for legibility (its performance being calculated on double the test data due to augmentation).

2.5 Discussion

2.5.1 Pre-processing

One of the main lessons for me was that I should first try and extract features from the raw data, without any pre-processing, instead of blindly pre-processing images based on a priori assumptions. In this case, pre-processing decreased the average model accuracy.

2.5.2 Training time bottleneck

As seen in Figure 11, the higher the data dimension (image size, scale), the longer it takes to compute the HOG descriptors. Computing HOG descriptors

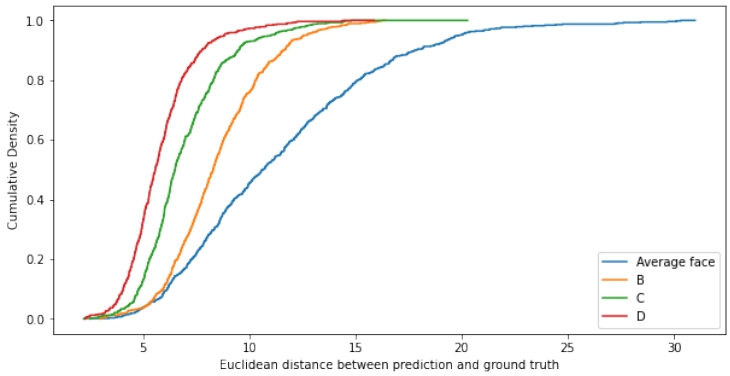

Figure 10: Cumulative density measures of discussed models (except ***Model A***)

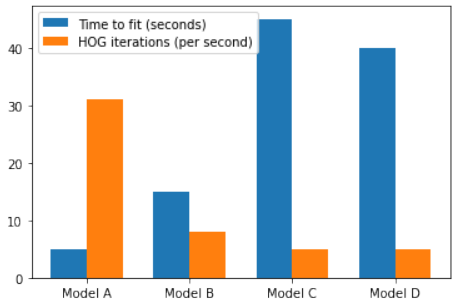

Figure 11: Training time performance comparison

is the clear time bottleneck of training and testing. For some applications (high frames-per-second video face alignment), fast descriptor computation is key, so a trade-off with accuracy might be needed.

2.5.3 Potential improvements

I am very surprised by ***Model D***'s accuracy, given its very simple architecture. However, there are many areas of improvement (e.g. details around the eyebrows, eyes or lips). Learning high-level information (roll, yaw, pitch) and low-level information (smile, open mouth) all at the same time might be too demanding from one linear regression model.

One possibility of improvement is having a cascade of linear regression models, with early models learning high-level information and later models dealing with fine adjustments.

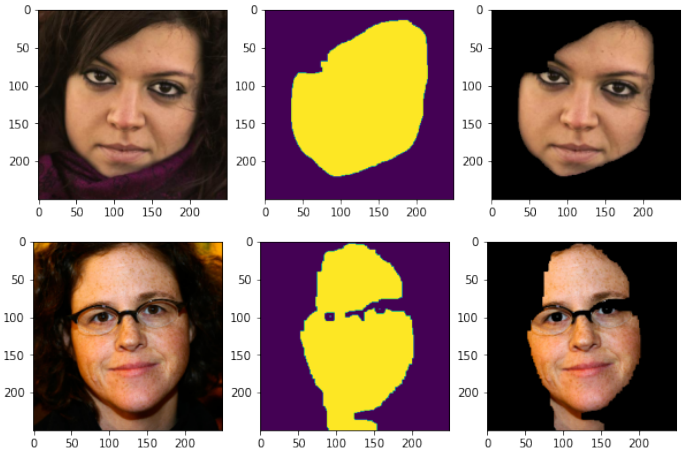

Figure 12: Pixel segmentation using GrabCut (original image, 5th iteration, face pixels extracted

3 Pixel segmentation

A potential improvement for the GrabCut method to include all pixels within the boundary defined by the predicted landmarks.

4 Face effects

Inspired by Canu [1], I have created a Clown Nose face filter. The red nose dynamically scales according to the scale of the face (landmarks). Potential improvements include brightness and contrast matching (to the source image) and 3D perspective alterations using the face landmarks.

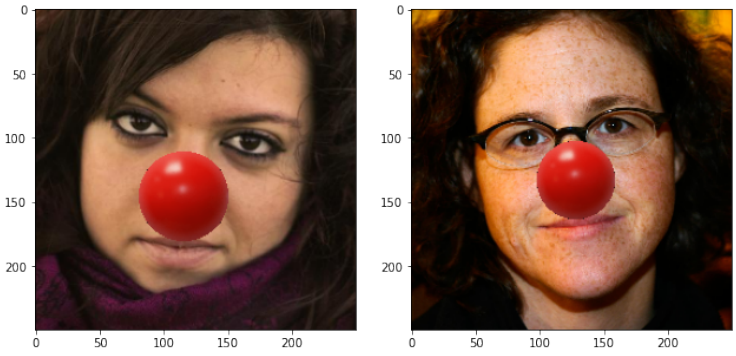

Figure 13: Clown Nose filter

References

[1] Sergio Canu. Pig's nose (instagram face filter) - opencv with python, Jun 2019.

[2] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2887–2894, 2012.

[3] P. Dollár, P. Welinder, and P. Perona. Cascaded pose regression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1078–1085, 2010.

[4] Lin Feng, Caifeng Liu, Shenglan Liu, and Huibing Wang. A fast online cascaded regression algorithm for face alignment. *CoRR*, abs/1905.04010, 2019.

[5] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1867–1874, 2014.