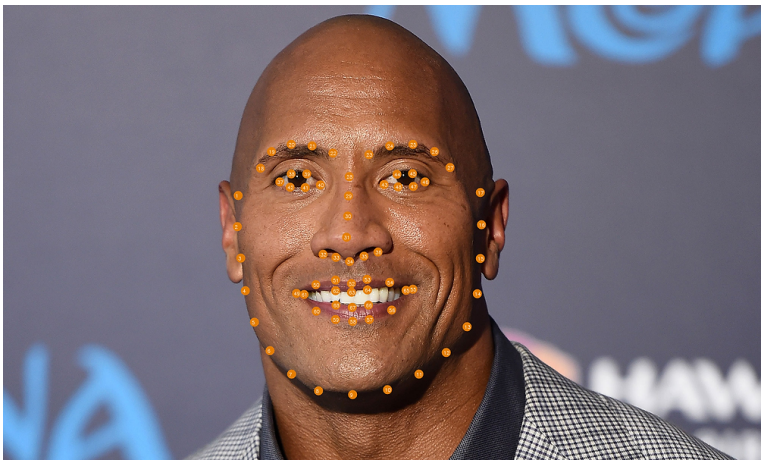
University of Sussex



Year 2 Computer Science w/ Artificial Intelligence - Computer Vision

Face Alignment system

May 12, 2022

Report

234591

Contents

Abstract

Facial Alignment systems have many applications ranging from simple face filters like those on Snapchat[2] to facial recognition when issuing identity or preventing fraud[3]. In the world, there are various methods to create facial alignment systems like one created in the dlib and mediapipe libraries. The libraries provide functionality for creating a simple face alignment system however, in this task we were specifically asked not to use any libraries that directly solve the task at hand.Through analysis of various computer vision techniques, we found it reasonable to assume that we can use haar cascades in a viola jones detector to be able to identify specific facial features like the eyes, nose and mouth to fit a template to each of the features. After realizing that this challenge would be excessive, we stuck with the idea of a template and used different edge detectors to get an outline for the face with varying degrees of success. Lastly, we realized that with the use of the points given we could pass these into a linear regression model and let the model predict the rest. This came out successful and on wards from that we began to alter what data was passed into the model. In the end we found that using hog features on an image as pre-processing and passing that into the linear regression produced the best results

1 Introduction

Face Alignment is a computer vision technology that identifies the geometric structure of a face through the training of numerous digital images of faces. Given the position of a face, a facial alignment system will determine the shape of face components seen within the image. The facial alignment accuracy greatly depends on the implementation of the system and on the number and nature of training images used to train

the system. In this report, I will describe the process of creating my own facial alignment system including all the challenges that came along with it

With this context, the following objectives below were given:

1.This assignment will involve you designing, building, testing and critiquing a system for per- forming face alignment, aka. locating facial landmarks in images.
2. You will design and implement a system for modifying the colour of the lips and/or the eyes in the image. This could be achieved through segmentation, purely using the estimated landmarks. or more advanced means (e.g. generative adversarial networks)

---

2 Methods

2.1 Haar Feature Detector Method

When first beginning the process of making a facial alignment system, we believed we had to follow 3 steps in order to plot the facial landmarks in the correct position; Locating the correct position of a face in the image to draw a template, pre-processing the data and passing this data into the model being used. Initially, I attempted to locate the position of the face using a Viola-Jones detector which outlines a box where it detects haar-like features.[1]

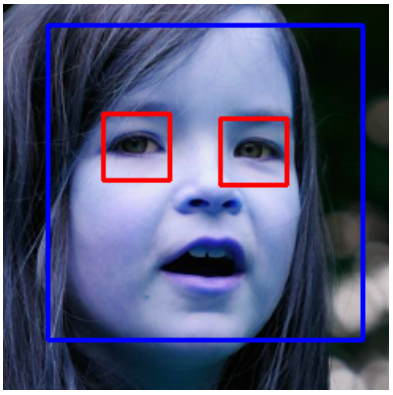


Figure 1: the facial box of a Viola-Jones detector using haar cascades

However, through more intensive research we discovered that the best method to create a facial alignment system would be to train a set of points using a classification model.

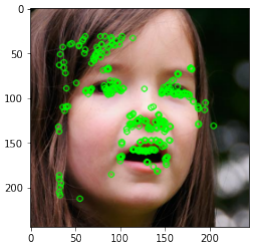
2.2 Pre-processing

For my facial alignment system, we had to process the image data and the points data. Pre-processing helps the model recognise correct keypoints in a picture

2.2.1 Greyed flatten Image (Regular Linear Regression)

It is necessary to flatten the images for a regular Linear Regression because linear regression works with vectors so the original 3d shape of the images will not fit into the linear regression. It is useful to know that for the size of the traintestsplit we chose to use and 80 - 20 split as we wanted a model that represented the training set well.

2.2.2 Oriented Oriented FAST and Rotated BRIEF (ORB)



Using ORB to preprocess an image returns an image with keypoints dotted over the image.To pre-process photos using ORB features, we created a function called orbimages() that loops through a set of images and draws the key points onto the image. we then proceeded to pass the gray images from before into traintestsplit() so that when we am fitting the model we can run orbimages onto the training images and testing images. This allows the Linear regression model to compare these features when drawing the facial landmarks. we chose orb as one of the pre-processes because ORB is proven to have much greater efficiency when it comes to unprocessed matching potential and reliability of image matching applications than any other feature descriptor.[4]

2.2.3 Canny Edge Detection

The pre-processing of a image using a canny edge detctor returns an edge map of a face. As part of the pre-processing for the Canny linear regression model, we had to turn every image into Canny image and then flatten them to make sure that they can fit in the regression model. we decided to incorporate Canny edge detection because It is generally immune to a noisy environment and some of the images given contain lots of noise that other edge detectors might struggle with.[6]

2.2.4 Histogram of Oriented Gradients (HOG)

The pre-processing of HOG returns an image that has the gradient orientations of each edge in the picture giving a good outline in the image. Due to the fact that HOG splits images into cells of 16x16, HOG returns the same features for each image which is ideal when fitting to a linear regression. TO pre-process an image using HOG I created a function, called hogimages(),that turns a set of images to HOG descriptors so that i

can fit the model with training images that have been ran through hogimages(). we then predicted the points on a image that has been ran though hogimages()[5]

2.3 Linear Regression

Linear regression uses the value of the independent value to predict the value of the dependent variable.we decided on using a linear regression model due to the ease at which one can implement, interpret and train it and because we can use the pre-processed data we already had to make the model perform better. To implement our system we first created the Linear Regression model and then fit the processed points and images into the linear regression model.

2.4 Lip Modification

We threw around a range of ideas to do with lip modification. Initially, we thought that using a haarcascade to get the area of the mouth and then using cv.polyfill to change the colour of the points inside this area. However, we realised that the points of 22-41 were always the points drawn for lips so we put this range of points into a new array and then ran cv.polyfill, passing these points into cv.poly fill which changed the colour of the lips

3 Results

To calculate the results, we created a euclidean distance function that measured the distance from the ground truth points to the predicted points we drew with our model. We then totaled the euclidean distance of all the points in all the images and then averaged them. We also collected the min value and the max value of euclidean distance of each pre-process

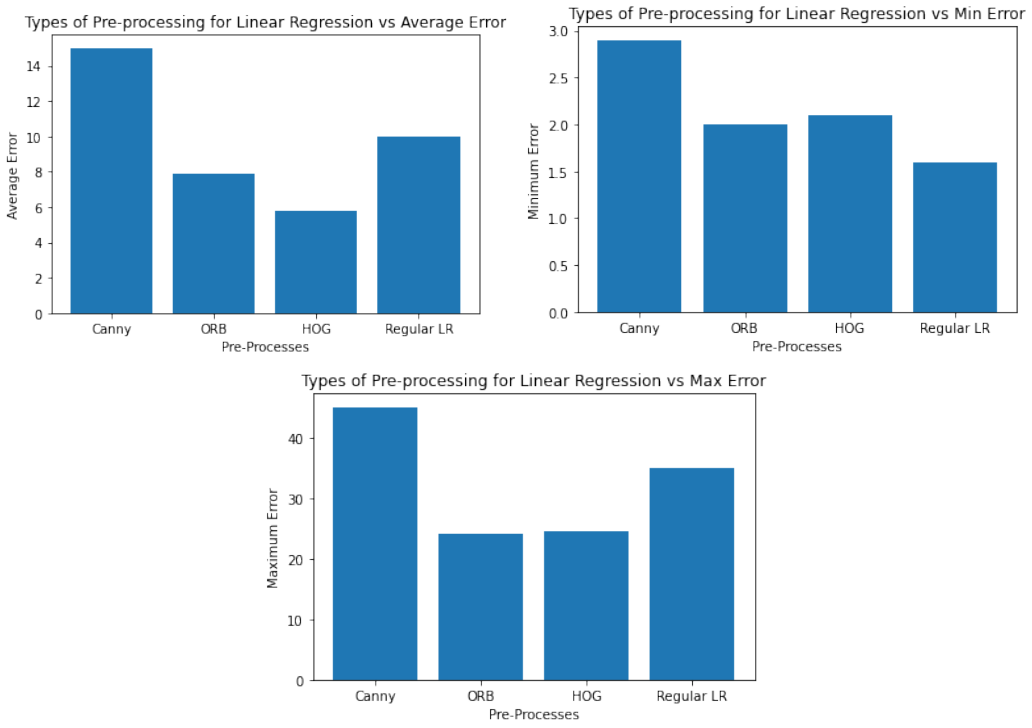


Figure 2: The average, minimum and maximum error for each pre-process

3.1 Regular Linear Regression

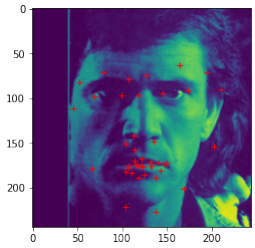


Figure 3: face alignment with Regular Linear Regression

Regular LR scored 3rd:
Average error: 10.0
Minimum error: 1.6
Maximum error: 35.0

3.2 Canny Linear Regression

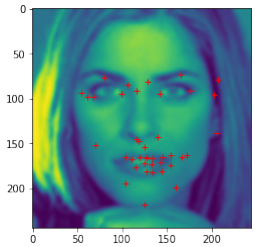


Figure 4: face alignment with Canny Linear Regression

Canny LR scored the worst:
Average error: 15.0
Minimum error: 2.9
Maximum error: 45.1

3.3 ORB Linear Regression

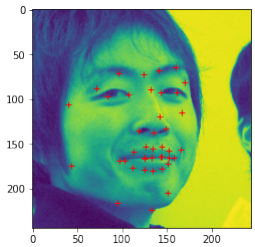


Figure 5: face alignment with ORB Linear Regression

ORB scored 2nd:
Average error: 7.9
Minimum error: 2.0
Maximum error: 24.2

3.4 HOG Linear Regression

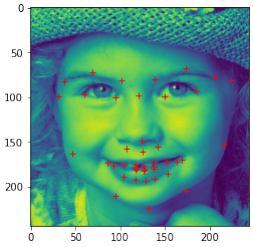


Figure 6: face alignment with HOG Linear Regression

HOG LR scored the best:
Average error: 5.8
Minimum error: 2.1
Maximum error: 24.5

3.5 Lip Modification

Figure 7: Results of the Lip Modification

The Lip modification came out successful and it works exceptionally well even when the mouth is open due to the accuracy of the points used to fill the polygon

4 Discussion

It was interesting to see that the Canny edge detection was the worst out of all of them. This may be due to the fact that canny Gaussian smoothing may cause the location of edges to be off depending on the size of the kernel or because the canny edge detector did not run perfectly on all images:

Figure 8: failed canny image

As you can see the outline of the face is not entirely there so, the model may struggle to draw all facial landmarks which led to a decreased accuracy and an increased error[7].
In addition to this, it was interesting to see that the regular linear regression also had the lowest value for the minimum error indicating that the regular linear regression land marked an image the most accurately. On the other hand, it was no surprise that HOG came out on top due to the fact that HOG is the best feature descriptor because it provides orientation in the information and not just the edge.

5 Conclusions

Facial alignment accuracy greatly depends on the system used to pre-process the data and the model in which you use to classify it. In our model, we decided to use Linear Regression due to it's ease of use when implementing and training. We found out that pre-processing the image data with Hog features provided the best average result however, it is something to say that the most accurate image was scored by the regular Linear Regression. This implies that certain pre-processes are useful for different types of images.

References

[1] *Cascade classifier*. URL: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html.

[2] James Le. *Snapchat's filters: How computer vision recognizes your face*. Feb. 2018. URL: https://medium.com/swlh/snapchats-filters-how-computer-vision-recognizes-your-face-9ce536206fa7.

[3] Mani Mishra. "Study of Nonlinear Control Techniques". In: *SSRN Electronic Journal* (2021). DOI: 10.2139/ssrn.3833784.

[4] *Orb (oriented fast and rotated brief)*. URL: https://docs.opencv.org/3.4/d1/d89/tutorial_py_orb.html.

[5] Mrinal Tyagi. *Hog(histogram of oriented gradients)*. July 2021. URL: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f.

[6] Tee Yee Yang. *Canny edge detection in 5 minutes*. Dec. 2020. URL: https://towardsdatascience.com/canny-edge-detection-in-5-minutes-27236af71ce2.

[7] Dan Dan Zhang and Shuang Zhao. "An improved edge detection algorithm based on canny operator". In: *Applied Mechanics and Materials* 347-350 (2013), pp. 3541–3545. DOI: 10.4028/www.scientific.net/amm.347-350.3541.