# Computer Vision Assignment: Achieving Millisecond Facial Alignment with a Cascade of Gradient Boosted Trees

198680

University of Sussex, Computer Science & Artificial Intelligence BSc

## Table of Contents



Main text word count including capations, exlcluding references: **1597**

# Introduction

For this project we were asked to design a system that could align a set of 68 facial landmarks to unseen face images. I designed my system using an ensemble of gradient boosted trees that uses binary features to split samples at each of the tree's nodes (Sullivan & Kazemi, 2014). A binary feature is a simple test that can be applied to a set of images - choose two pixels, if pixel one is greater than pixel two return true. The method also runs over several stages, at each stage a new ensemble of gradient boosted trees is generated along with a new set of 400 binary features. These binary features are transformed in relation to the current prediction of the face shape (Sullivan & Kazemi, 2014). The aim of the algorithm is to minimise the sum of squared errors over all training samples whilst avoiding overfitting.

To develop on work seen in other papers, I made several novel adjustments: the feature transform used between stages and the use of different patch sizes to perform the binary feature tests (i.e. a 3x3 pixel patch). In this paper I will discuss and justify each step of my model including all pre-processing steps and optimisation techniques. I also highlight the failure cases in my model and offer potential solutions to address them. I conclude with a short summary of a method for face segmentation, along with a method for applying basic graphical effects.

# Facial Alignment Model

This section outlines each step of my model, demonstrated by a random subset from my labelled training data:

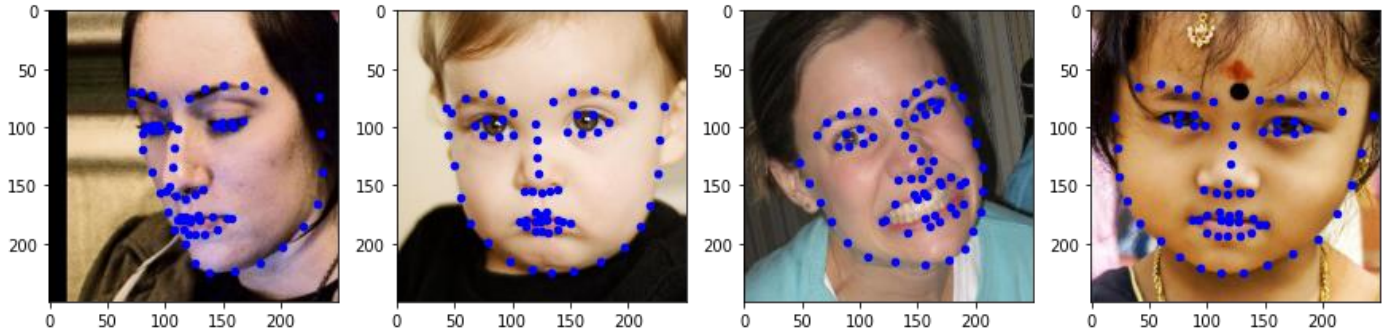



*Figure 1 - A random example subset to demonstrate steps taken in my model. This is the subset before any steps are taken. Ground truth shown in blue. (Menneer, 2020).*

## Train Test Split

This is a supervised machine-learning task and therefore 20% of my labelled data is used as validation set to evaluate the performance of my model after training.

## Pre-processing steps

1) All images were converted to grayscale to make the model colour invariant i.e. light hue shouldn't affect my model and binary features only need a single intensity value for each pixel.
2) Image resolution was reduced to 50x50, reducing noise making binary features a more effective test and speeding up training time.
3) Image normalisation was used to scale pixel values into a standard range. This improves image contrast and introduces a standard range for splitting samples in decision trees.
4) A Gaussian blur is applied with a 3x3 kernel reducing image noise.
5) Histogram equalisation was applied stretching out intensity ranges of the images, this improves lighting conditions and allows more information to be extracted from an image.

1. **Convert Images to Grayscale**

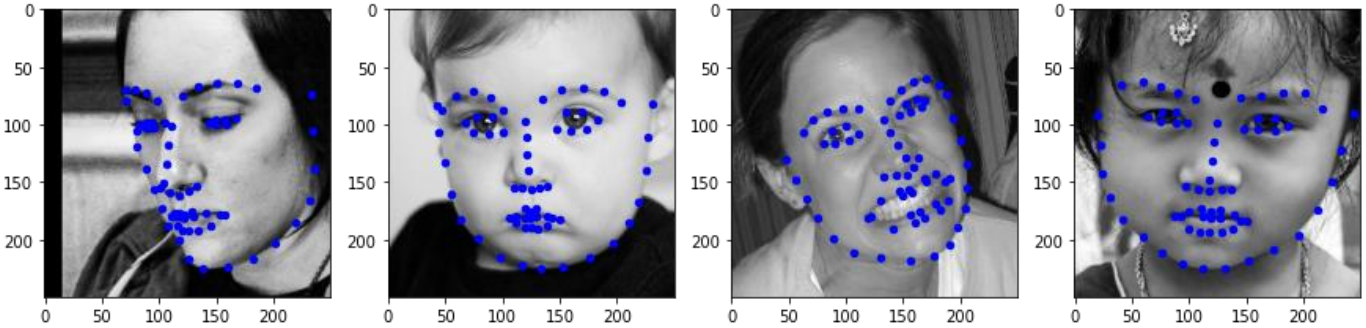


*Figure 2 - Converted to grayscale using CV2. (Menneer, 2020).*

2. **Reduce Image Resolution**

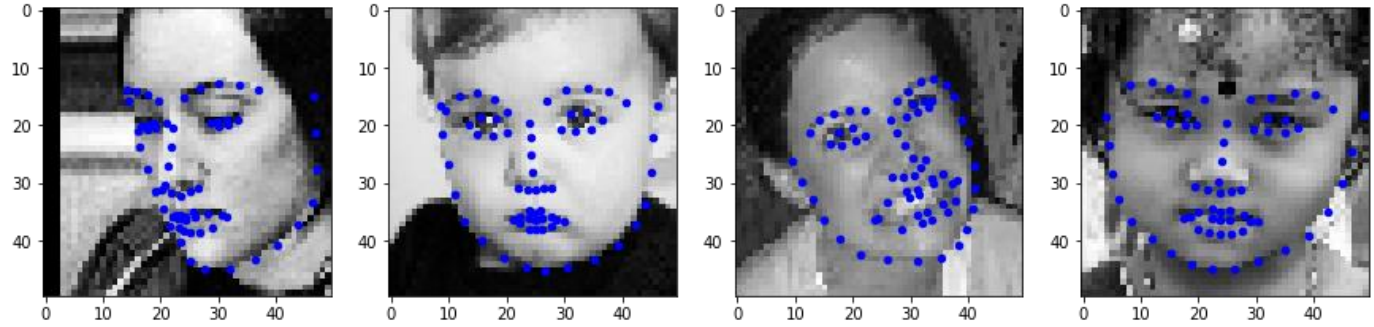


*Figure 3 - Resolution reduced from 250x250 to 50x50 using cv2. (Menneer, 2020).*

3. **Image Normalisation** – image omitted as no visible change is apparent
4. **Apply Gaussian Blur**

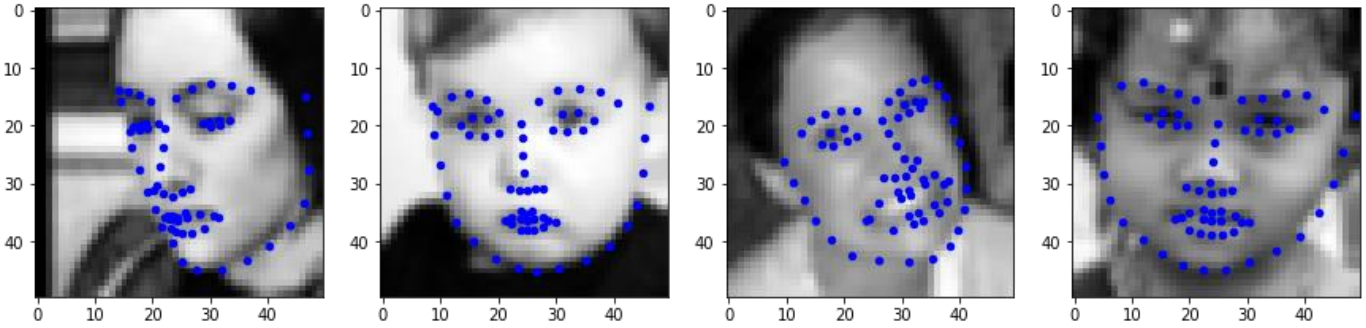


*Figure 4 – Gaussian blur applied. (Menneer, 2020).*

5. **Apply Histogram Equalisation**

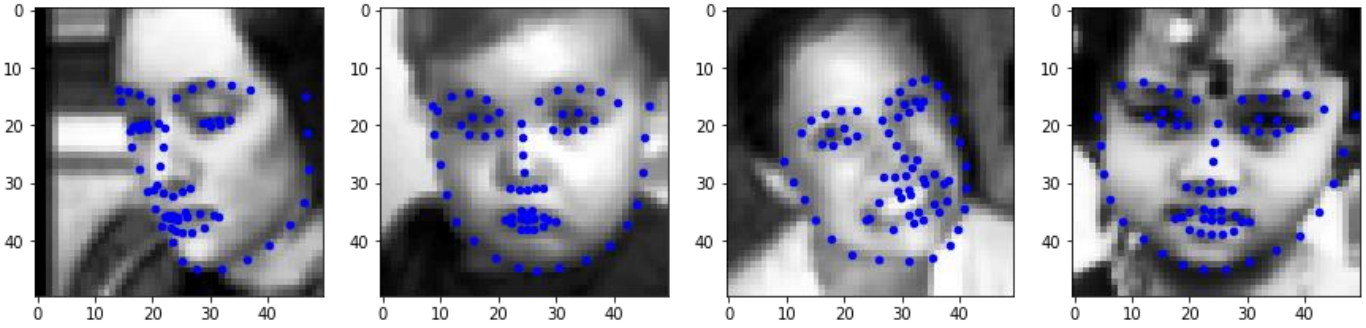


*Figure 5 – Histogram equalisation applied. (Menneer, 2020).*

## Generate the Mean Face Shape – Shape 0

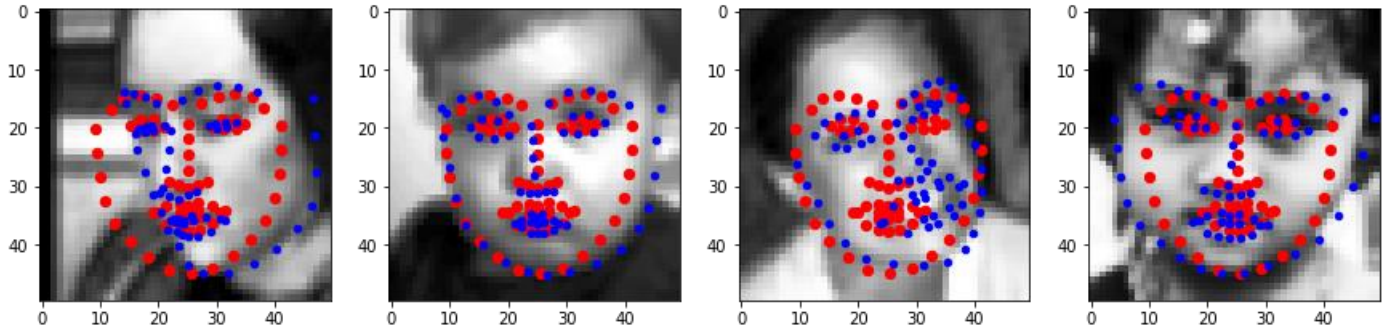


*Figure 6 – The mean face shape or 'shape 0' in red, the ground truth for each image in blue (Menneer, 2020).*

For cascaded regression we need an initial guess. The initial guess (shape 0) is the first prediction of our model, we then create our gradient boosted trees to reduce the residual error between our current prediction (initially shape 0) and the ground truth.

## Training the Ensembles of Gradient Boosted Trees

A model was developed from scratch following the gradient boosting algorithm, no prebuilt models were used. The model aims to minimise the squared error by using the derivative of the function (known as the residual error):

$$Squared\ error = \frac{1}{2}(Ground\ truth\ - Current\ prediction)^2$$

$$Residual\ error = (Ground\ truth\ - Current\ prediction)$$

The error function minimisation is achieved through the summation of many gradient boosted trees. Each tree is built recursively until the max depth is reached, at which point the average of all residuals within the same leaf are used to update the current prediction of the face shape, these are known as the delta steps $\Delta S$. The model's prediction of a sample after $n$ iterations is equivalent to (See figure 16 on page 9 for illustration):

$$Shape^{X} = Shape^{X-1} + \sum_{i=0}^{i=n} \Delta S^{i} \cdot learning\ rate$$

*Note: This is the models prediction after each stage of regressors, initally* $Shape^{X-1}$ *is Shape 0.*

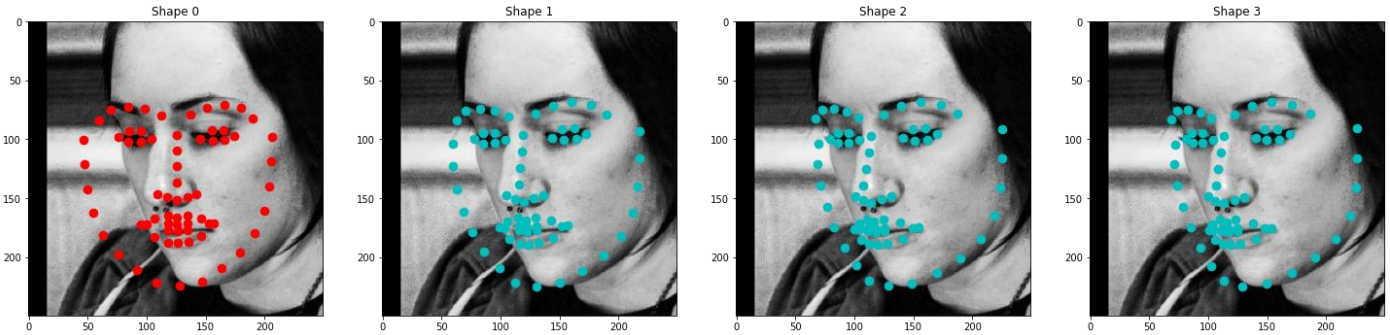



*Figure 7 – From Shape 0 (left) to Shape 3 (right) after each stage of regressors has been applied. Predictions are linear combinations of labelled examples meaning the result is always semi-realistic. (Menneer, 2020).*

## Generating a Set of Binary Features for Each Stage

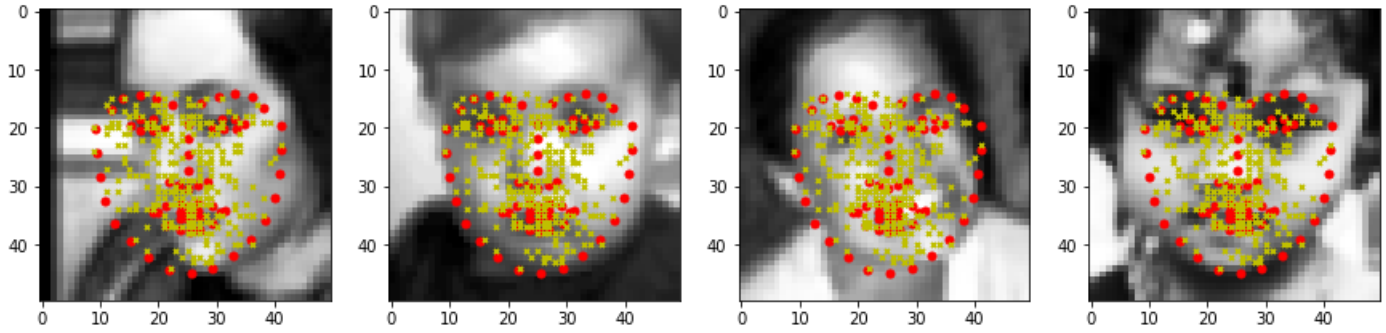


*Figure 8 – An example of 400 random points used for binary feature tests (shown in yellow), these are extracted from within the current prediction. Mean face shape is shown in red. (Menneer, 2020).*

To facilitate binary feature tests, points are extracted from the images. Pairs of these are used to divide candidates at each of the tree's nodes. Pairs of pixels are used as they are stronger features than single pixels due to their insensitivity to lighting conditions (Sullivan & Kazemi, 2014).

In Sullivan and Kazemi's 2014 paper they suggest the use of an exponential prior to select points close to one another, as they were found to produce better split candidates. They also suggest a similarity transform to translate features at the start of each stage, so points are approximately the same position relative to their current shape.

Here I propose a new solution, I extract '*true features*' by selecting two facial landmarks on the mean face shape and a random distance (%) between them. At run time these are transformed to '*run time features*' using the current prediction of the face shape (see figure 9). This method adds a probabilistic prior to selecting points around the centre of the face and allows for computationally efficient transformation of features.

$$True\ feature = (Landmark\ 1, Landmark\ 2, distance\ between\ (\%))$$

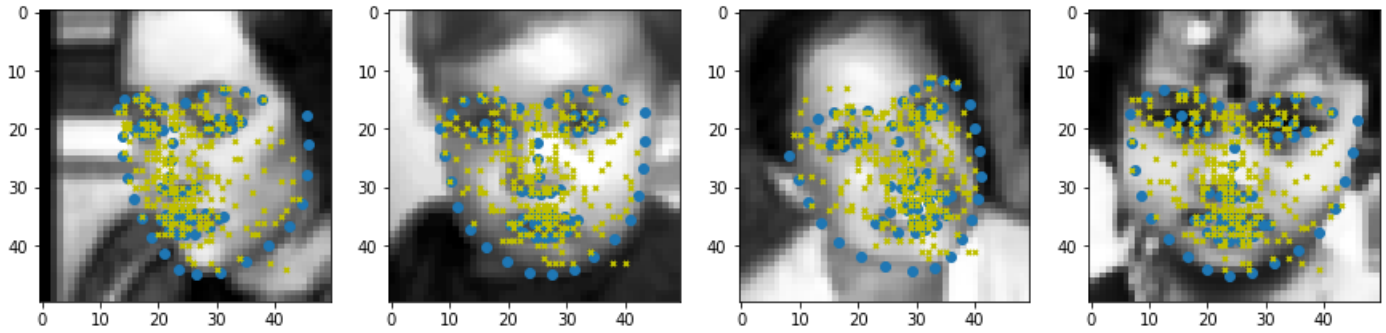


*Figure 9 - 400 random points (in yellow) used for binary feature tests transformed to the current shape prediction of the face. This is done at the start of each stage to position points relative to current estimate. Current prediction shown in blue. (Menneer, 2020).*

Inspired by the Viola and Jones method I also included the use of different rectangular patch-sizes when conducting binary tests (Viola & Jones, 2001).

## Reducing Overfitting and Hyperparameter Tuning

A stochastic method was used to prevent overfitting, meaning each tree was built using a subsample of all of the training data. Other techniques were also used to reduce overfitting such as minimum leaf weight meaning that if too few samples fall into a leaf no delta step is applied to update the shape. In order to tune the hyperparameters, a grid search technique was used and validated against my validation set. Parameters used:

| **Parameter** | *Stages* | *Iterations* | *Depth* | *Subsample* | *Min-weight* | *Learning-rate* |
|---|---|---|---|---|---|---|
| **Value** | 10 | 500 | 6 | 60% | 5 | 0.01 |

## Model Evaluation

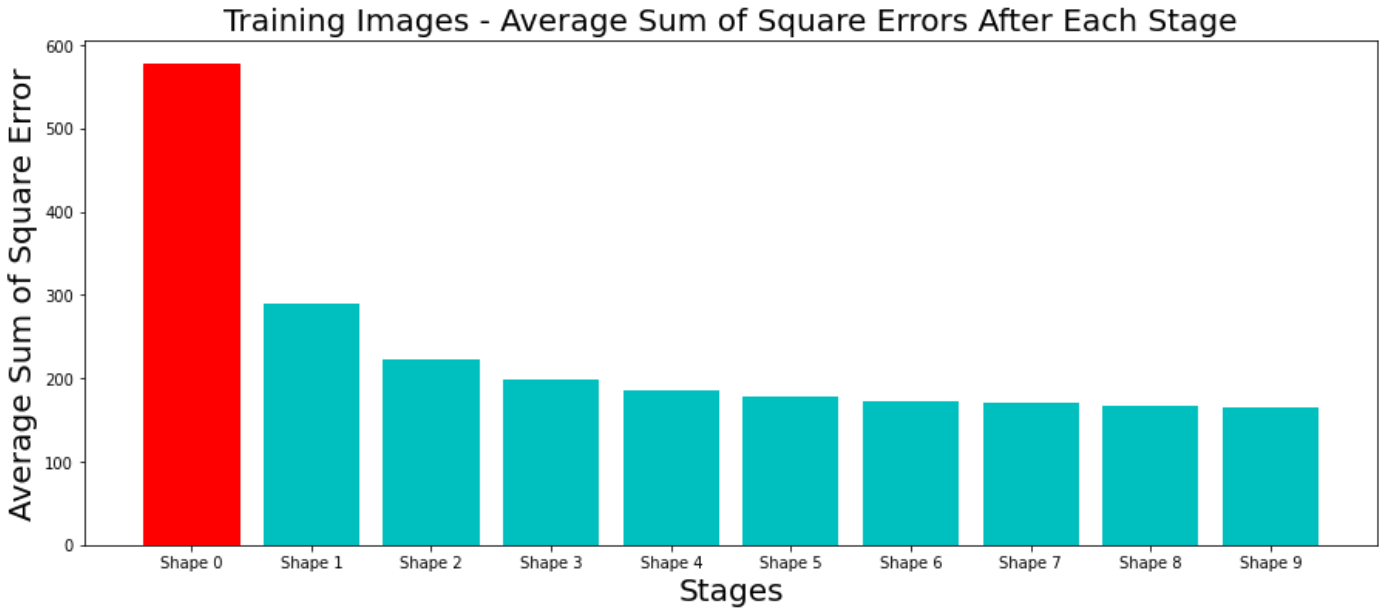



*Figure 10 – Average sum of square errors for training images over 10 regression stages. Notice the most substantial reduction in error happens in the first few stages. Stage 10 average error:* ***165.13*** *(Menneer, 2020).*

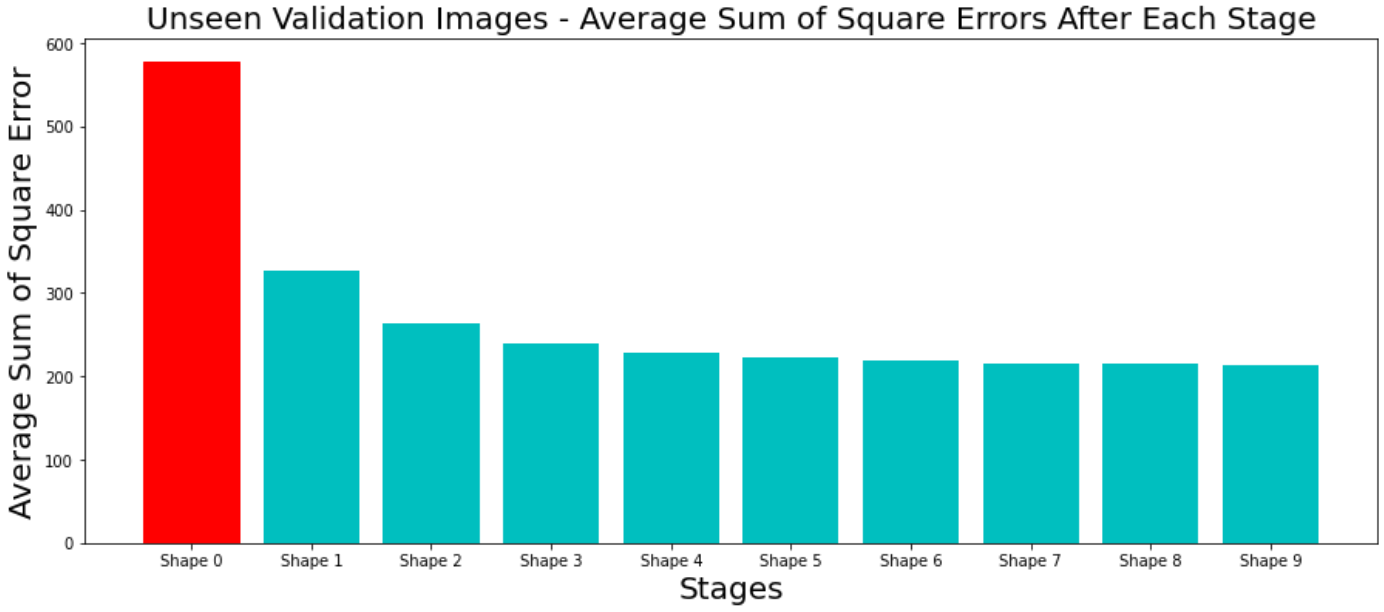



*Figure 11 – Average sum of square errors for validation images over 10 regression stages. Stage 10 average error:* ***212.91*** *(Menneer, 2020).*

## Applying the Model to Unseen Images

Once the trees have been created, these can be applied to unseen images by applying the same binary tests and delta steps. Although training time is large over 10 stages, run time application can be performed in milliseconds.

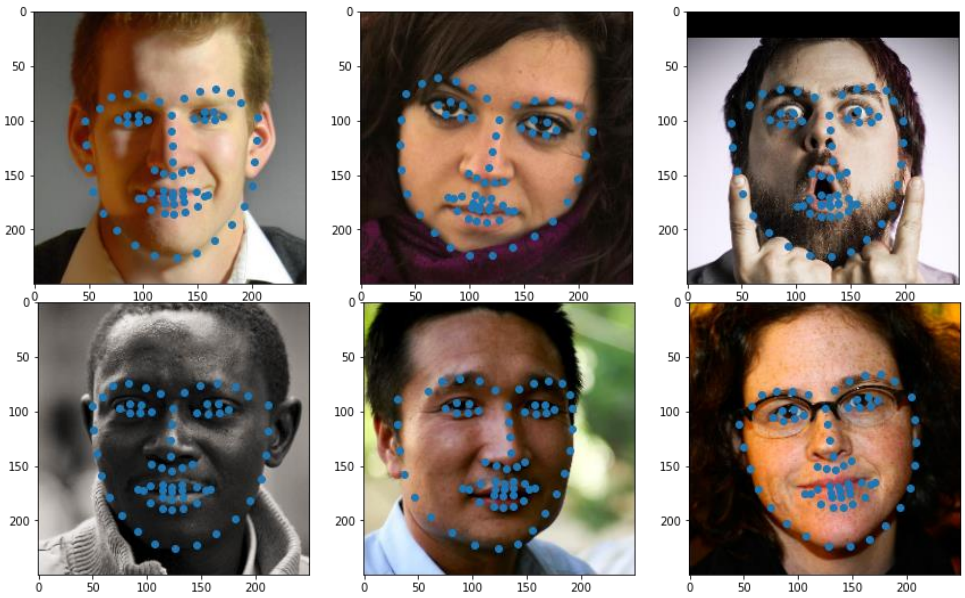


Figure 12 - Facial landmarks aligned on unseen example images after application of cascaded regression model (Menneer, 2020).

## Failure Cases and Critical Analysis

There were several weaknesses present in my model, this was particularly apparent in samples that contained strange poses or open-mouthed positions. Since the majority of the labelled data did not fall into one of these categories there is a clear imbalance in my dataset for edge cases which cascaded regression is not particularly resilient to. One solution would be to up-sample edge cases. Another solution would be to include the addition of multiple starting poses, although initialising the correct starting pose is a difficult task within itself.

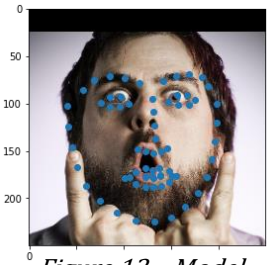


*Figure 13 – Model failure on edge case image with open-mouth position (Menneer, 2020)*

# Face Segmentation

To provide a simple solution for face segmentation the following steps were taken:

1) Isolate the forehead region of the image by slicing between landmark 1 and landmark 17 – the top of the left and right jaw respectively.
2) Apply K nearest neighbours where K = 2 to the forehead section of the image.
3) Apply an opening operation to retain shape and remove any small image noise.
4) Find contours of shapes within this section, discard all but the largest shape.
5) Take the top 20% of contour points to find top edge of the forehead.
6) Sort contour points and jaw points 1-17 into clockwise arrangement.
7) Use fill poly to connect points.

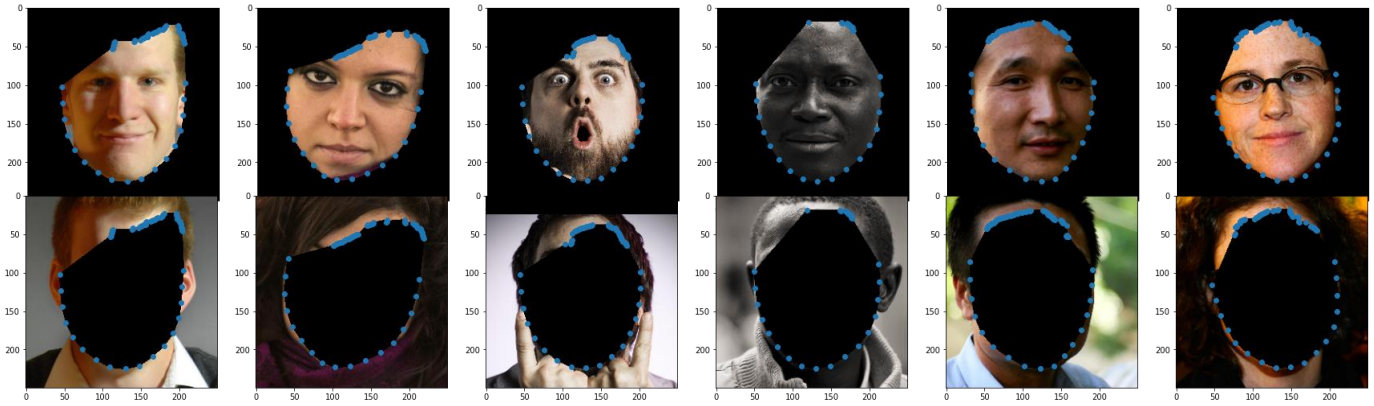
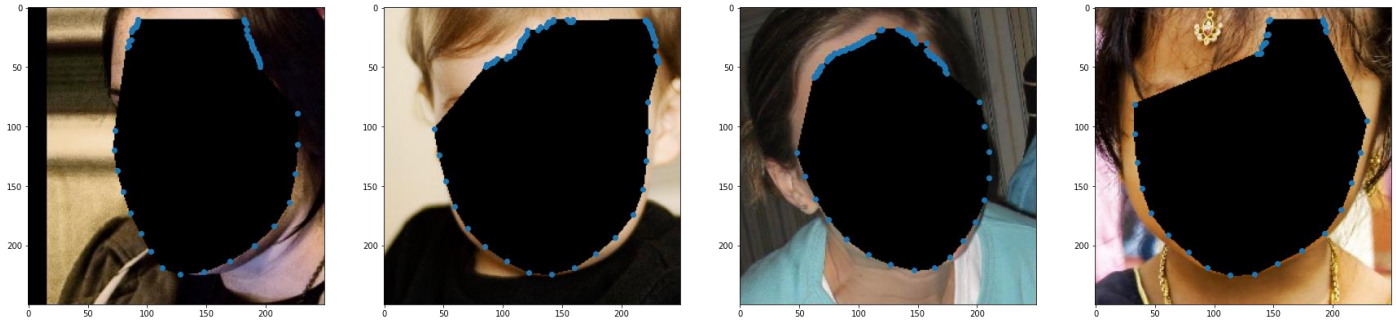


*Figure 14 – Example images(above) training images (below) after face segmentation solution has been applied. Also shown in blue are landmark points 1 to 17 and the top edge points of forehead shape found with contouring and k nearest neighbour's method described. (Menneer, 2020).*



## Simple Graphical Effect

I implement a method for enlarging the eyes and mouth and then apply an effect to each of them. The following the steps were taken:

1) Isolate landmark coordinates for each face feature.
2) Individually enlarge these cut out features of the face.
3) Apply a simple effect to each feature – here histogram equalisation was chosen.
4) Overlay altered features to original image.

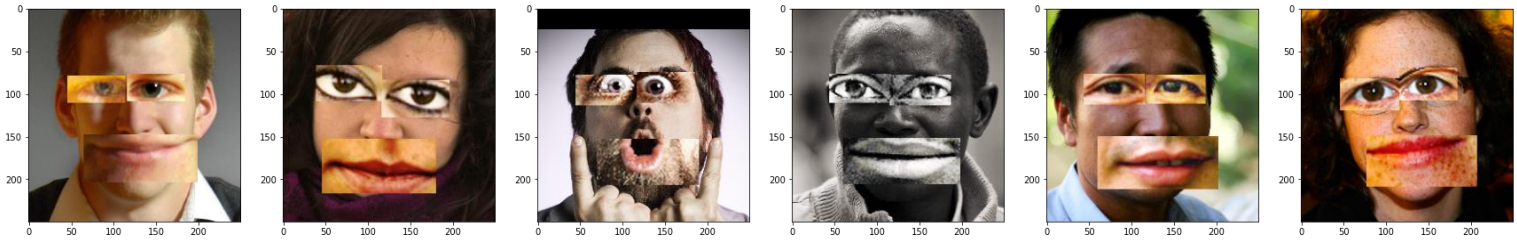
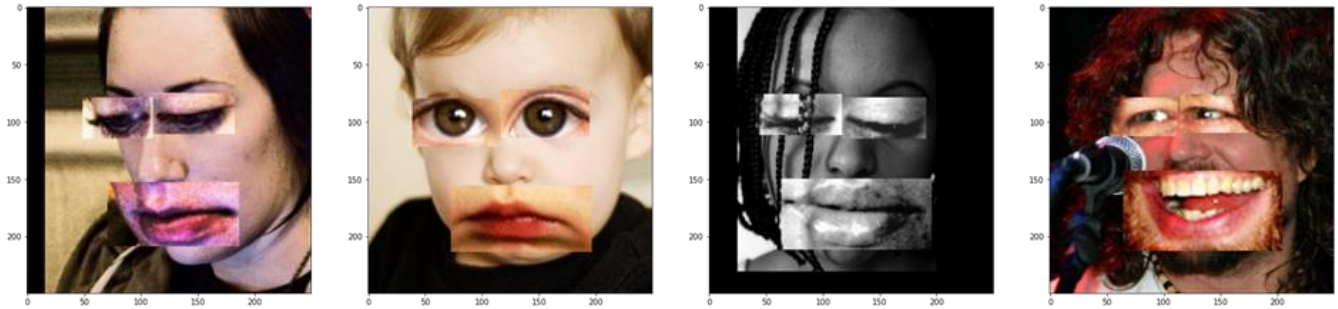


*Figure 15 – Example images(above) training images (below) after simple graphical effect applied (Menneer, 2020).*



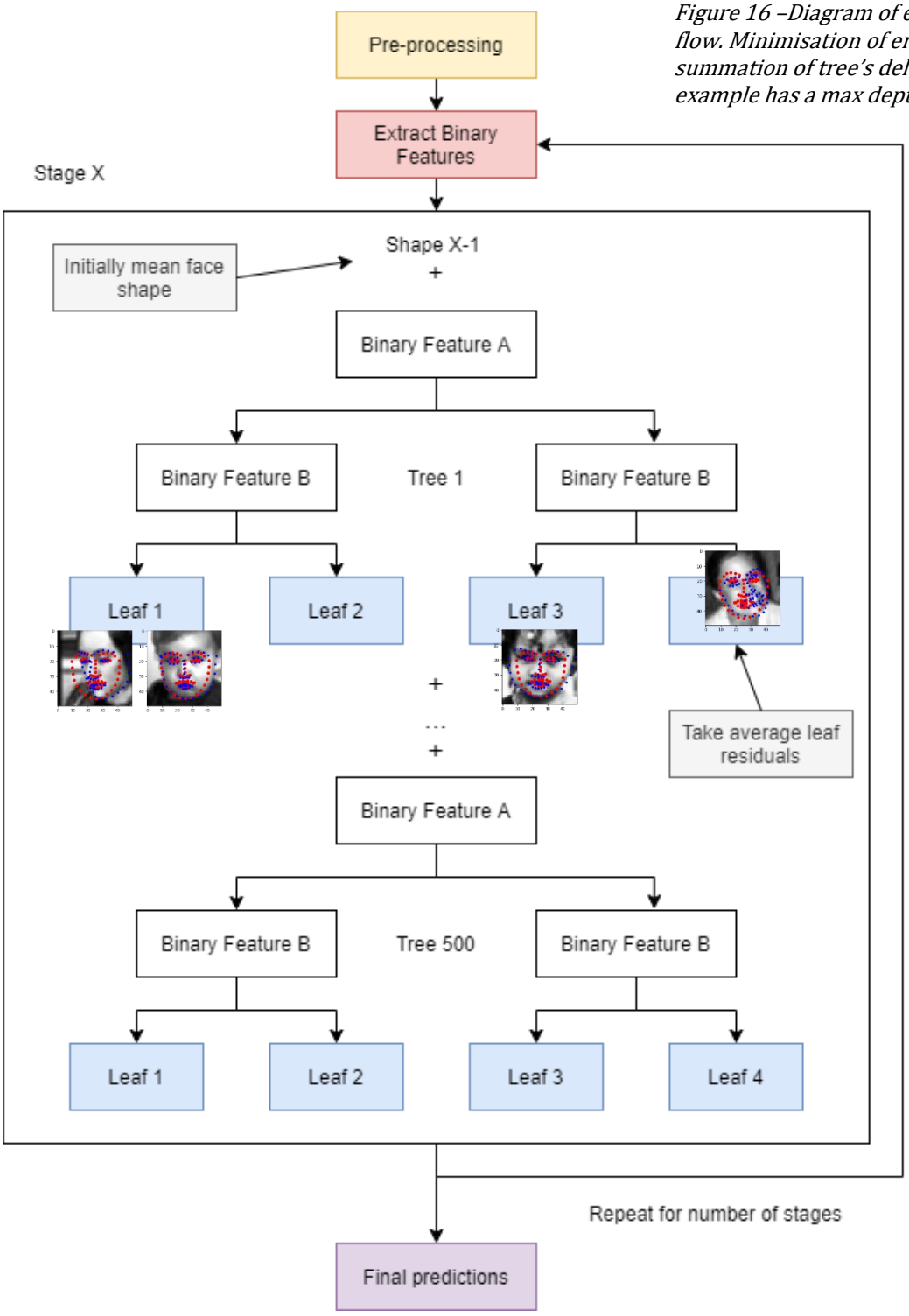



*Figure 16 –Diagram of example model flow. Minimisation of error through summation of tree's delta steps. This example has a max depth of 2.*

# Bibliography

Menneer, R., 2020. *My own work, all images and graphs created and exported in Python on the Google Collab environment,* Brighton: Menneer, Rupert.

Sullivan, J. & Kazemi, V., 2014. *One Millisecond Face Alignment with an Ensemble of Regression Trees,* Stockholm: CVPR2014.

Viola, P. & Jones, M., 2001. *Rapid Object Detection using a Boosted Cascade of Simple Features,* Cambridge: ACCEPTED CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION.