Computer Vision Assignment

*Word Count = 1564*

# Face Alignment

Face alignment is a process of locating facial landmarks, such as the eyes, nose and mouth, by identifying the geometric structure of faces in digital pictures ([1] Papers with Code - Face Alignment, 2020). The aim is to get the estimated shape S of the face as close to the true shape as possible, minimizing:.

## Methodology

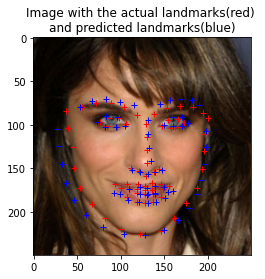
*Flowchart1 shows* the face alignment model plan. The evaluations will be completed using:

1. Euclidean distance =
2. Mean absolute error (
3. Mean square error (
   1. Root mean squared error (RMSE) =



***Flowchart1: The process of face alignment***

### Original Model



***Image1: Original model predicting the points of a testing image.***

The data is flattened to ensure that the data is a 1-dimensional linear vectors, so the linear model can be fitted correctly. 2-fold cross-validation is used to shuffle and split the data into training and testing data. From Skicit-learn, I used the linear regressor which uses the known data to represent a linear model to predict the dependent variables. The regressor will be fitted by the training data using the built-in method ‘fit()’ and the testing data will be used to predict the landmark points using ‘predict()’.

**Original model measurements (to 3dp) using all samples:**

Euclidean distance mean: 8.699

MAE: 5.520

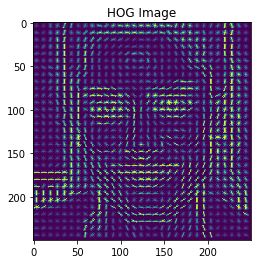
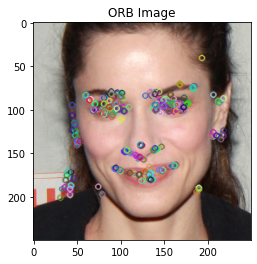
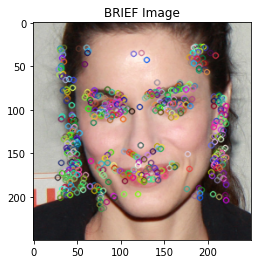
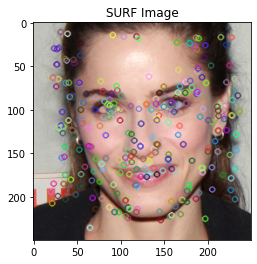
MSE: 58.718

RMSE: 7.663

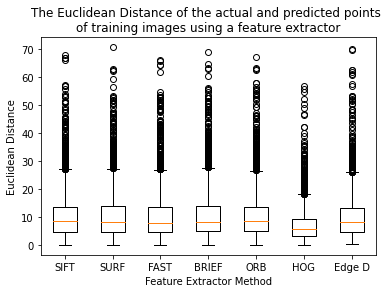
### Feature Extraction

From previous studies, extracting features before training the model is essential. This technique will help predict the landmarks by indicating the interest points of the image. I will be testing and comparing different feature detectors shown below in *image2*:

***Image2: The feature extracted/detected techniques used on a training image.***



***Figure1: The Euclidean distance of test images using the feature extractors/detectors***



HOG is the most effective feature extractor, seen in *figure1,* with a MAE of 4.127(3dp) and a mean Euclidean distance of 6.636(3dp), compared to the other methods producing mean Euclidean distances around 9.5. In *image2*, its observed that ORB, FAST, SIFT and BRIEF doesn’t detect all needed features, e.g. jaw line, whereas, SURF detects randomly scattered features, whilst the lines from edge detection are singular and very defined, all these reasons have led to a less accurate model. Moreover, HOG detects close to all the features that we want to identify and also describes the correlation between the landmarks e.g. the jaw shape of a jaw.

**Measurements (to 3dp) after feature extraction, (200 samples):**

Euclidean distance mean: 6.636

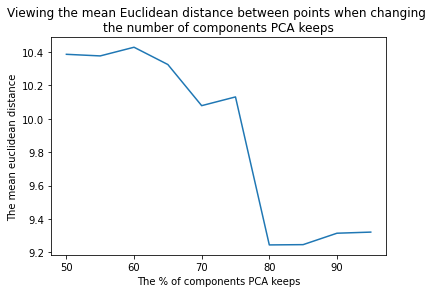
MAE: 4.217

MSE: 35.832

RMSE: 5.986

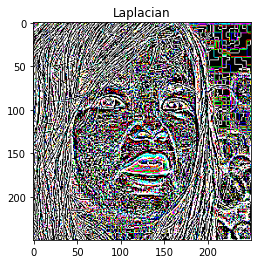
## Preprocessing

Preprocessing is used to prepare data, to ensure that it is easier for the model to analyze and process before regression; leading to a more accurate model. PCA (Principle Component Analysis) is a very common technique, it is predominantly used for dimensionality reduction and to filter noisy datasets. The data needs to be scaled before PCA is performed. Standardization is better than normalization when the model has to make assumptions about the data, however PCA is said to work better with normalized data, therefore, the data will be normalized before PCA. I also experimented with the percentage of components that PCA keeps seen in *figure2,* from these results I decided to have the number of components at 85%.



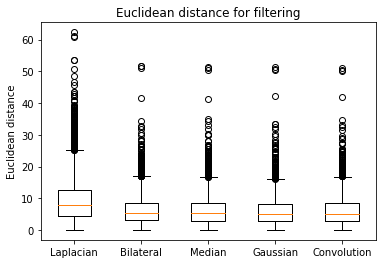
***Figure2: The mean Euclidean distance when changing the percentage of components PCA keeps***

The next preprocessing technique is filtering, this smooths images and reduces noise. *Image3* shows the 5 techniques:



***Image3: The filter techniques used on a training image. Labelled at the top with the filter used.***

After implementation, all filters, besides Laplacian, reduces the mean Euclidean distance to around 5.2-5.3 (*figure3)* and the MAE to around 3.3, the most accurate being the Gaussian filter, therefore this is the filter that will be used.



***Figure3: The Euclidean distance of test images using filters***

Finally, images are converted to grayscale and are resized to a lower resolution, these images will still be sufficient enough for the face alignment task and will prevents the model from being overcomplicated.

**Measurements (to 3dp) after preprocessing (200 samples):**

Euclidean distance mean: 5.013

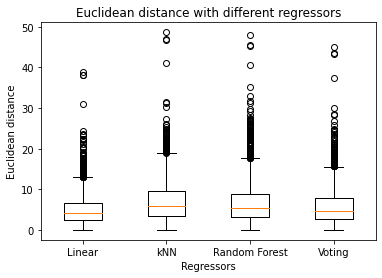
MAE: 3.198

MSE: 19.584

RMSE: 4.425

### Other Regression Methods

K-Nearest Neighbors(kNN) is a regressor that predicts the target points by the association of the nearest neighbors from the training set. Random forest regressor trains decision tress and uses them to obtain the predictions, both have been implemented and compared.



***Figure4: The Euclidean distance of test images using the regressors***

*Figure4* shows that linear regression is the most accurate regressor with a mean Euclidean distance of 4.416(3dp), compared to the kNN with 6.689(3dp) and random forest with 6.575(3dp). A large part of the model’s performance is affected by how the data is preprocessed, during implementation, I compared techniques just with the linear regressor therefore, this may have impacted these results.

Finally, I implemented a voted system that takes the predicted points from each regressor and takes the mean of the predicted points, however the linear regressor was still the most accurate regressor.

## Final Model Evaluation

*Flowchart2* shows the final process of the face alignment model that aims to predict the 68 coordinates of face landmarks. **Final model measurements (to 3dp) using all samples:**

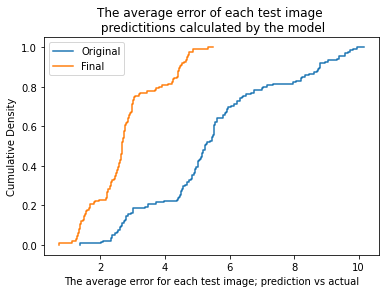
Euclidean distance mean: 4.416

MAE: 2.804

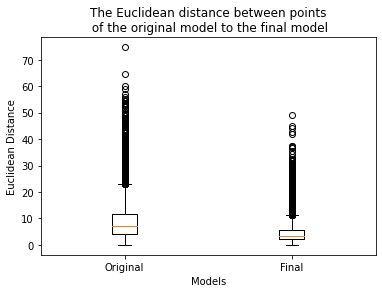
MSE: 15.684

RMSE: 3.960

The final model has nearly halved the mean Euclidean distance, seen in *figure5,* as well as halving the MAE and RMSE, compared to the original model. *Figure6*, shows the cumulative density of the point’s average error of the original model compared to the final model. The percentage of testing images below an average error of 3 for the original model is 20%, compared to 80% for the final model. The maximum average error has nearly halved from around 10 in the original model to around 5 in the final model.

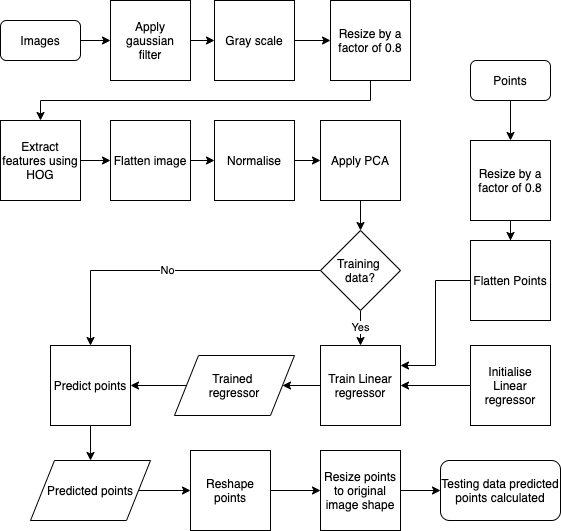


***Figure6: Cumulative density of the average error between actual and predicted points for the 703 test images***



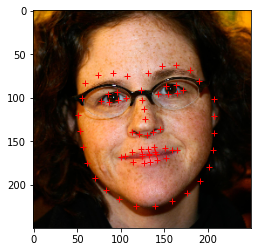
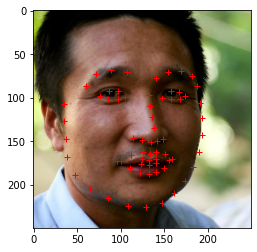
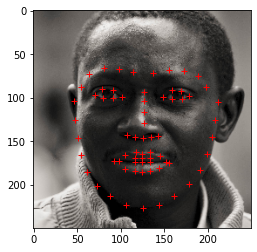
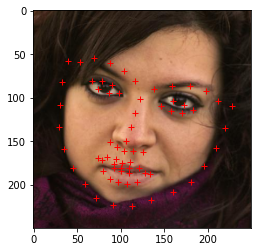
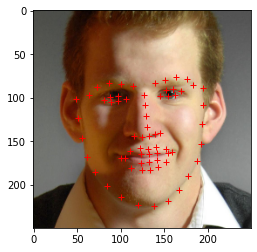
***Figure5: The Euclidean distance of the original regressor and the final regressor***

***Flowchart1: The process of the final face alignment model***



The model was used to predict the facial points of the given example images, seen in *image4.* Overall, the model has performed well at predicting the landmarks. There is an obstruction (the glasses) in the sixth image, however, the predictions are nearly perfect for both eyes. Nevertheless, as the glasses are covering the left eyebrow, the model has predicted the eyebrow points a lot higher than the true values. The shadows/brightness on the images seem to affect the predictions, for example, in first and third image, the nose points (28-31) are to the right of the actual landmark points. To improve on this, I would add to the preprocessing step to eliminate such big contrasts of the brightness.

The third image’s predicted points are very inaccurate, this is because it’s a very expressive face that is very different from most of the training images. The model maybe has been overfitted to the more common faces in the training data therefore leading to an inaccurate prediction for this image.

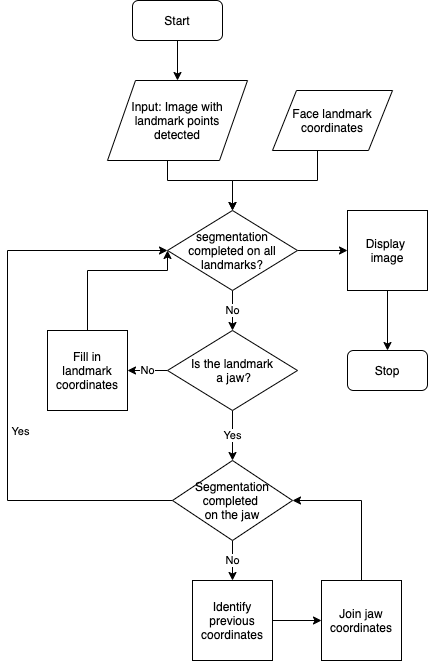


***Image4: The final face aligner method used to predict the points on the example images.***

In conclusion, if I had more time, I would create and compare other regressors, including a cascaded regressor as reports state that it produces more accurate results. Supervised descent method is another method that uses several iterations to produce good results that would improve the model.

I think that the face alignment model produces, on the whole, accurate results and the aim of minimizing has been achieved, however, using other regression techniques, I think that the model could minimize this further.

# Face Segmentation

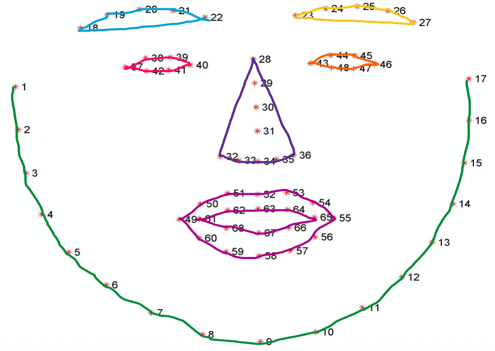


***Flowchart3: Shows the process of the face segmentation***

The aim is to split the face into different facial segments using the facial points (points 1-68) predicted in the facial alignment section, through the process indicated in *flowchart3*.

*Image5* shows the plan for segmenting the face into landmarks with the corresponding location points: jaw line 1-17, left eyebrow 18-22, right eyebrow 23-27, nose 28-36, left eye 37-42, right eye 43-48, mouth 49-68.

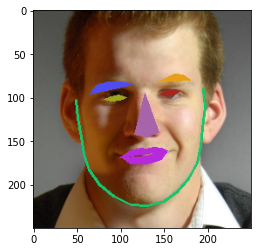
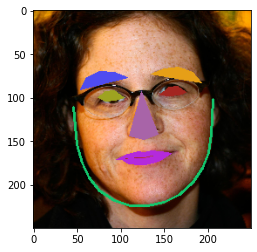
I will be using the drawing functions from open cv; for the jaw I am using cv.line() method and cv.fillPoly() to fill in the rest.



***Image5: Face segmentation plan***

## Results

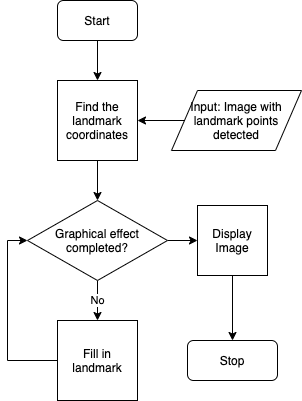
*Image6* shows the example images with face segmentation applied. The segmentation of the images is more accurate when the points predicted from the face alignment model is more accurate, e.g. referring back to *image4,* the third example image’s points are less accurate, leading to less accurate segmentation. However, images such as the fifth, have been segmented correctly as the predicted points are more accurate.



***Image6: Face segmentation performed on all example images***

Key: jaw=green, left eyebrow = blue, right eyebrow = orange, left eye = yellow, right eye = red, nose = purple, mouth=pink

# Graphical Effects



***Flowchart4: Shows the process of the graphical effects***

## Outline

I decided to use the know landmarks predicted in the face alignment to add an effect by colouring in the eyes and mouth, the process is shown in *flowchart3*. I am expecting the images to be a bit creepy. I will again be using cv2.fillpoly() to implement this effect.

## Results

In *image7*, the graphic effect has been applied to the example images. As expected, the images with accurate face alignment has more accurate effect applied, for example image 5 has come back with a good result, whereas image 3 is a failure case.



***Image7: Graphical effect performed on all example images***

# References

1. Paperswithcode.com. 2020. *Papers With Code - Face Alignment*. [online] Available at: <https://paperswithcode.com/task/face-alignment> [Accessed 7 May 2020].
2. Santos, N., 2019. *Python Opencv: Converting An Image To Black And White*. [online] techtutorialsx. Available at: <https://techtutorialsx.com/2019/04/13/python-opencv-converting-image-to-black-and-white/> [Accessed 8 May 2020].
3. Singh, A., 2019. *Feature Descriptor | Hog Descriptor Tutorial*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/> [Accessed 10 May 2020].
4. Tyagi, D., 2019. *Introduction To SIFT( Scale Invariant Feature Transform)*. [online] Medium. Available at: <https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40> [Accessed 10 May 2020].
5. Tyagi, D., 2019. *Introduction To SURF (Speeded-Up Robust Features)*. [online] Medium. Available at: <https://medium.com/data-breach/introduction-to-surf-speeded-up-robust-features-c7396d6e7c4e> [Accessed 10 May 2020].
6. Tyagi, D., 2019. *Introduction To ORB (Oriented FAST And Rotated BRIEF)*. [online] Medium. Available at: <https://medium.com/data-breach/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf> [Accessed 10 May 2020].
7. Tyagi, D., 2019. *Introduction To FAST (Features From Accelerated Segment Test)*. [online] Medium. Available at: <https://medium.com/data-breach/introduction-to-fast-features-from-accelerated-segment-test-4ed33dde6d65> [Accessed 10 May 2020].
8. Tyagi, D., 2019. *Introduction To BRIEF(Binary Robust Independent Elementary Features)*. [online] Medium. Available at: <https://medium.com/data-breach/introduction-to-brief-binary-robust-independent-elementary-features-436f4a31a0e6> [Accessed 10 May 2020].
9. Goonewardana, H., 2019. *PCA: Application In Machine Learning*. [online] Medium. Available at: <https://medium.com/apprentice-journal/pca-application-in-machine-learning-4827c07a61db> [Accessed 12 May 2020].
10. Computer vision for dummies. n.d. *Feature Extraction Using PCA - Computer Vision For Dummies*. [online] Available at: <https://www.visiondummy.com/2014/05/feature-extraction-using-pca/> [Accessed 12 May 2020].
11. Uberoi, A., n.d. *Bilateral Filtering*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/python-bilateral-filtering/> [Accessed 16 May 2020].
12. Opencv-python-tutroals.readthedocs.io. 2013. *Smoothing Images — Opencv-Python Tutorials 1 Documentation*. [online] Available at: <https://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_imgproc/py\_filtering/py\_filtering.html> [Accessed 16 May 2020].
13. Opencv-python-tutroals.readthedocs.io. 2013. *Image Gradients — Opencv-Python Tutorials 1 Documentation*. [online] Available at: <https://opencv-python-tutroals.readthedocs.io/en/latest/py\_tutorials/py\_imgproc/py\_gradients/py\_gradients.html> [Accessed 16 May 2020].
14. Demir, N., n.d. *Ensemble Methods: Elegant Techniques To Produce Improved Machine Learning Results*. [online] Toptal Engineering Blog. Available at: <https://www.toptal.com/machine-learning/ensemble-methods-machine-learning> [Accessed 18 May 2020].
15. Xiong, X. and De la Torre, F., n.d. *Supervised Descent Method And Its Applications To Face Alignment*. [online] Available at: <https://www.ri.cmu.edu/pub\_files/2013/5/main.pdf> [Accessed 19 May 2020].
16. Cao, X., Wei, Y., Wen, F. and Sun, J., 2012. *Face Alignment By Explicit Shape Regression*. [online] Microsoft. Available at: <https://www.microsoft.com/en-us/research/wp-content/uploads/2013/01/Face-Alignment-by-Explicit-Shape-Regression.pdf> [Accessed 20 May 2020].
17. Learned-Miller, E., Huang, G., RoyChowhury, A., Li, H. and Hua, G., 2007. *Labeled Faces In The Wild: A Survey*. [online] Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.696.1377&rep=rep1&type=pdf> [Accessed 20 May 2020].
18. Singh, A., 2018. *K-Nearest Neighbors Algorithm*. [online] Analytics Vidhya. Available at: <https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/> [Accessed 22 May 2020].