

# Generation of Facial Cartoons

School of Computer Science and Informatics  
Cardiff University

CM3203 – One Semester Individual Project  
May 2017

Student: Wai Yin Leung (C1424864)

Supervisor: Paul L Rosin

Moderator: YuKun Lai

# Abstract

The idea behind a generation of facial cartoons is to create a cartoon stylized version of a given input facial image. In today's social applications, there are many cartoon styling tools that allow users to browse through and select their favourites. However, this can take a long time and still not provide the wanted results.

This project aims to solve this problem by developing a program that automatically generates a cartoon stylized version of a frontal face image using a combination of image processing techniques with an image library.

This project uses an image library, which is a collection of facial components. The library is created using a program called "OpenFace" to locate then extract crucial features including eyebrows, eyes, nose and mouth from both input images and their cartoon stylized version reproduced from [14]. When the user inputs an image, the program will extract the features and compare with the library. There are 6 different matching techniques and for each feature, each technique will cast a vote towards one feature image and the feature image with the most votes will be selected. Then the selected features are placed as rectangles onto a hair image generated using "Coherent-line-drawing" with their original orientations.

One of the limitations of this project is the lack of image and cartoon pairs, hence insufficient examples in the library. The total number of examples are 28 pairs. Therefore, this project implements both rigid and non-rigid transformations to minimise the impact of the inaccuracies of selected feature images.

The results show that the features in the output images before the rigid and non-rigid transformations do show approximate accuracies in their positions and sizes due to correspondence in position placements and feature resize before matching. However, they do not provide any rotations, meaning if the input images have a different orientation such as a 20-degree difference will generate poor results. Because of this, it showed the significance of the rigid and non-rigid transformations to minimise these errors.

The number of test images ran was 54 in total. The results after rigid transformation provide the correct orientations with a low percentage error but the shapes of the features might not be as accurate due to limited examples. The non-rigid transformation provides both positive and negative results, where positive results include better fit to the shape of the input images comparing to results after rigid transformation. However, some features that are deformed into more realistic shapes may give a less attractive output.

After the further discussions and analysis on whether which approach is better to suit our aim of balance between realism and attractiveness, overall, the results after non-rigid transformation show the best balance and we will be using that as our final results.

## Acknowledgements

I would like to express my appreciation to my supervisor Paul L Rosin for his guidance and inspiration throughout the project.

I would also like to thank my personal tutor Yukun Lai for our project discussion and provides suggestions for improvements.

I would like to thank our guest Professor David Mould (Carleton University) for his valuable suggestions.

Lastly, I would like to thank the participants for comments on results and output.

# Table of Contents

1	Introduction .....	8
2	Background .....	9
2.1	Related work .....	10
2.1.1	“Data-Driven Synthesis of Cartoon Faces Using Different Styles” [14] .....	10
2.1.2	“Face Cartoon Synthesis Based on the Active Appearance Model” [12] .....	13
2.1.3	Example-based Facial Sketch Generation with Non-parametric Sampling [4] .....	14
2.1.4	Other related work .....	15
2.2	OpenFace Program .....	16
2.3	Coherent-Line-Drawing Program .....	17
3	Specification and Design .....	19
3.1	Pipeline .....	19
3.2	OpenFace system to extract facial features .....	20
3.3	Hair segmentation using histogram analysis .....	21
3.4	Coherent Line Drawing to generate line images .....	22
3.5	Matching algorithms .....	23
3.5.1	Mean Squared Error .....	23
3.5.2	Peak signal-to-noise ratio .....	24
3.5.3	Structural Similarity Index (SSIM) for measuring image quality .....	24
3.5.4	Intensity mismatch .....	24
3.5.5	Cosine similarity .....	25
3.5.6	Absolute distance measure .....	25
3.5.7	Potential problem with the similarity measures .....	25
3.6	Open mouth recognition .....	26
3.7	‘bwareaopen’ function to remove small objects .....	26
3.8	Rigid transformation .....	27
3.8.1	Pixel image registration using fminsearch .....	27
3.8.2	Pixel image registration using fminsearch grid search .....	28
3.8.3	Rotation using OpenFace coordinates .....	28
3.9	Non-rigid image registration using thin-plate-spline .....	29

4	Implementation and Testing.....	31
4.1	Choosing the bounding rectangles face scale value .....	31
4.2	Coherent-line-drawing on features.....	32
4.3	Hair segmentation using colour histogram analysis .....	32
4.4	'bwareaopen' .....	35
4.5	Open mouth recognition.....	36
4.6	Matching .....	37
4.6.1	User study on similarity measure .....	38
4.6.2	User study on feature state matching .....	39
4.7	Choice of method in rigid transformation .....	40
4.7.1	Pixel image registration using fminsearch.....	40
4.7.2	Grid search .....	40
4.7.3	Affine transformation using OpenFace coordinates .....	41
4.8	Implementing non-rigid transformation.....	41
4.8.1	Finding the coordinates of the landmarks.....	42
4.8.2	Relocating features .....	42
4.8.3	Testing the implementation .....	43
4.9	Other implementations.....	43
4.9.1	Perform transformations on bad examples.....	43
4.9.2	Perform feature warping before matching .....	44
5	Results and Evaluation.....	45
5.1	Evaluation Approach .....	45
5.2	Hair Segmentation .....	45
5.3	Hair and Facial Lines.....	46
5.4	Transformation Progression.....	48
5.5	Transformation on Bad Examples .....	50
5.6	Feature Warping before matching.....	51
5.7	Final Results.....	53
5.7.1	Limitations.....	56
5.7.2	Overall System Evaluation .....	57
6	Future Work .....	58

## Generation of Facial Cartoons

6.1	Increase library size .....	58
6.2	Facial expression recognition .....	58
6.3	The direction of vision .....	58
6.4	More details on the face .....	59
6.5	Create a background .....	59
6.6	Multiple faces in one image .....	59
7	Conclusions .....	60
8	Reflection on Learning .....	61
9	Reference .....	62
10	Appendix .....	64

## Table of Figures

Figure 1. Overall pipeline for “Data-Driven Synthesis of Cartoon Faces”. Reproduced from [14].	10
Figure 2. Bounding rectangles calculation using facial landmarks. Reproduced from [14].	11
Figure 3. The coordinate system used to represent features locations. The location of the features is represented by a 13-dimension vector, where $x$ and $y$ are the coordinates and $w$ is the width. Reproduced from [14].	11
Figure 4. Results of hair extraction. Reproduced from [14].	12
Figure 5. Input images and their corresponding generated facial cartoon with all matching features. Reproduced from [14].	12
Figure 6. Process of hair segmentation. Reproduced from [12].	13
Figure 7. (a) and (d) are input images, (b) and (e) are artist drawing, (c) and (f) are their results. Results reproduced from [12].	13
Figure 8. Overall pipeline for “Example-based Facial Sketch Generation with Non-parametric Sampling”. Reproduced from [4].	14
Figure 9. Input images and their corresponding generated facial cartoon. Reproduced from [4].	15
Figure 10. From left to right: Frontal face image as input for OpenFace program. Output image with landmarks. Frontal cartoon image as input for OpenFace program. Output cartoon with landmarks.	16
Figure 11. Images showing Open Face detection error.	17
Figure 12. Frontal face image as input for Coherent-Line-Drawing Program and their corresponding output line image.	17
Figure 13. Process overview for Coherent-line-drawing. Reproduced from [5].	18
Figure 14. The pipeline of creating features library of this project.	19
Figure 15. The overall pipeline of this project.	19
Figure 16. Landmarks numbering system for OpenFace program.	20
Figure 17. Image showing landmarks on a left eye with an approximate bounding rectangle. ..	20
Figure 18. Image showing the upper part of an input frontal face image with an approximate region for hair histogram analysis.	21
Figure 19. Image showing orientation of a left eye.	28
Figure 20. From left to right: Part A, image showing wanted right eye orientation. Part B, image showing current right eye orientation. Part C, image showing how orientations are found, the yellow line represents when $Y_2$ is larger than $Y_1$ , the orange line represents when $Y_2$ is larger than $Y_1$ , the orange line represents when $Y_1$ is larger than $Y_2$ . Left top corner ( $X_1, Y_1$ ), right top corner ( $X_2, Y_1$ ), Left bottom corner ( $X_1, Y_2$ ), right bottom corner ( $X_2, Y_2$ ).	28
Figure 21. Images showing extracted features	31
Figure 22. Extraction of eyebrows capturing edges of the eyes.	32

Figure 23. Results of hair segmentation. ....	33
Figure 24. Bad results for hair segmentation. ....	33
Figure 25. Left to right: Line drawing image generated using Coherent-line-drawing. Line drawing after colour modification and features removal. ....	35
Figure 26. Left to right: Line drawing image generated using Coherent-line-drawing. Line drawing created by applying 'bwareaopen' function. ....	36
Figure 27. Left to right: Input image. SSIM result. Voting system result. ....	38
Figure 28. Table show results of user study on similarity measures. ....	38
Figure 29. Table showing results from user study on feature state matching. ....	39
Figure 30. Left to right: Results from pixel image registration using fminsearch. Results from grid search. Results from affine transformation using OpenFace coordinates. ....	41
Figure 31. Images showing input images and their corresponding hair segmentation. ....	46
Figure 32. Left to right: Input image. Extracted line drawing. Final line drawing with facial lines. ....	47
Figure 33. Left to right: Input image. Results after scaling and translation. Results after feature rotation. Results after non-rigid transformation. Results after non-rigid transformation with enhanced landmarks difference on mouth. ....	49
Figure 34. Left to right: Input image. Bad examples' results after feature rotation. Bad examples' results after non-rigid transformation. ....	50
Figure 35. Left to right: Input image. Feature warping before matching-result after feature rotation. Feature warping before matching-result after rigid transformation. ....	52
Figure 36. Finalised cartoon images with their corresponding input image. ....	55
Figure 37. Cartoon images of three different design styles. ....	56



# 1 INTRODUCTION

---

In the trend of facial enhancements and special effects in many social applications today, the demand for new and additional features is increasing. However, these features are focused on partial enhancements or effects on the actual person's face rather than recreating a facial image that is recognizable and enhanced but different from the original image taken.

To increase popularity and prove the feasibility of creating a tool that can further develop in this area, the idea of this project is to develop a program that can automatically generate a cartoon stylized version of a frontal face image using a combination of image processing techniques with many sets of training image data. There are some related projects that use Active Appearance Model to train and adjust with over 100 sets of images. However, our project has limited resources which is not sufficient to follow similar methods.

The aim of this project is to have a single cartoon image as the output that provides the aspects of attractiveness and realism for each input image. Our system can be used in different applications such as profile images in "chatting apps" and stylized cartoon characters in many games. It would be beneficial for the applications to have customized cartoon faces for their users which can improve user experience. Also, with the increase of virtual reality applications, there can certainly be areas where cartoon faces are useful.

The first objective of our project is to be able to create an image library that contains both the facial images and the paired cartoon images. Then, for each pair, the facial features are extracted in several bounding rectangles located using the 64 landmarks generated by "OpenFace system". The purpose of the image library is that when a user inputs an image, the features of the input image are extracted in the same way and compared with the image library to find the best match.

Our matchings are done in 3 different states, including "photo to photo", "line to line" and "line to cartoon". In "photo to photo" matching, we will be using the extracted feature images, whereas "line to line" is to compare line drawings created using the "Coherent-line-drawing" program. Lastly, we will be matching line drawings with the extracted cartoon features.

To further improve realism, different transformations including rigid and non-rigid will be applied to the best-matched features. In this project, we will be using affine transformations and warping using Thin Plate Splines to deform the shapes of the features. Then the features are placed in the background template in the corresponding and calculated coordinates where the background is extracted from the input image.

This report will first describe the background for this area and related projects. Then it will give details of the design and approach of this project. After that, it will explain the implementation and evaluate testing results. Finally, there will be a discussion on the future possible approaches and a summary of this project.

## 2 BACKGROUND

---

This project is under the topic “Non-photorealistic rendering (NPR). NPR is a technique that modifies contents of digital photo images into a wide range of artistic stylized images. These artistic styles include art paintings, line drawings, stippling, comics, etc. There are many uses of NPR, such as technical illustration models, video games and animated movies.

In a child cartoon, the characters are generally in a bright and colourful style to attract a child’s attention. Also, these characters are not presented in a photo-realistic way because a child can easily lose focus on them. Many cartoon animated movies have shown successes on using this non-photorealistic approach, providing both attractiveness and realism.

While the overall look is important, the appearance of the faces can have large impact on whether an image looks good, so, there should be some focus on the facial contents in generating non-photorealistic portraits.

There are many projects relating to generation of non-photorealistic portraits. The most used approach is to use an active appearance model (AAM) or an active shape model (ASM) to find the locations of the facial features by iteratively displacing the model to find the best fit. Then, an image is produced by replacing the features with the features of the portrait’s style and by modifying the background.

To create a cartoon style, some projects will use a mathematical approach such as geometric transformation or bounding box calculation to extract and alter the features to match with examples in an image library that contains the real feature photos and their corresponding cartoon image. After finding the best matches, the features are sometimes deformed to improve realism of the cartoon while some projects focus on the colour used in their results. In the end, features are composed onto the background image using pre-defined coordinates or the coordinates from the input images to form the cartoon images.

In this part of the report, we will first describe some approaches of the related work and how our project will differ in section 2.1. Then, there will be a brief description on how the “Open Face” program (section 2.2) and the “Coherent-line-drawing” program (section 2.3) works in theory. Also, we will describe how they will be used in our project and some comments on how well their results satisfy our needs.

### 2.1 Related work

#### 2.1.1 “Data-Driven Synthesis of Cartoon Faces Using Different Styles” [14]

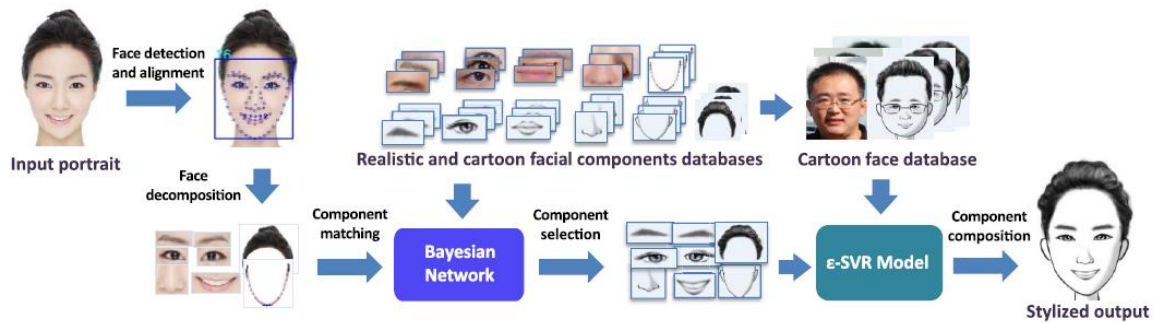


Figure 1. Overall pipeline for “Data-Driven Synthesis of Cartoon Faces”. Reproduced from [14].

Their work begins with creating two databases, one for realistic facial components collected from portrait photographs and the other for cartoon facial components which are drawn by artists. Then they learn how to select the cartoon components and assemble them together by using Bayesian network to train a selection model and  $\epsilon$ -SVR to train a composition model. Then for each feature, the five most similar components are selected. Combinations of selected features are put together to evaluate their attractiveness using trained selection model. Then their output will be the combination with the highest value with some adjustments to the feature compositions.

The principle of their pipeline (figure 1) is similar to our project except for the details in feature selection and composition. In our feature selection stage, we do not use any training model. We have feature images in three different states and using different matching techniques to vote for the most similar features. For feature composition, we use the same coordinates as how we extracted the features to place them. Lastly, they do not consider deformation. This can be done because they have an artist to create cartoon images for over 60 images per feature, which have a higher chance to have a perfect match. On the other hand, we have limited examples in our image library which will have less chance of having similar features, so we will require deformation to improve realism.

However, this means that all the result images produced by their program have the same orientation. Also, their limitation of their system is that they can only deal with frontal or near-frontal faces and they do not consider some characteristics of input faces, such as wrinkles or marks, so their output will not contain these details.

## Generation of Facial Cartoons

Their project aim is to enhance attractiveness rather than similarity because their target users are young people. Therefore, their result images are younger looking even if the input images are not.

On the other hand, they still consider results not being good that their future work is to have a larger database for facial features and consider algorithms for deformation.

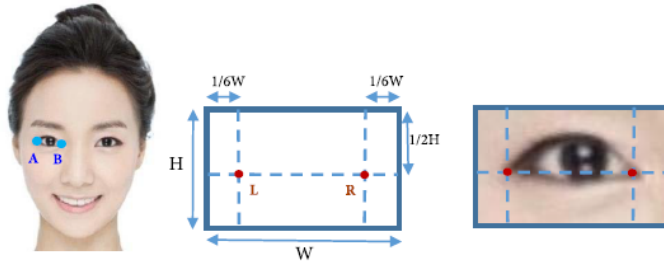


Figure 2. Bounding rectangles calculation using facial landmarks. Reproduced from [14].

There are different ways to determine the sizes of the bounding rectangles. In this project, the bounding rectangles are of a fixed size, 62x100 for an eye for example. Then to locate where they should be, it is using the method shown in figure 2 with the two endpoints of the eye. By doing this, there is no need to resize the extracted features, however, the rectangles cannot be entirely sure that they will bound the feature, therefore, it is very important to choose the correct values.

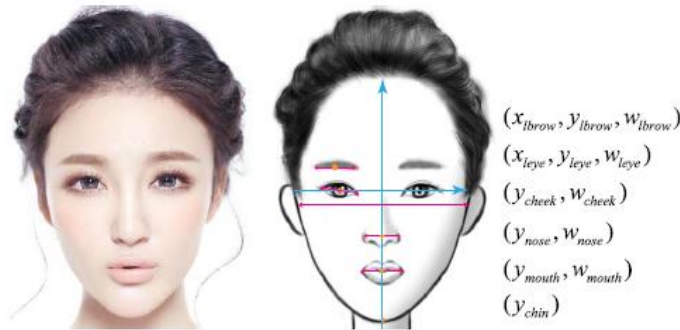


Figure 3. The coordinate system used to represent features locations. The location of the features is represented by a 13-dimension vector, where x and y are the coordinates and w is the width. Reproduced from [14].

The coordinates of different facial features are determined by the defined x and y axis and using w for width. Also, this project uses  $\epsilon$ -SVR method to let the system learn how to adjust the facial composition of a cartoon. So, the variables in figure 3 are the coordinates of all the features after alignments.

## Generation of Facial Cartoons

As mention above, the output images must be aligned, meaning there will not be special movement or expression of features shown.

In terms of hair segmentation, they require their image background to have significant difference to the hair colour. They first acquired the hair region by using “Detection and analysis of hair”, Y.Yacoob and L.S.Davis [13]. Then, they utilize the KNN-matting algorithm [11], an algorithm to separate multiple images layers to extract the hair. Finally, apply smoothing to obtain the hair contour.



Figure 4. Results of hair extraction. Reproduced from [14].



Figure 5. Input images and their corresponding generated facial cartoon with all matching features. Reproduced from [14].

The above input and output images clearly show which feature are matched with which cartoon and show attractive results that meet their aim. On the other hand, as they mention, there are still limitations and some output being unrealistic. Therefore, there are areas where we can learn from and try to improve or even solve.

### 2.1.2 “Face Cartoon Synthesis Based on the Active Appearance Model” [12]

Their method starts with obtaining an average face by learning from a face photo database. The average face with its feature points are fitted onto the input image iteratively in the Active Appearance Model (AAM) by adjusting the parameters. Then it performs patch matching with the training database by searching. The results show the ability to preserve the colour of the image photos but lacking facial details. Also, their images are realistic and recognizable which meets their aim. Moreover, their future work is to do some facial expressions to enhance the humorous effects which is something we will be considering in our project.

Similar to the project in section 2.1.1, it only considers images in a single orientation which requires image preprocessing. Additionally, input images that are very different to all training images are not considered which may be a limitation of their approach.

Their method of hair segmentation is to convert the image into binary level and then extract the largest connected part as the mask for the hair. Then, to find the hair colour, they calculate the gradient of the hair segment (figure 6, image3) and transform the gradient into gray level with inverse ratio. Their approach requires the colour of the face and the hair to be significantly different in order to have clear distinct separations in the binary format.



Figure 6. Process of hair segmentation. Reproduced from [12].

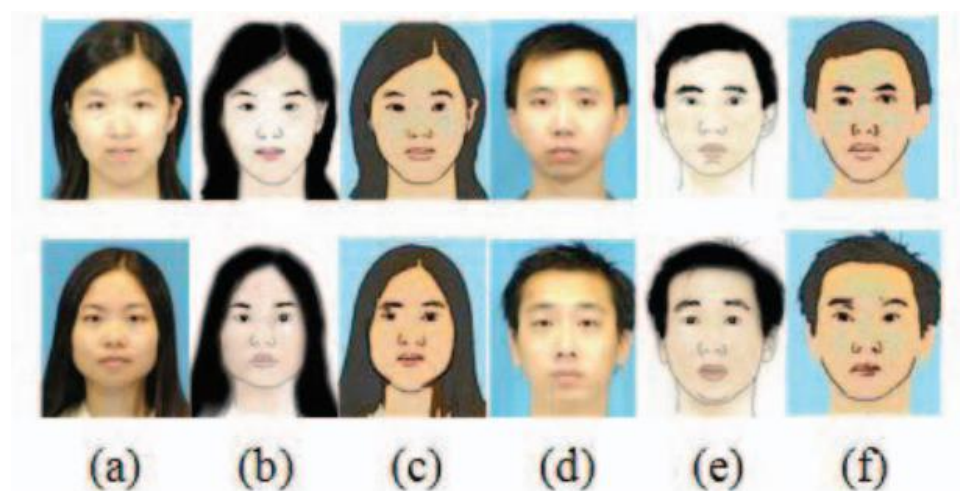


Figure 7. (a) and (d) are input images, (b) and (e) are artist drawing, (c) and (f) are their results. Results reproduced from [12].

### 2.1.3 Example-based Facial Sketch Generation with Non-parametric Sampling [4]

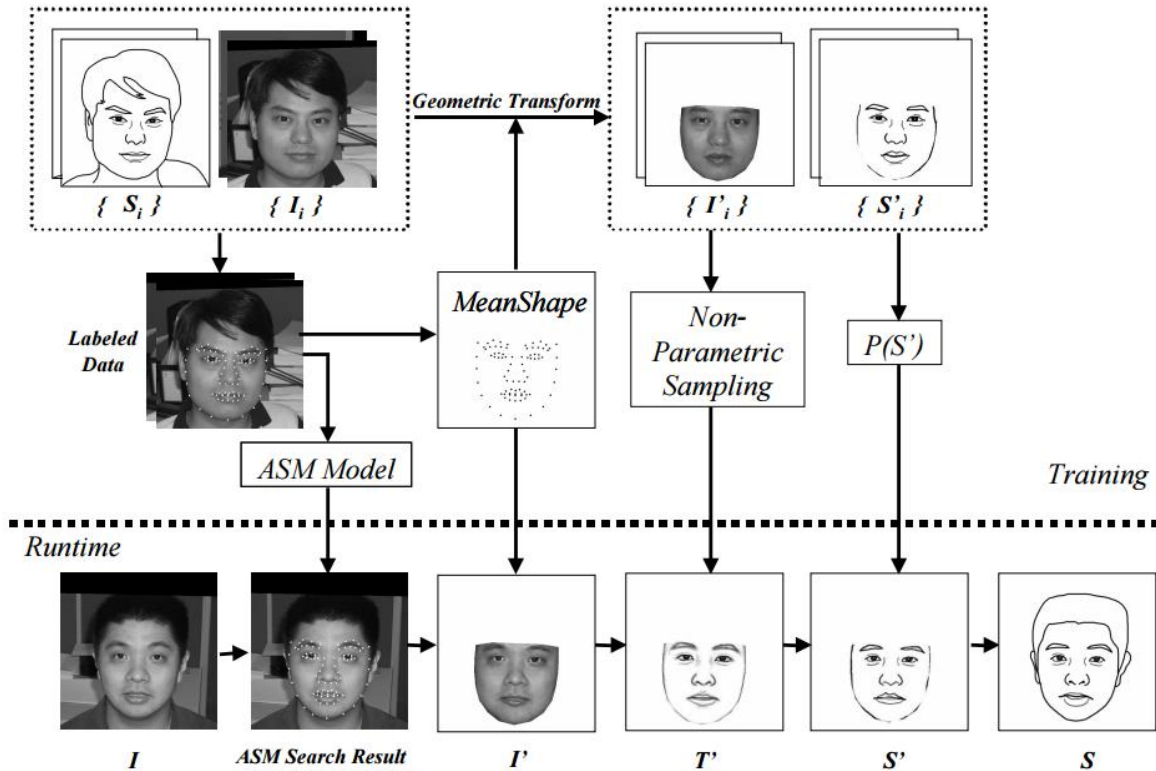


Figure 8. Overall pipeline for "Example-based Facial Sketch Generation with Non-parametric Sampling". Reproduced from [4].

This project aims to generate a sketch based on a facial image regardless of the background. They first have an artist to draw sketches for 162 images and these 162 pairs are used as training data.

Their program begins with training an Active Shape Model (ASM) model to automatically locate the facial features points in training images. Using these points, they define an average shape of the training images. Then a geometric transformation is applied to warp any training images into the average shape.

Then the program moves onto the runtime phase where ASM and geometric transformation is applied to the input. After that, it employs non-parametric sampling to obtain the expected sketch and apply prior probability. Finally, compute final sketch using the inverse geometric transformation to warp the input back to the original shape.

This method is a good way to preserve orientation of the face using geometric transformation to warp images. Also, it is able to show facial expressions and slight rotations to the features.



## Generation of Facial Cartoons

In comparison to our project, they are capable of more expressions but the direction of vision in their output is the same as the face orientation rather than the actual direction. Also, because they are generating sketches, they have less details than our approach which could be less attractive.

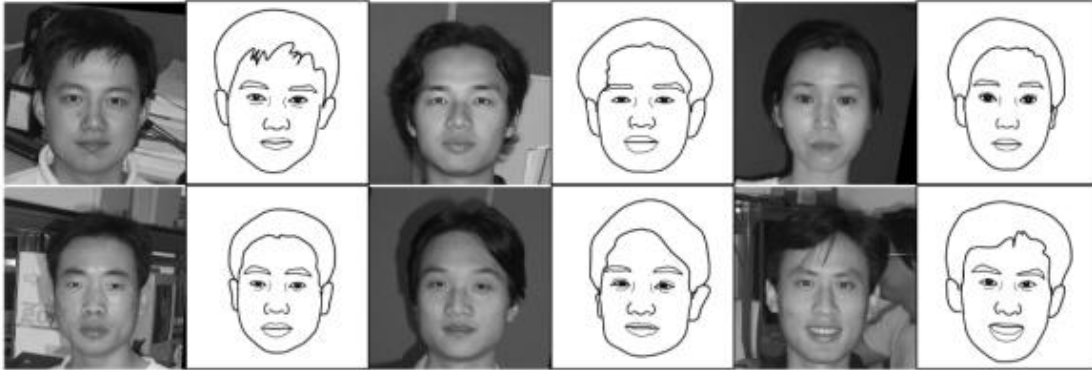


Figure 9. Input images and their corresponding generated facial cartoon. Reproduced from [4].

### 2.1.4 Other related work

There are other projects that use similar methods such as displacing the ASM as it is a flexible and robust method, and finding the best matched patches for different features.

In “A Novel Hierarchical Decomposition Model for Facial Caricature Synthesis” [6], they are using patch sampling and SVM based classification to determine which example the input is the best matched. They have a rendering dictionary for all example features which contain a set of connected nodes to replace the patches on the features. Then the nodes are fitted by a non-periodic B-Spline curve to form a line drawing and further decorated with colour filling and image warping.

In “Cartoon-like Avatar Generation Using Facial Component Matching” [2], they are matching faces as well as the features using a small image and avatar paired library. For the faces, they are using eigen pictures for reducing sensitivity to statistical redundancies from input images. Their input faces are represented as weight vectors and the closest weighted face is selected. For the features, they are using a distance measure called “The Hausdorff distance”. Their matching approach has only one measure whereas we have 6 because we assume that there is no best measure. Also, their avatar features style is in a very different style to ours, which makes it difficult to compare the results. However, their results do show some facial expressions but the style is far from being realistic.

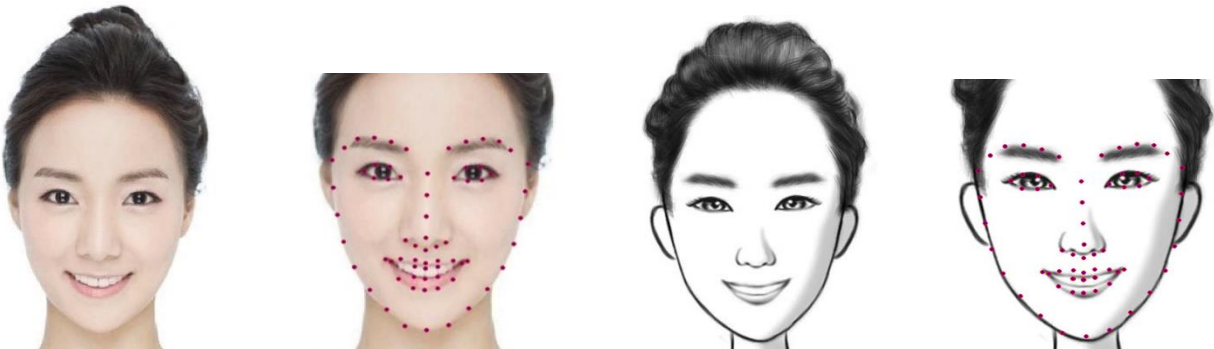
These approaches are similar to the approach in section 2.1.2 and 2.1.3 where ASM is used to locate the facial features, where in our project, we are using “OpenFace” to find the feature



## Generation of Facial Cartoons

landmarks. Also, some projects require large number of training images which we are lacking or having different aims for results images. So, it is difficult for us to use similar methods.

### 2.2 OpenFace Program



*Figure 10. From left to right: Frontal face image as input for OpenFace program. Output image with landmarks. Frontal cartoon image as input for OpenFace program. Output cartoon with landmarks.*

Our interested functionality of the “OpenFace” program is to detect facial features including eyebrows, eyes, nose, mouth and the outline of the input image.

The program first detect faces on an image with a set of pre-trained models from OpenCV. Then the program preprocesses each face to create a normalized and fixed-sized input to a neural network. The neural network is used to represent the face on a low dimensional unit hypersphere which is more efficient to perform classification similarity detection and clustering. Then training is performed on the face representation using a dataset with over 100 million images.

This program has several output choices but the two we are interested in are the .txt file which are the 68 landmark (X,Y) coordinates and the .bmp image file that shows where the landmarks are located on the facial image.

These details are very important to locate where the features are and resulting better features extraction. Additionally, if there are errors in landmarking, we can easily spot the errors and make adjustments.

Through running the 28 examples (56 images) in our image library, there were 2 images that contain major detection errors.

## Generation of Facial Cartoons

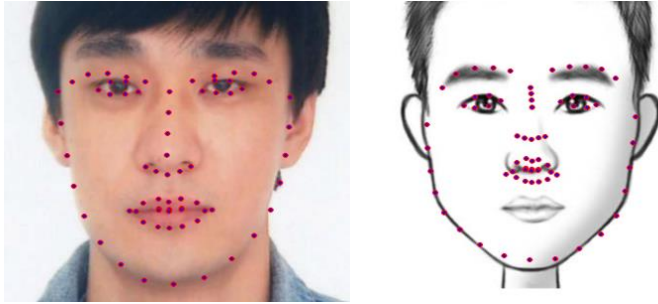


Figure 11. Images showing Open Face detection error.

The run-time of this program is rather long if the input images are high resolution, for example, a 2144x1424 image takes around 2 minutes on my computer.

Despite 3.57% of detections are faulty in locating the correct feature locations, this program on average generate positive results especially on the eyes and the noses where the shapes are captured perfectly. Whereas, some eyebrows have detected as extended lengths and some mouths have faulty results in detecting between the lips. Overall, “Open Face” is still reliable in majority situations which we can mostly rely on.

## 2.3 Coherent-Line-Drawing Program

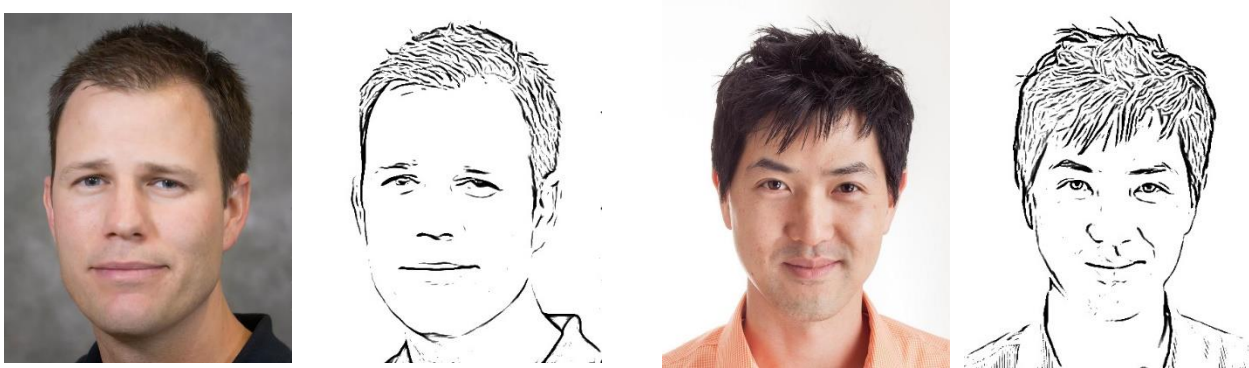


Figure 12. Frontal face image as input for Coherent-Line-Drawing Program and their corresponding output line image.

The main functionality of the Coherent-line-drawing program is to convert an image into a line drawing as shown in figure 12. This program implements a non-photorealistic rendering technique that is an extension to a normal edge detector, which is for detecting highly coherent lines and minimizing noise.

## Generation of Facial Cartoons

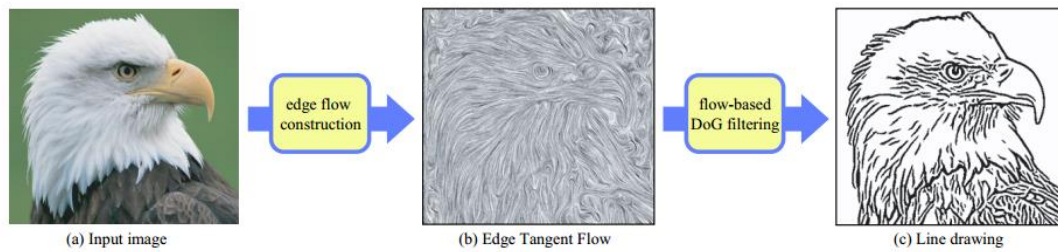


Figure 13. Process overview for Coherent-line-drawing. Reproduced from [5].

The program begins with constructing an edge flow field (Edge Tangent Flow, ETF) from the input image. This is done by applying nonlinear vector smoothing to all pixels in the image, which would generate (b) in figure 13. Then the program applies a flow based “Difference of Gaussians” (FDoG) filter onto the image, where the kernel shape is defined by the local flow recorded in the ETF. Then the program iteratively applies FDoG to improve line coherence.

This is a useful tool in creating cartoon images due to its ability to remove noise but preserve significant line details. Meaning that it would be possible to create a line drawing for each feature and use that to match with the line version of our library which may enable better-matched results.

The overall results of “Coherent-line-drawing” show good line extractions, providing some details to allow matchings to be done.

### 3 SPECIFICATION AND DESIGN

This section details the methods and the algorithms that were used in this project. The aim is to help readers to understand the approach of my project design.

#### 3.1 Pipeline

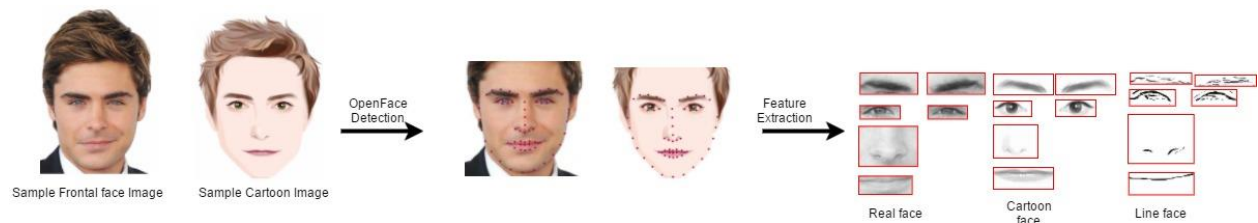


Figure 14. The pipeline of creating features library of this project.

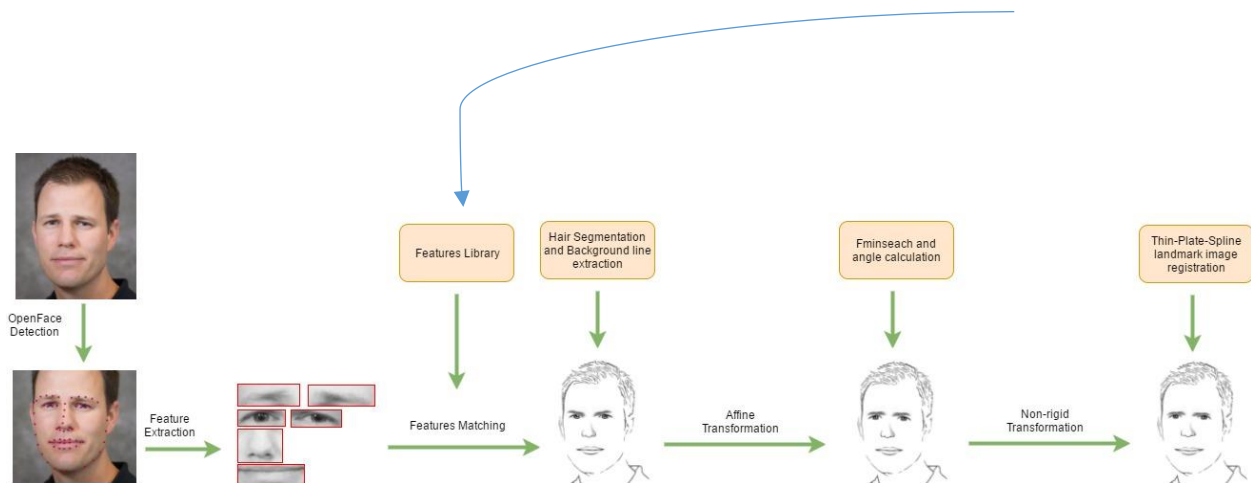


Figure 15. The overall pipeline of this project.

### 3.2 OpenFace system to extract facial features

As described in the background section, OpenFace system provides 68 sets of (x,y) coordinates of the facial components in an input image. With the information given from their GitHub site, I know which set of coordinates belong to which part of the features. This allows me to process this information easily and locate a bounding rectangle on the interested region.

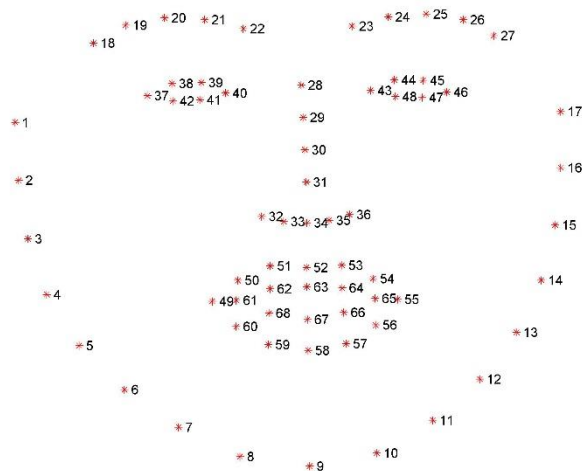


Figure 16. Landmarks numbering system for OpenFace program.

Before any extraction, the images are scaled to gray level. This minimizes the effects of limited correspondences in colour matching and allows a simpler matching method to execute.

Then the next task is to choose the correct sizes of the bounding boxes.

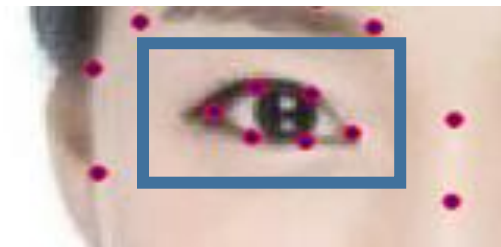


Figure 17. Image showing landmarks on a left eye with an approximate bounding rectangle.

For example:

A left eye is represented by coordinates from 37 to 42. To calculate the bounding rectangle, it is required to find the maximum and minimum of both x and y value of coordinates 37 to 42.

Using these maximum and minimum values, extend these values to have a clearer view of the facial components.

The original idea for choosing the extension values was the size of the image. However, since images with the same size can have different sizes of faces, meaning that this will cause inconsistencies for extracting the features. So, the other method is by the scale of the size of the face which are the (1:17) sets of coordinates. This will provide a better scaling standard regardless of the image resolution.

### 3.3 Hair segmentation using histogram analysis

Hair is an important feature to identify a person. As mentioned in section 2.1.1, their program first detects the hair region from [13] and separate image layers using the KNN-matting algorithm [11] to obtain the hair contour. Finally, apply some smoothing to generate example output in figure 4. In section 2.1.2, they first convert the image into binary level and the largest connected region will be used as the mask for the hair region. Then they use the inverse ratio of the gradient value to find the colour of the hair. Results shown in figure 6. The two approaches require the colour of the face and the hair to be significantly different in order to have clear distinct separations in the KNN-matting algorithm [11] and in the binary format.

The idea behind hair segmentation for this project is based on the principle taken from the two literatures that the hair colour is different to the skin colour and the rest of the image.

Therefore, it is possible to perform a histogram analysis on the pixel intensities for an input image to find which pixels are within the range of the hair intensities.



*Figure 18. Image showing the upper part of an input frontal face image with an approximate region for hair histogram analysis.*

Using the OpenFace coordinates of the eyebrows (18:27). It is possible to find the hair region on the upper part of the head and use that to perform a histogram analysis. Using this region should allow more accurate results due to less influence by other features and the background intensities.

Using the results of the histogram analysis, it is then to find the intensity with the highest frequency in the range of 0 to 115 in gray scale, assuming the most frequent hair colour is in this range of intensities which is a darker colour.

Then finding the minimum intensity value to calculate the maximum intensity allowed for that image by subtracting the most frequent intensity(value) by the minimum value, (range):

$\text{range} = \text{abs}(\text{value} - \text{minValue})$

$\text{maximum} = \text{value} + \text{Range}$

For pixels in other intensity range will be set as 255 in gray scale so that there will be a clear hair segment. To clear the regions of the features and within the outline of the face, their corresponding bounding rectangle area will be set as 255 as well.

As mentioned above, both literatures and our method of hair segmentation require distinct difference in colour. So, the limitations expected in our method are that:

- Poor segmentations if the hair colour is similar to the colour of the forehead.
- If the person's head on the image is far away from the top, there will be more unwanted region.
- If there are multiple hair colours, our program will not be able to segment the wanted hair.

### 3.4 Coherent Line Drawing to generate line images

Facial images can sometimes be more meaningful by the background of the image. Using this program is a good way to outline any lines to enhance the image.

It is also capable of finding lines of the faces, such as wrinkles or lumps. Which can result in increasing realism for the output images.

The most important use of this program is to perform a line extraction for all features in the library, and this will be used in one of the three types of matching which is a 'line to line' comparison.

## 3.5 Matching algorithms

Matchings will be performed in three different states of comparison.

1. Both images in original gray level
2. Both images in extracted line state
3. One in extracted line state and one in cartoon gray level

This is done to test which state of matching provides the user desired result. Which means there will be a user study to examine the output images generated from them.

There are many methods to find the similarity between two images, in this project, I am using some MATLAB built-in functions as well as coding some existing similarity measures. The idea is that given an input facial image, all the features are saved and each will take turn to be compared with all the corresponding feature images in the library.

Using the similarity measures on the features image, each measure will cast a vote on the most similar library image. Finally, the image for that feature that has the highest vote will be considered as the result for that part.

The similarity measures include:

1. Mean Squared Error (MSE)
2. Peak signal-to-noise ratio (PSNR)
3. Structural Similarity Index (SSIM) for measuring image quality
4. Intensity mismatch
5. Cosine similarity
6. Absolute distance measure

### 3.5.1 Mean Squared Error

Mean squared error finds the cumulative square error between two images.

$$MSE = \frac{\sum_{m,n} [I_1(m,n) - I_2(m,n)]^2}{M * N}$$

M: number of rows in image

N: number of columns in image

I: image



### 3.5.2 Peak signal-to-noise ratio

This is used to find the ratio between the power of signal and the power of noise that affects the image.

$$PSNR = 10 \log_{10} \left( \frac{R^2}{MSE} \right)$$

R: determined by data type, 255 for 8-bit unsigned integer, 1 for double-precision floating-point

### 3.5.3 Structural Similarity Index (SSIM) for measuring image quality

SSIM is a more sophisticated similarity measure due to the range of aspects it covers. The aspects it looks for are luminance, contrast and structure, which are calculated as the following:

$$l(x, y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1},$$

$$c(x, y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2},$$

$$s(x, y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$$

After calculating each aspect, the formula for finding the index is:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

$\mu_x, \mu_y$ : local means for image x and y.

$\sigma_x, \sigma_y$ : standard deviations for image x and y.

$\sigma_{xy}$  : cross-covariance for image x and y.

Using this measure may be useful to use a lot of details within the image, however, due to low complexity in images in different states such as cartoons and line drawings, there might be less significance to use a more complex measure. So, this should be tested to see any major improvement by using this method.

### 3.5.4 Intensity mismatch

This measure is simply a comparison between the intensities across the images and count the number of pixels that have different intensity values.

### 3.5.5 Cosine similarity

This measure is used to find the cosine angle value between the intensities in two images.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}},$$

### 3.5.6 Absolute distance measure

This measure is used to find the intensity difference across all pixels of two images. Using this method can help us to find the significance between feature displacements and overall intensity differences which can determine true matches.

### 3.5.7 Potential problem with the similarity measures

The problem with using MSE, PSNR is that it is comparing images pixel by pixel based on the intensity value. For example, consider 3 images a, b and c, each image is two pixels in size:

a: pixel 1 is 100 and pixel 2 is 255

b: pixel 1 is 150 and pixel 2 is 150

c: pixel 1 is 100 and pixel 2 is 100

If we were to match image a and b to c, the squared error would be  $(255-100)^2 = 24025$  for (a to c), and  $2 \times (150-100)^2 = 5000$  for (b to c). However, there is a matched pixel for (a to c) whereas no matched pixel for (b to c) but these results show significant difference in favor to (b to c). Therefore, it is showing the limitation of the fact that an image can have different intensities in all pixels compare to the matching image but still voted as the best matched. On the other hand, similarity measures including intensity mismatch, cosine similarity and absolute distance measure will not have this problem.

The problem with intensity mismatch is complete opposite to MSE, where an image can have more matching pixels but with many pixels that have large intensity difference. Measures such as cosine similarity and absolute distance do not emphasize large differences, so even some pixels are largely different, they will have less impact towards the results.

The problem with cosine similarity is that it measures the relative difference, using the same example in the beginning of this section, (a to c) will have a value of 23.58 whereas (b to c) is 0, making (b to c) as the better match. Therefore, in cases as such, we could have a match that is similar in shape but totally different in colour, which will not be considered as realistic.

Before matching, feature images are resized into the corresponding feature size due to the requirement by the similarity measures. 11x44 for eyebrows, 15x33 for eyes, 38x46 for nose and 18x45 mouth.

After choosing the features, the features are first resized to the corresponding feature size of the input image then they are placed in the calculated coordinates which are determined by where the bound rectangles were extracted. So, resulting image will have some degree of scaling and translation. This can help identifying the better state of matching.

### 3.6 Open mouth recognition

Some images have big smiles which cause gaps between the upper and lower lips. To avoid bad matching for an open mouth. There will be an open mouth recognition using the Y(62:64,66:68) sets of coordinates. If the difference between the maximum of Y(62:64) and the minimum of Y(66:68) is larger than the face scale value, then the mouth is considered to be open and will only be matched with the image library with open mouth.

### 3.7 'bwareaopen' function to remove small objects

'bwareaopen' is a MATLAB built-in function that removes small objects in a binary image.

The basic steps are

1. Determine the connected components:  
`CC = bwconncomp(BW, conn);`
2. Compute the area of each component:  
`S = regionprops(CC, 'Area');`
3. Remove small objects:
  4. `L = labelmatrix(CC);`  
`BW2 = ismember(L, find([S.Area] >= P));`

For the first step, the connectivity can be specified by users, in our case, 4-way or 8-way connectivity in two-dimensional images. Then for step two and three, it finds the pixel size of the objects and removes them by user size selection.

Therefore, this function can be used to set a high area value to only keep the large and significant facial lines.

### 3.8 Rigid transformation

The results generated from the matching algorithm only provides some degree of scaling and transformation but not rotation. Therefore, the next step is to apply rotation for each feature. To be able to find the most suitable rotation value, there are three methods to do so.

#### 3.8.1 Pixel image registration using `fminsearch`<sup>1</sup>

The idea of using `fminsearch` is to use the MATLAB function `fminunc` from the optimization toolbox to find the parameters that minimize the pixels' differences between two images:

```
XF =fminunc (@(X) SSD(X, img1, img2), start, opt);
```

For this function, what we have to do is to write a “SSD” function that finds and returns the error difference between two images. Then give an initial “start” estimate to the transformation parameters we expect them to be. In our program, we assume the rotation required is not large, so the initial value will be zero. Lastly, give some optimization options “opt” for other purposes such as viewing run-time details. For this project, we will use the default settings for “opt” to keep consistencies.

A potential problem for this method is that we need to provide a good initial estimate for the `fminunc` function to find the optimized output. However, our program does not have a way to generate a good estimate. Additionally, this function may sometimes search in a wrong direction and remains around the incorrect value region. Therefore, we will require a grid search to allow multiple initial starting values.

---

<sup>1</sup> <http://cmp.felk.cvut.cz/~garcia/Teaching/Medical%20Imaging/Excercise1b.htm>

### 3.8.2 Pixel image registration using fminsearch grid search

Using grid search allows us to use more initial values, then combining all searches, we can find the result with the lowest SSD error. The values that will be used are [-12,-9,-6,-3,0,3,6,9,12]. These values are only estimations, which could be changed later during implementation when images require larger rotations.

The potential problem for using grid search is that the time efficiency is low due to the number of searches need to be done to find the optimized value, which is determined by the size of the grid, for example, 9 in our case.

### 3.8.3 Rotation using OpenFace coordinates

The last and simplest method is to use the openFace coordinates to determine the rotation angles.

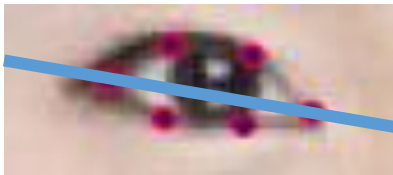


Figure 19. Image showing orientation of a left eye.

Using the coordinates sets 37 and 40, we can find the two side points of the left eye. To find the angle between these two points, we simply use trigonometry:

$$\tan(\theta) = \frac{Y2 - Y1}{X2 - X1}$$

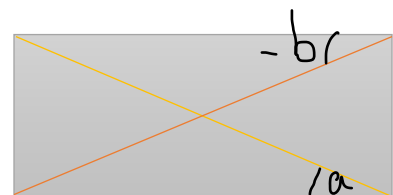
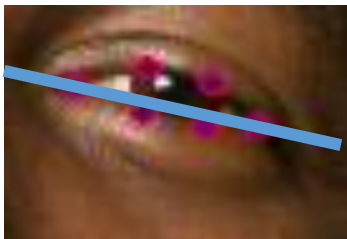


Figure 20. From left to right: Part A, image showing wanted right eye orientation. Part B, image showing current right eye orientation. Part C, image showing how orientations are found, the yellow line represents when Y2 is larger than Y1, the orange line represents when Y1 is larger than Y2. Left top corner (X1,Y1), right top corner (X2,Y1), Left bottom corner (X1,Y2), right bottom corner (X2,Y2).

There has been a consideration for how the angles should be calculated. For instance, if figure 20-part B were to rotate to figure 20-part A:

- 1) From figure 20-part A, the angle found will be (a).
- 2) From figure 20-part B, the angle found will be (-b).
- 3) From figure 20-part C, using correspondence triangle theory, the two labelled angles are the same but one is negative and one is positive.
- 4) Then the rotation angle is simply  $(a) - (-b)$ .

For a normal case where the orientations of two right eyes are the same, meaning Y1 is larger than Y2 in both images. Then the rotation angle is  $(-a) - (-b)$ . Otherwise,  $(a) - (b)$ . Therefore, this means we can use the same formula to calculate angles for all features.

Using this method compare with the two other options is that the run time is shorter as only trigonometric calculations are done. However, this method relies on the correctness of the OpenFace landmarks. Therefore, it requires testing to see which option is most suited for our project.

After finding the rotation angles for all the features, only affine transformations for rotation will be applied. There will an analysis to see which method generate the best results and with consideration of time complexity.

### 3.9 Non-rigid image registration using thin-plate-spline

Non-rigid image registration is an image processing technique that compares two images, then one is to be changed into the shape of the other. The method we are using, thin-plate-spline(TPS) is a technique for data interpolation and smoothing. It includes a mapping algorithm which has the option to compare landmarks instead of using pixel intensities and a warping algorithm that uses interpolation and smoothing to create a registered image.

In this project, we will be using the mapping algorithm to find the differences between the wanted shape landmarks and the current shape landmarks, hence, finding the vector for transformation. Then we will use the vector to map all points on the current shape on the wanted image plane. Finally, using the warping algorithm, pixels are interpolated to form the shape of the wanted feature.

## Generation of Facial Cartoons

This is the last stage of the generation of cartoon images. However, since we are using code from “Fitzgerald Archibald”<sup>2</sup> on “uk.mathworks.com”, the code still require examination and extensive testing to make sure it is providing positive results.

---

<sup>2</sup> Copyright (c) 2009, Fitzgerald Archibald

## 4 IMPLEMENTATION AND TESTING

---

This project is written in MATLAB and implements two C++ programs, OpenFace and Coherent-Line-Drawing. MATLAB provides a lot of built-in functions that allows us to implement and focus on reaching objectives, such as 'bwareaopen', 'fminunc' and some matching algorithms used in this project.

The system environment of the computer:

Processor: Intel® Core™ i5-4210U CPU @ 1.70GHz 2.4GHz

RAM: 8.00 GB

System type: 64-bit Operating System, x64-based processor

### 4.1 Choosing the bounding rectangles face scale value

In the design stage, I considered using the size of the face as a scale to determine the extension to a bounding rectangle. Through testing, I have created two variables "faceX" and "faceY" where "faceX" is the horizontal length of the face divided by 20 and "faceY" is the vertical length divide by 20. These values were chosen because the majority of the features uses these scaling values as extensions.



Figure 21. Images showing extracted features

The following are the extension values used for each feature:

**Eyebrows:** `I (ymin-faceY:ymax+faceY,xmin:xmax+faceX) ;`



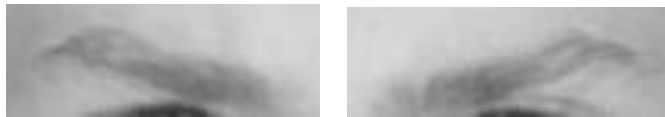
## Generation of Facial Cartoons

**Eyes:** `I (ymin-faceY2:ymax+faceY2,xmin-faceX:xmax+faceX) ;`

**Mouth:** `I (ymin:ymax+faceY2,xmin:xmax) ;`

**Nose:** `I (ymin:ymax+faceY2,xmin-2*faceX:xmax+2*faceX) ;`

The advantage of using these values is that the features are clearly visible within the bounding rectangles. However, the problem is the extraction of the eyebrow and the eye. This is because the eyebrows tend to curve down very close to the y level of the eye, meaning that the extraction may sometimes capture part of the eyes.



*Figure 22. Extraction of eyebrows capturing edges of the eyes.*

In figure 22, we can see that it is required to reach to that level to capture the entire eyebrow. Therefore, I have noted this problem and will consider it in the later stages of the project. On the other hand, the cartoon version of these features has been modified to remove these details, so, this will not affect the output images.

## 4.2 Coherent-line-drawing on features

As mentioned in section 3.4, a line drawing version of all the features in the library will be created for the matching algorithm to run. For this part of the program, we will be using the unmodified code from “Coherent-line-drawing” program which will give us the original line drawings. We have a modified setting and the purpose of it will be explained in section 4.3.

## 4.3 Hair segmentation using colour histogram analysis

Hair segmentation has been done two ways. The first method is to convert the image into gray level in the beginning then perform the histogram analysis and the second method is to

## Generation of Facial Cartoons

perform histogram analysis on the RGB channels separately and convert the image into gray level in the end. Results show that converting the image into gray level in the beginning gave a better segmentation. The following figure shows a result of hair segmentation:



*Figure 23. Results of hair segmentation.*

This method was further tested on different images and shown good output. However, the major problem is that this method relies on the clear difference in colour between the hair and the skin. Therefore, when I run more images that may cause problems, the results showed bad segmentations.



*Figure 24. Bad results for hair segmentation.*

## Generation of Facial Cartoons

The reason for bad segmentation can be caused by many conditions in the images. For example:

1. Natural colour closeness such as:
  - a. blonde hair with bright skin
  - b. dark hair with dark skin
  - c. Similar hair and background colour
2. Person with less hair or bald
3. Lightings onto the image

It is shown that this method has is unreliable and has too many constraints. So, while trying to find a better solution, the background line images created using Coherent-Line-Drawing showed a different style of hair but it provides good amount of details as well as reliability. Therefore, it is used as a replacement for hair segmentation.

It is important to note that the “Coherent-line-drawing” code used in this part of the project has been modified. The modification was onto the two sigma values where we have lower the ratio from  $0.4/3$  (1:7.5) to  $1/4$  (1:4). Also, it is important to resize input images that are larger than 800X800 so the facial details can be shown using the modified settings. The current method is to resize in 0.5 ratio but before that, we will check if the image is over the maximum resolution 1500X2200 and if so, they are resized to the maximum resolution in their aspect ratio.

This gave us line drawings that have better solid lines with reduced noise and line segments on the face, which is a more ideal solution than the original settings for generating the hair and the facial outline as a more solid style.

The only problem with this method is that it is generating lines in a more approximate or an overacting style which is giving less details for the features by giving larger and thicker solid lines. Therefore, this explains why we are not using the modified settings for section 4.2.



*Figure 25. Left to right: Line drawing image generated using Coherent-line-drawing. Line drawing after colour modification and features removal.*

In addition to the line drawing, features regions are also set to white to form from left image to right image in figure 25. To make the line drawings more realistic compare with the actual hair colour, the original idea of histogram analysis is used to find the most frequent intensity of the input image and then is used for all the lines in the line drawing, which resulted in the right image in figure 25.

### 4.4 'bwareaopen'

The use of line drawings may result in excessive lines on the face, which some we would like to keep such as the significant ones where other lines are better to remove.



Figure 26. Left to right: Line drawing image generated using Coherent-line-drawing. Line drawing created by applying 'bwareaopen' function.

In the left image of figure 26, it shows excessive dots around the face which can be caused by facial hair, spots or even lightings. If an output image contains these details, they would be looked as noises rather than features that would enhance the output. So, this method is implemented with input parameters 80 for the threshold for the number of connected pixels required and an 8-way connectivity option.

As mentioned in section 3.7, this method only works on binary images, also, it only looks for white objects to be removed. Therefore, I had to first convert the line drawing (feature regions removed) to binary format which is simple to choose the thresholding value as all pixels that are not white will be set to black. Then I had to create an inverse of that image so that the lines are white with black background for the function to work. After that, the function has been run and the resulting image will be reversed back so that the background is white and convert back the gray level scale. Finally, this image will be used as a mask such that if the pixel value in the mask is not white, then the corresponding pixel in final output image will be its original value.

One of the example output generated is the right image in figure 26, which shows result with significant lines and removed unwanted dots despite some lines around the mouth which should be covered when placing the cartoon stylized mouth into the drawing.

## 4.5 Open mouth recognition

As mention in the design stage, the decision for an open mouth is also by the face scale value. In this case, it uses "faceY" which is the vertical face length divide by 20. The aim of this

function is to mark mouths that are largely opened as “open mouth”, while the rest are regarded as “closed mouth”.

So, this function has been tested using three types of face images and 54 examples in total:

1. Close mouth
2. Slightly open mouth
3. Open mouth

The objective of this test is to look at the results and determine whether the face scale value used is correct by user examination. Therefore, each image will be first classified by user. Also, we should consider if there is an OpenFace error to cause incorrect classification.

The testing results show that there were only 4 examples classified differently compared to the user classification, and all those classified differently were because of OpenFace landmarking errors. So, the results prove that this function can provide an “open mouth” recognition and will be implemented in this project.

## 4.6 Matching

As mentioned in section 3.5, feature matchings are done using different similarity measures and in 3 different states. For each feature, there will be 6 votes in the end and the most voted feature image will be used.

During implementation, I found out that there were matchings with the same similarities, such as examples 2,9 and 16 have the same MSE. However, because my implementation starts from example 1 to 28, meaning example 2 is always selected in this case rather than 9 or 16. So, my solution was to include all examples that have the same similarities for the current matching, which provide a fair chance for all examples. In the end, there could be possibly more votes due to this solution.

Another problem found during implementation was that the votes from PSNR and MSE will always be the same. This is due to the fact to PSNR is calculated using the MSE value, meaning I should have noticed it during the design stage and should be something to improve from.

The problem is that this will give an advantage to the example voted by these measures, and the solve this problem, I have removed MSE from the similarity measures to improve fairness.

### 4.6.1 User study on similarity measure

As mentioned in section 3.5, SSIM seems to be a sophisticated measure compare to others which are more simple. So, to prove that we should only use SSIM rather than a voting system, I have constructed a user study on 54 example output images to see which method provides user's preferred results.



Figure 27. Left to right: Input image. SSIM result. Voting system result.

The study has been carried out to compare each feature, the feature that is preferred will be awarded a point and if the features are the same, they will both gain a point.

Results: <sup>3</sup>

	Voting System	SSIM	Number of Same example
<b>Eyebrows</b>	46 – 51.1%	44 – 48.9%	36
<b>Eyes</b>	43 – 58.9%	30 – 41.1%	19
<b>Nose</b>	44 – 54.3%	37 – 45.7%	27
<b>Mouth</b>	48 – 52.7%	43 – 47.3%	37

Figure 28. Table show results of user study on similarity measures.

Even though there were only 54 examples, the results from figure 28 show that both methods would generate 55.09% of the same features from the same example. Meaning there are high correspondences in both methods.

On the other hand, if we remove the number of same output to each measure, the results will be higher significant differences. Eyebrows: 10 (55.6%) to 8 (44.4%), eyes: 24 (68.6%) to 11 (31.4%), nose: 17 (63.0%) to 10 (37.0%) and mouth: 11 (64.7%) to 6 (35.3%) in favor of voting system. Results show that when the feature examples are different, the one voted by the voting system method are preferred in a high significant difference. Therefore, the voting system method will be used in this project.

<sup>3</sup> See appendix A for full results

### 4.6.2 User study on feature state matching

The matchings are also done in different states, including “line to cartoon”, “line to line” and “photo” to “photo”. To find the best matching state for each feature, a user study had been carried out to find what output a user would prefer.

The user study is separated into 5 parts and each part will be analyzed independently. The parts include analysis on the whole image, eyebrows, eyes, nose and mouth. The number of input images is 9 faces, and there are 6 images generated for each state for each input.

The user was first to compare two result sets generated from states “line to cartoon” and “line to line” then “line to line” and “photo to photo” then “line to cartoon” and “photo to photo”. Images are compared two at a time and all images from the 2 sets were compared with all images with the other set, so all user had to do is to give a score to the preferred image according to the part being analyzed. If the images look the same, then a score will be awarded, however, if both images do not look similar to the input image, then no score will be awarded. If there is no repetition or bad matching, the total score should be 675 for one analysis part.

The results from this user study:<sup>4</sup>

	line2cartoon	line2line	photo2photo	Total
<b>Whole image</b>	111	334	231	676
<b>Eyebrows</b>	175	264	289	728
<b>Eyes</b>	155	299	280	734
<b>Nose</b>	194	285	247	726
<b>Mouth</b>	148	317	258	723
<b>Total</b>	783	1499	1305	3587
<b>Percentages</b>	21.83	41.79	36.38	100

Figure 29. Table showing results from user study on feature state matching

The overall percentages show that there is a clear preference for “line to line” states of matching which is suggesting this should be the method used. However, analysis for eyebrows shows that there is a higher score on “photo to photo” matching, this might be due to extraction problem described in section 4.1. As a result of that, we will be using “photo to photo” matching for the eyebrows and all other features will be using “line to line” matching.

<sup>4</sup> See appendix B for full results.



Additionally, the results from the “total” column show that the score for each matching states are exceeding the score 675, meaning there are less bad matches compare to good and same matchings. Which is showing good matching results generated by the matching algorithms.

### 4.7 Choice of method in rigid transformation

In the design stage, there were 3 methods to choose from to determine the correct rotation value. So, to find out the most suitable method, I have implemented all three and compared the results to see which method gave the most reliable results.

#### 4.7.1 Pixel image registration using fminsearch

I have followed the method described in the design stage and wrote a working program. However, the output was inconsistent in the fact that out of the 6 components, there will always be at least 1 to 2 components that gave wrong rotation values and some values had very large differences compare with the desired values. The major problem as mentioned in the design section 3.8.1, the searching direction was wrong and remained around the incorrect value region, which was shown in the command window in MATLAB that the output fminsearch value continues to be the same and terminated because the program exceeded the maximum number of searches. Therefore, the assumption made was correct and there was the need to use a grid search to prevent this from happening.

#### 4.7.2 Grid search

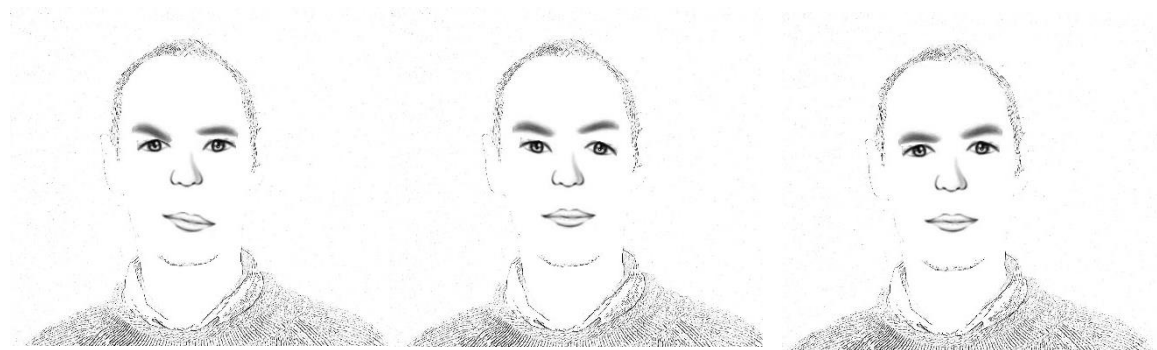
Using the method described in the design stage, the output was also inconsistent but was better than the previous method. There will be sometimes one or two components giving the wrong rotation values but in small errors which are a significant improvement in comparison.

However, there is no guarantee that new input will give small errors as well because this method is reliant on local searches. Also, there are still errors that can affect the appearance and the realism of the output. Moreover, the run-time for the grid search is much longer due to the number of searches needed to be done. Therefore, we should consider another approach that can provide consistent results.

### 4.7.3 Affine transformation using OpenFace coordinates

As mentioned, this is a simple approach that relies on the correctness of the OpenFace coordinates. However, since our program is reliant on the OpenFace program to generate correct coordinates in the first place when extracting the features, they will be assumed to be correct when reaching this stage.

Following the method describe in the design, the images generated were mostly in the correct orientations with very low error percentages and the errors were significantly low and within the 1 to 3-degree differences. The major reason for the errors is that the design of the features in the input image is very different compared to the cartoon features. So, OpenFace may locate landmarks differently according to the styles given.



*Figure 30. Left to right: Results from pixel image registration using fminsearch. Results from grid search. Results from affine transformation using OpenFace coordinates.*

From figure 30, we can clearly see that using OpenFace coordinates provides better results, also, it is the most reliable and cost-efficient method to find the rotation value for all features. Therefore, it is chosen to be implemented in the program.

## 4.8 Implementing non-rigid transformation

As mentioned in the design section 3.9, I will be using some existing code developed by “Fitzgerald Archibald” that in description, provides the functionality for non-rigid image registration. The aspect considered was whether the code would generate results we desire or

expected to be despite some positive comments. So, what I should do is to implement the code into my program and have some images tested to see whether the code makes the features shape more realistic towards the original input images.

This method works by first getting the landmarks for both the current shape and the wanted shape. Using these landmarks, it is able to compute the thin-plate spline mapping algebra. Then, this algebra will be used to find the mapping for all the points in the current image plane to the wanted image plane using the radial basis function. After that, this method warp the current image into the wanted image using the mapping. Finally, due to possibly of unfilled areas given by the warping, it uses the median of the nearest neighbor as the interpolation method to fill in the gaps.

### 4.8.1 Finding the coordinates of the landmarks

As features are extracted into individual images, we have to recalculate the coordinates for them using the relation between their landmarks coordinates in the original image and the extracted location.

For example, a nose is extracted:

```
nose = I(ymin:ymax+faceY2,xmin-2*faceX:xmax+2*faceX);
```

and one of the landmark coordinates is (10,10). Then this landmark will be (10-ymin,10-xmin-2\*faceX) on the feature image.

### 4.8.2 Relocating features

As features are reshaped, how they are placed on the facial background will be different due to varying in size or displacements within the feature image. The method to fix this problem is to find the first and last non-white pixel in both cartoon images (before and after reshaped) in both horizontal and vertical directions. This is because the background is white, so we can track how much the features have been displaced. After finding the displacement values, I have recalculated the balance across 4 directions and found the most suitable location to place the features.

### 4.8.3 Testing the implementation

After first round of testing, the output images were giving very bad shapes that have no relations to the originally input images. So, my response to this was to check if my landmark extraction method was correct. In which, there was one coding error for the mouth there was no clue why the other features are in unwanted shapes.

Therefore, I have done the second round of testing, the shapes of the mouths improved to better looking, however, the sizes for all features are still not correct. So, I looked further into the code and I realised the x and y coordinates for the landmarks are reversed.

Then I attempted the third round of testing and finally, the results are in very good shapes and sizes. As a result of that, non-rigid transformation does provide more realistic features.

To further enhance the features, I have exaggerated the difference between landmarks on the mouth, which provides a larger smile. The results will be compared to see which method is better.

## 4.9 Other implementations

### 4.9.1 Perform transformations on bad examples

Through our implementation and testing, it is shown that both rigid and non-rigid transformation have a large impact on the appearances on the features. So, it was pointed out during the discussion of this project that it would be interested to see the significance of the transformation and the need of matching algorithms to generating the final images.

After considering this aspect, I have generated results from random feature examples that are not voted by the voting system as final result. The aim is to see if the transformations are enough to provide good results despite which example was chosen. Also, if results are good, then we should consider if the matching algorithms are still required in this project.

#### 4.9.2 Perform feature warping before matching

The second approach suggested due to positive results from transformation was to perform transformation to all features before matching. By doing this, we can possibly be able to find the best matching features due to the order of this method. The aim is to see if this can be a better replacement to the current pipeline of the project.

## 5 RESULTS AND EVALUATION

---

### 5.1 Evaluation Approach

The evaluation approach begins with a brief description of what the current section is aiming to achieve. Then with the images presented and chosen by random sampling, there will be description on how well have the results reach the aim. Then there will be evaluation on whether they show promising results that reach the aim and whether they will be included in our program.

### 5.2 Hair Segmentation



(a)

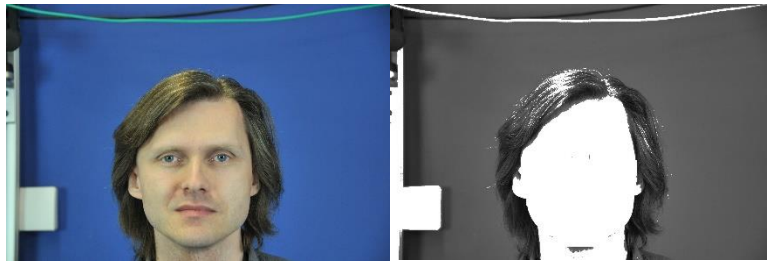
(b)



(c)

(d)

## Generation of Facial Cartoons



(e)

Figure 31. Images showing input images and their corresponding hair segmentation

This section is to evaluate whether the hair segmentation method described in section 4.3 is good to use in our program because as mentioned in section 4.3, there are many factors affecting successfulness of hair segmentation. The results show:

Image (b) is poorly segmented because there is less hair in the region of detection, so, the hair colour histogram analysis determined the skin colour as the hair colour. Images (a) and (c) had problems with the outline of the face when we cannot distinguish between the facial colour due to lightings and the hair colour but also caused by my coding approach to clear areas within the outline of the face. There could be further approaches to solve this problem, however, the fact that there are much more problems and inconsistencies, also, this method will only work when the skin and hair colour are dissimilar, it was better to use the line drawing method.

Images (d) and (e) show acceptable results but (d) still give some white boxes region and (e) is one of the examples that showed good results. Therefore, given the low percentage of successful and clear hair segmentation, this method has not been chosen for this project.

## 5.3 Hair and Facial Lines



## Generation of Facial Cartoons

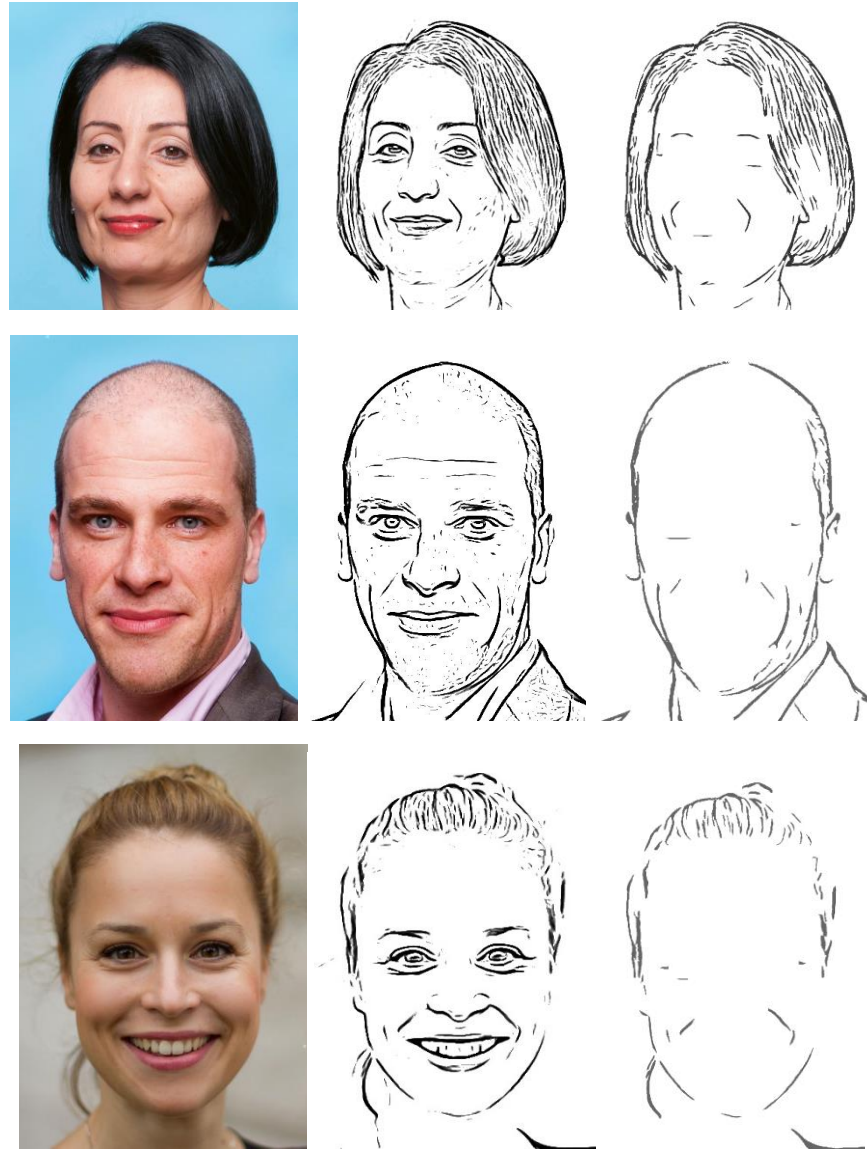


Figure 32. Left to right: Input image. Extracted line drawing. Final line drawing with facial lines.

The section is to evaluate the background line images created using Coherent-Line-Drawing and to examine if it is the option to replace hair segmentation, also, to determine if the significant facial lines are preserved to enhance the cartoon images.

Results from the second column show that the Coherent-line-drawing program is reliable in extracting all line segments from the input images. This is important for the images to have a cartoon hair style as well as clear facial outlines.

Results from the third column is the final line drawing that will be used in the final cartoon output. In terms of hair, the results show clear outline of hair with few missing details. Overall, it shows good standards in representing the hair.

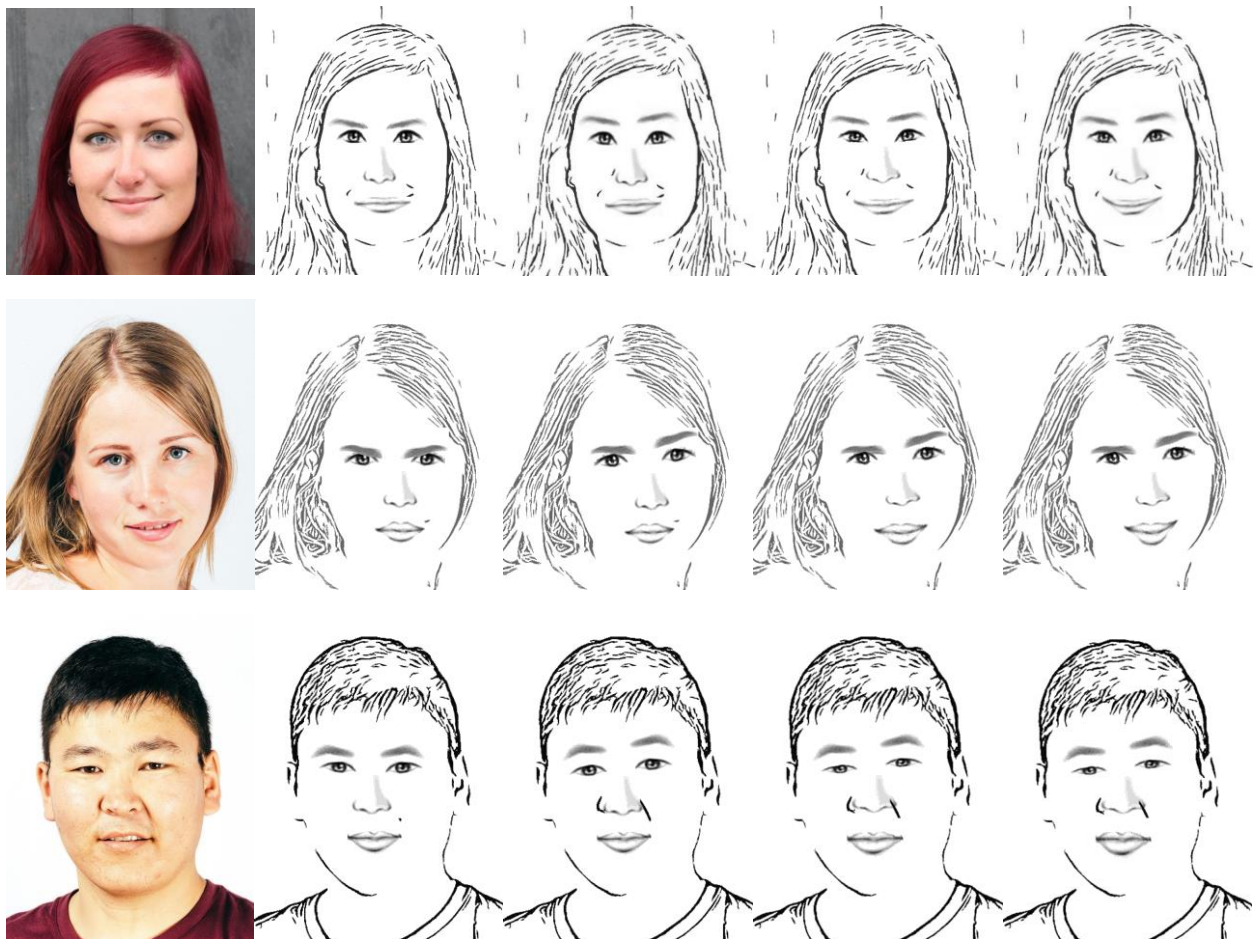


## Generation of Facial Cartoons

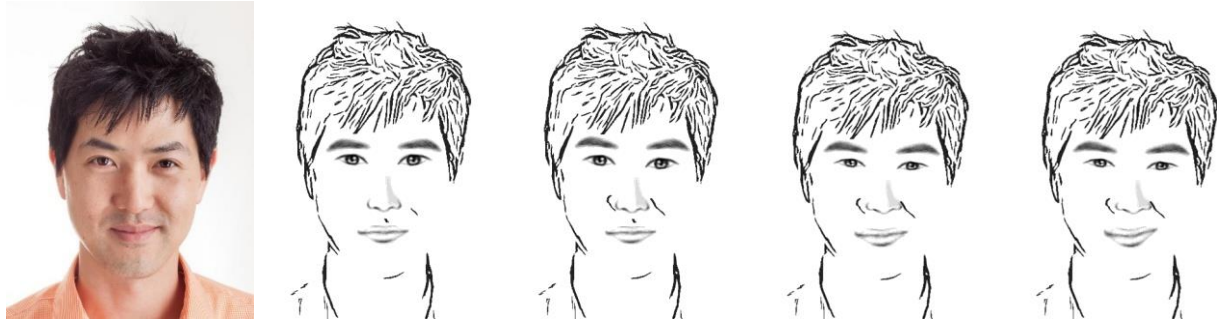
The results for the facial lines show that there are still some unwanted lines near the features but result from the first set is able to preserve the wrinkles on the top right side of the face, some outline of the beard and the “smile lines”. While the other three sets are able to preserve wrinkles around the eyes and large “smile lines”, which provide much more meanings to the cartoon images.

In summary, the results show that this is a good method to outline the hair and facial lines and proven to be better than the hair segmentation method. So, we will be using this as the replacement.

### 5.4 Transformation Progression



## Generation of Facial Cartoons



*Figure 33. Left to right: Input image. Results after scaling and translation. Results after feature rotation. Results after non-rigid transformation. Results after non-rigid transformation with enhanced landmarks difference on mouth.*

Results in this section show the development progress of my program. It is important to evaluate at which stage of the program it provides the best results. Also, to look at feature changes through the stages.

The first set shows differences in the eyes where they have been stretched horizontally to make the eyes wider. Also, a more pointed nose and most significantly, the mouth transformed from a flat shape to a smile which shows a major improvement in realism.

The second set shows the major differences in the orientation of the features where it will make more sense comparing with the input image. Also, the transformation of the nose and the mouth makes them look more realistic and fitting the shape of the original image.

The third and fourth sets shows changes in sizes of the eyes which are more realistic. Also, the orientation of the noses and the mouths are improved. However, arguably that the images in the third column maybe more attractive due to the larger eyes.

Overall, the results through the development stages show improvement on the realism aspect of the project, however, while some features such as the mouth have large improvements in both realism and attractiveness, other features such as the eyes look more attractive before the non-rigid transformation.

We will be using the results after non-rigid transformation as they are giving the most realistic output while preserving the attractiveness from the cartoon stylized features.

## 5.5 Transformation on Bad Examples



Figure 34. Left to right: Input image. Bad examples' results after feature rotation. Bad examples' results after non-rigid transformation.

The aim of this section is to see how well can the transformation methods deform the features and to decide the necessity of the matching algorithms.

## Generation of Facial Cartoons

The results from the first and the second sets show acceptable results with good facial expressions and the shapes of the features are not overly wrong. However, if we were to compare these results with the results in section 5.3, they do not have the same level of attractiveness and realism.

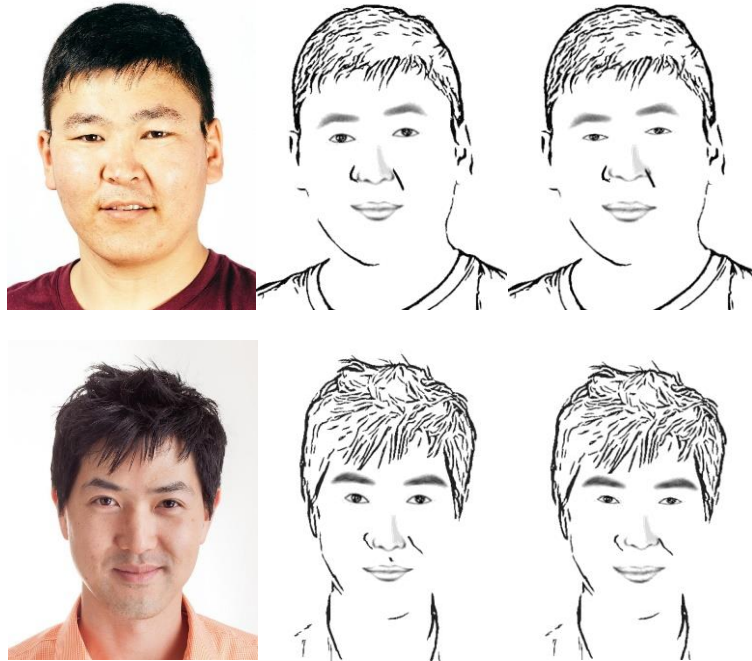
The results from the third and fourth sets show bad matches on the eyebrows and the noses, which make the images look poor. Even though the transformation attempts to deform the features and the facial expressions do look close to the input images, they do not provide a strong make-over for the features to look as good as the results from section 5.3.

Overall, the results from this section shows relevance in facial expressions but due to poorly shaped features, the results are not as good as results from section 5.3. Therefore, proving that the matching algorithms are important in this program despite having the non-rigid transformation functionality.

## 5.6 Feature Warping before matching



## Generation of Facial Cartoons



*Figure 35. Left to right: Input image. Feature warping before matching-result after feature rotation. Feature warping before matching-result after rigid transformation.*

The aim of this section is to see if the matches are better when features are first warped then matched.

If we compare results by looking features independently, the eyebrows for all four sets are very similar to the shape of the input images. Also, they are almost identical to the results from section 5.3.

Then the eyes for all except the first set are very similar to the input images, where the first set is slightly different. Although the shapes of the eyes are close for the third and fourth sets, the design of the extracted cartoon makes them less attractive but quite realistic.

Again, the noses are almost identical to the results in section 5.3.

For the mouths, the appearances of them are similar to the input images while different to the results from section 5.3. The point here is that they do not show any improvement over the 5.3 results. This could be due to the strong deformation on the mouth by the non-rigid transformation.

Overall, the results of feature warping before matching show some identical level of output that has no significance improvement while the eyes are less attractive. Therefore, this method cannot be a replacement for the current pipeline of this project.



## 5.7 Final Results



## Generation of Facial Cartoons



## Generation of Facial Cartoons

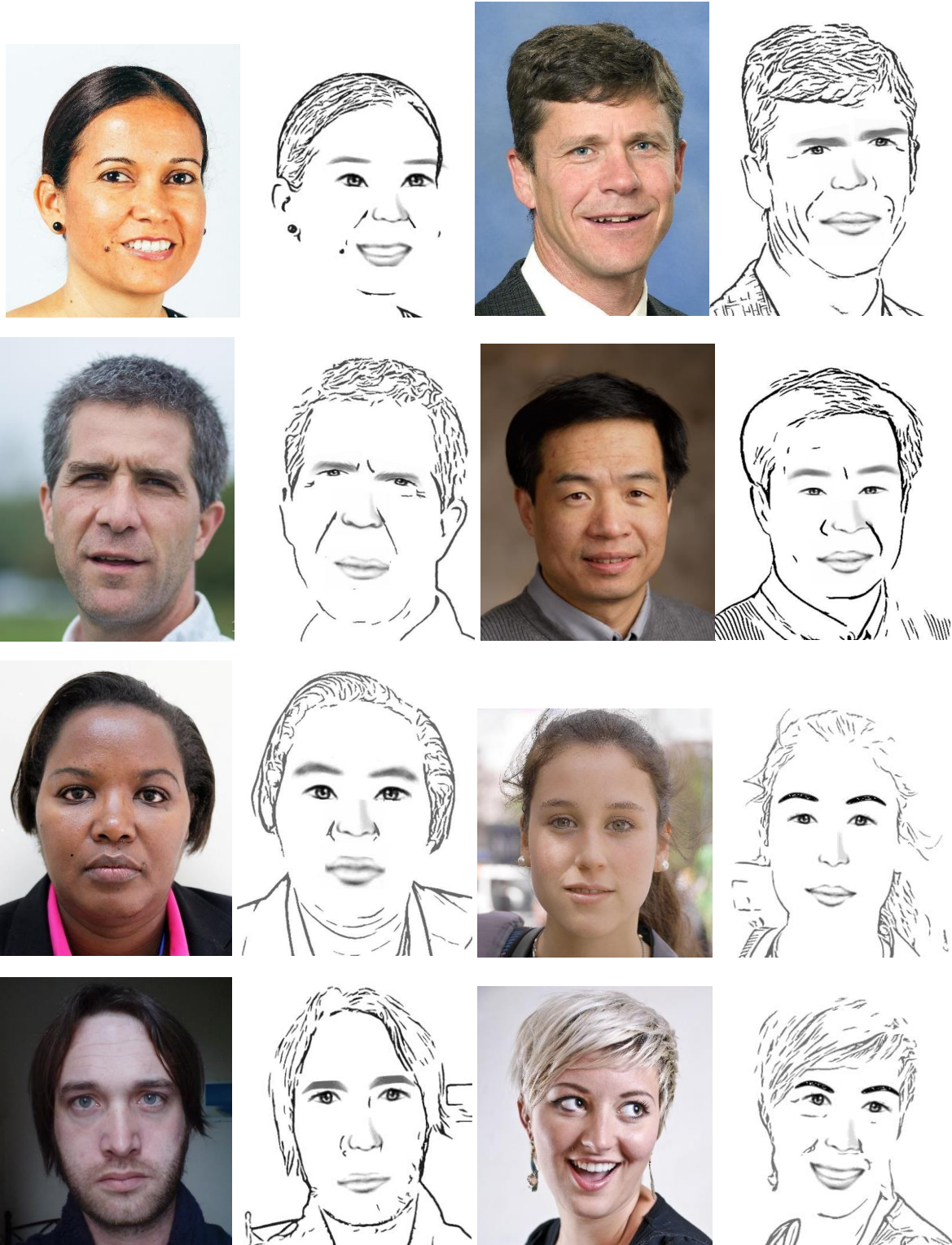


Figure 36. Finalised cartoon images with their corresponding input image.



The final results are presented as input image and cartoon pair. Input images are from different sets of sources which contain different background such as plain colour and natural environment. Features within the images are in different orientations.

The results show that regardless of the background types, the cartoons are not affected. Also, cartoon images do provide the attractive factor while still being recognizable.

### 5.7.1 Limitations

The negative side or the areas for improvement are for example the program is unable to recognize special expressions where results are not as attractive. Additionally, there is no track on the orientation of the face and the direction of vision, meaning in the cartoon images, there is an illusion where the direction of vision is the same as the orientation of the face. Although this may not affect the results in this section, it may possibly affect other input images.

The project also has only 28 examples in the library, which may not give a wide enough choice for all input images, therefore, resulting in bad matching in some features.

Another limitation of the image library is that there are three cartoon design styles, and each example only has one style:



*Figure 37. Cartoon images of three different design styles.*

Figure 37 shows the major differences between the design styles are the colours and the level of details. For example, the middle cartoon is more line drawing style filled with colours than the other two. Also, the left cartoon shows more shading on near the eyes and nose.

The problem of our output images as shown in figure 36 is that they will contain features in different level of details. If there is only one cartoon style, the output will be more consistent and may look better. However, the fact that we are using gray level output, we are minimizing the impact of the difference in colour brings to the output images.

Another problem is that if we look at results from section 5.6, which are the results generated by “warp before matching”, the eyes for the first, third and fourth rows are from the middle style, which provide less attractiveness and may be a reason why “warp before matching” does not show improvement for the results.

Finally, this project assumes the fact that the features are clearly visible. For example, if there is an image where the hair is totally blocking the eyebrows, the eyebrows are still visible on the face covering the hair, making the cartoon look rather bad.

### 5.7.2 Overall System Evaluation

After considering different aspects that can improve the results, our system has eliminated many methods that do not provide satisfactory results. The current system shows that it is able to process images that are in different backgrounds and orientations, also, results are in a good standard. On the other hand, there are still rooms for improvement such as more recognition techniques and extending the library to allow more accurate matchings.

Despite the fact that our results styles are very different to the results from other project mentioned in section 2.1, we attempt to compare:

Compare to section 2.1.1, our results show better realism with our hair styles and our facial lines. Also, the rotated and deformed features look more realistic. On the other hand, some of our images are lacking the attractiveness that their results have, such as modifying the size of features influences the appearances of the features.

Compare to section 2.1.2, our results may not show better realism due to their results being colour images. Also, they have a larger set of training images for matching, which allows them to have better matched features. However, our results contain facial lines which provide realism, also, our results have overall better outline of the face because of “Coherent-line-drawing”. Moreover, our results look more attractive due to the style of our cartoon features and the feature shapes are consistent if we compare our results with figure 7.

Compare to section 2.1.3, it is difficult to tell which results are more realistic because their results are simple line sketches which are missing a lot of details from the input images. However, the size of their features very close to the same as the input images. Whereas, in terms of attractiveness, our results would be better because of the details we provide.

Overall, different projects have different aims in creating their cartoon stylized images. While we can compare the positives or negatives between the small details in the images, we are not able to conclude the best results. Therefore, to summarize our results comparing with others, we have shown some better details in our results but we are still lacking the ability to create more attractive styles and more precisely matched features.

## 6 FUTURE WORK

---

### 6.1 Increase library size

One of the limitations of the program is the lack of examples in the library, causing some bad matches. To be able to minimise this problem, one solution is to add more examples to the library, which will require an artist due to limited sources for real face and cartoon pair. Also, we should generate a different cartoon style to better suit the theme of the input image.

### 6.2 Facial expression recognition

How a feature looks will differ due to the facial expression. For example, a blinking eye would result in a very different match compared to a normal eye. In the future, we should be looking at reading facial expressions and have different types of standard expressions to match with. For example, there could be some images in the library that represents a particular expression, and if the expression is read, then the program will be directed to match with those feature examples instead.

Another feature we should look further into is the mouth. Our program currently only recognizes a close or a largely open mouth, meaning the ones that are slightly opened are classified as closed. So, they are resulting in larger sizes. In the future, we should be able to have a more sophisticated method to classify a mouth, then matched with the library. The other option is to increase the types of mouth images in the library which allows the program to find the best match.

### 6.3 The direction of vision

As mentioned in the limitation section, there is an illusion that the direction of the vision is the same as the orientation of the face. In the future, we should be looking at detecting the direction of vision so the cartoon images are more realistic. Then we should compare the results to see if they are better than the results we currently have.

## 6.4 More details on the face

The program currently only considers lines that are generated by Coherent-line-drawing program. While some significant lines are extracted to enhance the cartoon, some lines are discarded in some images. What we should look at is a better method to find more details on the face, then discard the less significant ones. For example, if we develop a coloured cartoon style, we can detect colour changes, lightings and shininess on the face and reflect these onto the cartoon, which can enhance the attractiveness of the cartoon.

## 6.5 Create a background

Rather than a plain or line drawing background, in the future, we should be looking to develop a background for every cartoon image. These backgrounds do not need to be specialized themes from the input image. They can be randomly generated to enhance the cartoon for sole purpose.

## 6.6 Multiple faces in one image

Our program has only considered one face in one image. However, group photos are very often taken as a memory for an event or just a normal day. In the future, we should be looking to allow multiple faces to be generated in one cartoon image. By doing this, not only we are creating a cartoon face image for someone but we can create a better shared memory for a group of people using a group of recognizable cartoon faces.

## 7 CONCLUSIONS

---

In summary, this project develops a program that generates facial cartoons based on frontal face images using the most voted feature matches and some deformation techniques. The first part of the project is to extract features from example images and their stylized cartoon to create a feature library. The second part which is the program will extract features from input images and find the most voted matched example, where these selected features will go through both rigid and non-rigid deformation algorithms to improve realism. In the end, these features are placed onto the line drawings which contains the hair, the facial outlines and the facial lines.

While implementing the program, it was found that colour histogram analysis based hair segmentation had many problems and not feasible for the current stages of the project. Also, it is required to conduct user study to find the most suitable matching techniques.

The results of the generated cartoon images are reasonable with some good standards. Also, they do show balance between attractiveness and realism. Therefore, our aim has been achieved. However, there are still room for improvements which will be considered in our future work.

There are things that would have been done differently if the project starts now. For example, I would develop a more sophisticated matching algorithm, where we will only have a single best matching technique that is voted by a large group of users on a wide range of images. Secondly, the non-rigid transformation could change to use pixel-wise rather than landmark-wise to generate better deformed features, also, it would reduce the run-time of the program because we do not have to run the “Open Face” program multiple times.

## 8 REFLECTION ON LEARNING

---

Image processing has always been a topic of my interest throughout my time in Cardiff University. Even though I had limited experience in this area and non-photorealistic specifically other than the “Computer Vision” module taken in my third-year study, I am willing to learn through discussions with my supervisor and undertake research on topics that could improve the standard of my project.

The reason why I chose this project was that I am interested in the idea of automatic generation of cartoon images based on someone’s face. I like the fact that there are many similar ideas and tools in many current social applications and I think the idea of this project can really become popular. So, even though I found this topic quite difficult, I still chose it because I wanted to work on something I like and hoping to see the outcome.

In the beginning when I first discussed with my supervisor, he provided me with different approaches for this project and I was overwhelmed by some of the details presented in similar projects. So, in the first few weeks, I was struggling to find the correct method of doing things and resulted in many testing or going into the wrong direction.

In terms of the reason why I chose MATLAB as the programming language, it is because I have been using MATLAB throughout this course. So, it did help me to understand the functions I have implemented and it was a lot quicker to program in MATLAB compare with other languages.

Through the struggles that I had during this project, such as running out of ideas or struggling to understand certain topics or even not understand errors that I have. It showed the need of having a better plan with more background research on what I needed to do in this project. For example, there was time spent on how to solve some mathematical problems when I kept having errors but if I were to understand the theory first, there would not be time wasted.

Also, I would have managed my time better to spread evenly because there were times when I worked too hard during a week in order to complete planned task so I have some progress to show. However, I ended up getting sick which I spent more time in recovering.

Lastly, I believed that I have a good habit of backing-up my work and writing down notes for the things I have implemented, thought of and struggled with. These allow me to easily recover from over written images and files and certainly made it a lot easier to complete the report and reorganize the thought process I had for the past weeks. Therefore, I have learned to maintain these habits and improve the ones I should in the future projects.

## 9 REFERENCE

---

- [1] Archibald Fitzgerald. "Warping Using Thin Plate Splines - File Exchange - MATLAB Central". Uk.mathworks.com. N.p., 2009. Web. 14 Feb. 2017.
- [2] C-H.Rhee, C-H.Lee, "Cartoon-like Avatar Generation Using Facial Component Matching", International Journal of Multimedia and Ubiquitous Engineering Vol. 8. No. 4, July 2013.
- [3] Coherent Line Drawing: an open source line drawing standalone program modified from the paper 'Coherent Line Drawing' by Kang et al, Proc. NPAR 2007.
- [4] H.Chen, Y-Q.Xu, H-Y.Shum, S-C.Zhu, N-N.Zheng, "Exmaple-based Facial Sketch Generation with Non-parametric Sampling", Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001.
- [5] H.Kang, S.Lee, C.K.Chui, "Coherent Line Drawing", Aug.2007.
- [6] L.Weil, W.Gao, Y.Shen, Y.Zhu, R.Mo, Z.Peng, Y.Zhang, "A Novel Hierarchical Decomposition Model for Facial Caricature Synthesis", 5th International Congress on Image and Signal Processing, 2012.
- [7] Mathworks. "Compute Peak Signal-To-Noise Ratio (PSNR) Between Images - Simulink - Mathworks United Kingdom". Uk.mathworks.com. N.p., 2006. Web. 12 Jan. 2017.
- [8] Mathworks. "Remove Small Objects From Binary Image - MATLAB Bwareaopen - Mathworks United Kingdom". Uk.mathworks.com. N.p., 2006. Web. 5 Feb. 2017.
- [9] Mathworks. "Structural Similarity Index (SSIM) For Measuring Imagequality - MATLAB Ssim - Mathworks United Kingdom". Uk.mathworks.com. N.p., 2014. Web. 12 Jan. 2017.
- [10] OpenFace: an open source facial behavior analysis toolkit Tadas Baltrušaitis, Peter Robinson, and Louis-Philippe Morency, in IEEE Winter Conference on Applications of Computer Vision, 2016.
- [11] Q. Chen, D. Li, and C.-K. Tang, "KNN matting," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2012, pp. 869–876.

- [12] S.Liu, H.Li, L.Xu, "Face Cartoon Synthesis Based on the Active Appearance Model", IEEE 12<sup>th</sup> International Conference on Computer and Information Technology, 2012.
- [13] Y.Yacoob, L.S.Davis, "Detection and analysis of hair," IEEE Trans. Pattern Anal. Mach. Intell., vol. 28, no. 7, pp. 1164–1169, Jul. 2006.
- [14] Y.Zhang, W.Dong, C.Ma, X.Mei, K.Li, F.Huang, B-G.H, O.Deussen, "Data-Driven Synthesis of Cartoon Faces Using Different Styles", IEEE Transactions on Image Processing, 2016.



## 10 APPENDIX

---

Appendix A. Tables showing full results of user study on similarity measure.

EXAMPLE	EYEBROWS		EYES		NOSE		MOUTH	
	SSIM	Best Vote	SSIM	Best Vote	SSIM	Best Vote	SSIM	Best Vote
Z1	1	0	0	1	1	0	1	1
Z2	1	1	1	0	1	1	1	0
Z3	1	1	0	1	1	1	1	1
Z4	1	1	0	1	0	1	0	1
Z5	1	1	0	1	1	1	1	0
Z6	0	1	1	1	1	1	1	1
Z7	0	1	1	1	1	1	0	1
Z8	1	0	1	0	0	1	1	1
S1	1	1	0	1	0	1	1	1
S2	1	1	1	1	1	0	1	1
S3	1	0	0	1	1	1	1	1
S4	1	1	0	1	1	0	0	1
S5	1	1	1	1	1	1	0	1
S6	1	1	1	1	1	1	1	1
N1	1	1	1	1	1	1	1	1
N2	1	1	0	1	1	0	1	1
N3	1	1	1	1	0	1	1	1
N4	1	0	1	1	1	1	1	1
N5	1	1	1	1	0	1	1	1
N6	0	1	0	1	0	1	0	1
N7	1	1	0	1	0	1	1	1
N8	1	0	1	1	1	1	1	1
N9	0	1	0	1	0	1	1	1
N10	1	1	1	1	1	1	1	0
N11	0	1	1	0	0	1	1	1
N12	1	1	1	0	1	1	1	1
N13	1	1	1	0	1	0	0	1
N14	0	1	0	1	0	1	1	1
N15	0	1	1	0	1	1	1	1
N16	1	1	1	1	1	1	1	1
N17	1	1	0	1	1	1	1	1

## Generation of Facial Cartoons

<b>N18</b>	1	1	0	1	1	1	1	1
<b>N19</b>	0	1	1	1	1	0	1	1
<b>N20</b>	1	0	0	1	0	1	1	1
<b>Q1</b>	1	0	1	1	1	1	0	1
<b>Q2</b>	1	1	0	1	1	1	1	0
<b>Q3</b>	1	1	1	1	1	1	1	1
<b>Q4</b>	1	1	1	1	0	1	1	1
<b>Q5</b>	1	1	1	0	1	1	1	1
<b>Q6</b>	1	1	0	1	0	1	1	1
<b>Q7</b>	0	1	0	1	1	0	1	1
<b>Q8</b>	1	0	1	0	1	0	1	0
<b>Q9</b>	1	1	1	1	1	0	1	1
<b>Q10</b>	1	1	0	1	0	1	1	1
<b>Q11</b>	1	1	1	0	0	1	1	0
<b>Q12</b>	1	1	1	0	1	1	0	1
<b>Q13</b>	1	1	1	0	1	1	0	1
<b>Q14</b>	1	1	1	1	1	1	1	1
<b>Q15</b>	1	1	0	1	1	0	1	1
<b>Q16</b>	1	1	0	1	1	1	1	1
<b>Q17</b>	0	1	0	1	0	1	1	1
<b>Q18</b>	1	1	0	1	1	1	1	1
<b>Q19</b>	1	1	1	1	0	1	0	1
<b>Q20</b>	1	1	0	1	1	1	0	1
<b>TOTAL</b>	44	46	30	43	37	44	43	48

## Appendix B. Tables showing full results of states comparison user study.

<b>WHOLE IMAGE</b>				
	line2cartoon	line2line	photo2photo	Total
<b>Z1</b>	26	36	13	75
<b>Z2</b>	10	32	34	76
<b>Z3</b>	8	33	34	75
<b>Z4</b>	11	33	31	75
<b>Z5</b>	13	37	25	75
<b>Z6</b>	9	36	30	75
<b>Z7</b>	9	43	23	75
<b>Z8</b>	12	44	19	75
<b>Z9</b>	13	40	22	75
<b>TOTAL</b>	111	334	231	676

## Generation of Facial Cartoons

### EYEBROWS

	line2cartoon	line2line	photo2photo	Total
<b>Z1</b>	25	38	16	79
<b>Z2</b>	12	22	41	75
<b>Z3</b>	16	21	44	81
<b>Z4</b>	18	25	37	80
<b>Z5</b>	28	5	48	81
<b>Z6</b>	21	33	34	88
<b>Z7</b>	21	48	15	84
<b>Z8</b>	16	36	27	79
<b>Z9</b>	18	36	27	81
<b>TOTAL</b>	175	264	289	728

### EYES

	line2cartoon	line2line	photo2photo	Total
<b>Z1</b>	18	35	36	89
<b>Z2</b>	25	22	34	81
<b>Z3</b>	25	22	35	82
<b>Z4</b>	21	28	34	83
<b>Z5</b>	20	25	32	77
<b>Z6</b>	7	42	39	88
<b>Z7</b>	14	41	24	79
<b>Z8</b>	7	39	32	78
<b>Z9</b>	18	45	14	77
<b>TOTAL</b>	155	299	280	734

### NOSE

	line2cartoon	line2line	photo2photo	Total
<b>Z1</b>	18	38	24	80
<b>Z2</b>	30	31	24	85
<b>Z3</b>	23	44	29	96
<b>Z4</b>	13	38	27	78
<b>Z5</b>	42	9	28	79
<b>Z6</b>	19	20	38	77
<b>Z7</b>	16	41	24	81
<b>Z8</b>	12	38	22	72
<b>Z9</b>	21	26	31	78
<b>TOTAL</b>	194	285	247	726

## Generation of Facial Cartoons

### MOUTH

	line2cartoon	line2line	photo2photo	Total
<b>Z1</b>	37	27	16	80
<b>Z2</b>	16	35	35	86
<b>Z3</b>	12	35	30	77
<b>Z4</b>	7	32	41	80
<b>Z5</b>	23	30	26	79
<b>Z6</b>	6	40	32	78
<b>Z7</b>	17	37	32	86
<b>Z8</b>	24	35	19	78
<b>Z9</b>	6	46	27	79
<b>TOTAL</b>	148	317	258	723

	LINE2CARTOON	LINE2LINE	PHOTO2PHOTO	TOTAL
<b>WHOLE IMAGE</b>	111	334	231	676
<b>EYEBROWS</b>	175	264	289	728
<b>EYES</b>	155	299	280	734
<b>NOSE</b>	194	285	247	726
<b>MOUTH</b>	148	317	258	723
	783	1499	1305	3587
<b>PERCENTAGES</b>	21.82882632	41.78979649	36.3813772	100