Coursework Assignment

Deadline: 29th May 2020 at 4PM

This assignment brief was first released on 18th March 2020

1 Assignment Overview

This assignment will involve you designing, building, testing and critiquing a system for performing face alignment, aka. locating facial landmarks in images. There are also some small extension tasks detailed below.

It is worth 100% of the grade for this module. It's designed to ensure you can demonstrate achieving the learning outcomes for this module, which are:

- Write and document a computer program to extract useful information from image data.
- Propose designs for simple computer vision systems.
- Determine the applicability of a variety of computer vision techniques to practical problems.
- Describe and recognise the effects of a variety of image processing operations.

1.1 Extension Tasks

1. Provide a simple solution for segmenting the pixels of the face and/or skin (10%).
2. Implement a simple graphical effect that uses the information you have extracted from the image (10%).

2 What to hand in?

1. A report that comprises a *maximum* of 8 pages and 1500 words, including captions but excluding references. I'm expecting several pictures, diagrams, flowcharts and charts to be included.
    - A summary and justification for all the steps in your face alignment system, including discussion and quantitative comparisons between different approaches that you have experimented with. Explaining diagrammatically is very welcome.
    - A short summary of your face segmentation system and graphics effects methodology.
    - Qualitative examples of your methods running on the small set of provided example images, found in the compressed numpy file (examples.npz) here. Include pictures of the result of face alignment, face segmentation and graphical effects systems.

- Examples of failure cases in the face alignment system and a critical analysis of these, identifying potential causes and solutions.

2. A CSV file that contains the face landmark positions on the test set of images. Use the provided "save_as_csv" function in the colab worksheet to process an array of shape (number_test_image, number_points, 2) to a csv file.

3. Either links to Colab worksheets, .ipynb files, .py files containing annotated code for any data preprocessing, model training and testing.

3 How will this be graded?

The breakdown of marks for this assignment are given below:

25 Marks **Accuracy and robustness of face alignment**
These marks are allocated based on the performance of the face alignment method. This will be evaluated on the held out test set, which includes some difficult cases. The test images, without annotations are provided in the compressed numpy file (test_images.npz) here and the error on the predicted points will be calculated after submission. Marks will be awarded for average accuracy and robustness (% of images with error below a certain threshold) as well as qualitative evaluation on the example images.

30 Marks **Outline of methods employed**
Justifying and explaining design decisions for the landmark finding. This does not have to be in depth, and I do not expect you to regurgitate the contents of the lecture notes/papers. You should state clearly:

- what methods you have used, with what training parameters and why.
- what image features you have used, briefly describe how they were calculated, and why you chose them.
- any image pre-processing steps you have used, and why.

For top marks, you should clearly demonstrate a creative and methodical approach for designing your system, drawing ideas from different sources. Explaining using diagrams and/or flowcharts is very welcome.

20 Marks **Analysing results and failure cases**
Critically evaluate the results produced by your system on test/validation data. You should include quantitative and qualitative comparisons between different approaches that you have tried. Investigate and identify systematic failure cases, and propose potential solutions.

10 Marks **Face and/or skin segmentation**
Very briefly outline/provide a diagram/flowchart explaining how this was performed. Provide some qualitative results and illustrate some failure cases. I am only expecting very simple solutions. Marks for accuracy of segmentation on the qualitative test images and elegance of the solution.

10 Marks **Graphical Effects**
Very briefly outline/provide a diagram/flowchart explaining how this was performed. Provide some qualitative results and illustrate some failure cases. Marks for successfully applying the effect to a variety of images, creativity and elegance of the solution.

5 Marks **Code annotation** is for annotating sections of the training/testing code with what they do. To get maximum marks, explain each algorithmic step (not necessarily each line) in your notebook/.py files.

A few general points on the report.

- Provide references, if you read anything useful.
- Diagrams, flowcharts and pictures are very welcome, make sure you label them properly and refer to them from the text.
- Plots should have labelled axis.

4 What resources are provided for me?

The training images are provided for you in a compressed Python array. They have already been preprocessed to be the same size with the faces roughly in the middle of the image and the eye corners in the same position. The training data can be downloaded as a compressed numpy file (training_images.npz) here.

The data can be read by:

```
import numpy as np

# Load the data using np.load
data = np.load('training_images.npz', allow_pickle=True)

# Extract the images: shape = (2811, 250, 250, 3)
images = data['images']
# and the data points: shape =  (2811, 68, 2)
pts = data['points']
```

In this very basic colab worksheet I provide code for:

- Loading the data
- Estimating the error between predictions and ground truth.
- Visualising points on an image
- Saving the results to a .csv file

A set of test images, without landmarks is provided in the compressed numpy array (test_image.npz) here. This data is loaded the same way as before, but there are no points stored in the file.

I also include 6 images to use for qualitative comparisons found in the compressed numpy array (examples.npz) here. These images should be included in your report to demonstrate face alignment, segmentation and graphical effects.

4.1 Notes on using Colab

Either you can complete this project using Google colab, which gives you a few hours of computing time completely free of charge, or you can use your personal/lab machine. If you are using Google colab, try and familiarise yourself with some of it's useful features. To keep your saved models, preprocessed data etc. you can save it to Google drive following the instructions here. You can also directly download a file you make in colab using the code below:

```
from google.colab import files
files.download(filename)
```

If you're refactor code into extra .py files, these should be stored in your google drive as well, or on Box such that they are easy to load into your Colab worksheet.

4.2 Most important links

Contents	**filetype**	**links**
Training images and points	compressed numpy array (training_images.npz)	link
Test images	compressed numpy file (test_images.npz)	link
Examples images for qualitative comparisons	compressed numpy file (examples.npz)	link
Colab worksheet with some useful functions	colab worksheet	link

4.3 What library functionality can I use?

You're free to use fundamental components and functions from libraries such as OpenCV, numpy, scipy, scikitlearn to solve this assignment, although you don't have to. Here, fundamental components refers to things like regression/classification models and pre-processing/feature extraction steps and other basic functionality. What you are **not** allowed to use are library functions that have been written to directly solve the tasks you have been given, i.e. face alignment/segmentation/graphical effects.

In terms of tools and frameworks, it's absolutely fine to use convolutional neural networks (CNNs), which are introduced in fundamentals of machine learning. The best packages would be either TensorFlow (probably with Keras) or PyTorch. If you use such an approach you should be sure to document how you chose the architecture and loss functions. A well justified and high performing CNN approach will receive equivalently high marks as if you'd built it any other way.

In terms of sourcing additional labelled data, this is **not** allowed for this assignment. This is because in real-world commercial projects you will typically have a finite dataset, and even if there are possibly useful public datasets available, their license normally prohibits commercial use. On the other hand data augmentation, which effectively synthesises additional training examples from the labelled data that you have, is highly encouraged. If you use this, please try and add some text or a flow-chart of this process in your report.

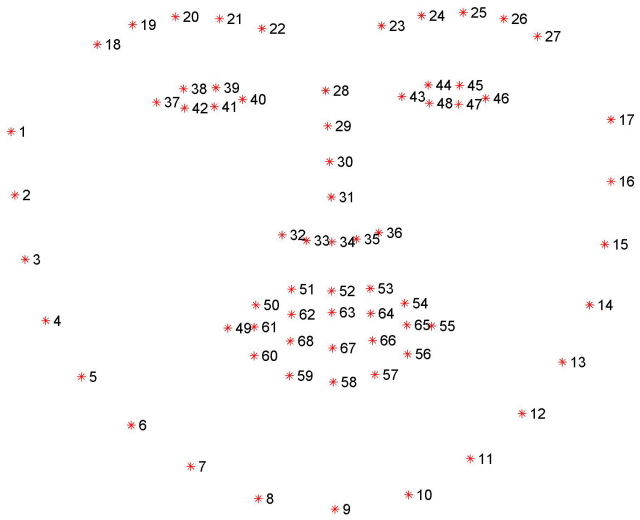

Figure 1: Illustration of the **1-indexed** (counting from 1, rather than 0 as you would in Python) locations of the points on the face. For example, if we wanted to find the tip of the nose, that's index 34 so we would look up points[34-1,:], which would give you the x and y coordinate of the tip of the nose.

5 Where do I start

5.1 Face Alignment

Face alignment is covered in lecture 14, so that's a good place to look for information. I briefly discussed the assignment at the end of the lecture, which you can listen to on Canvas. I've also included some references below.

I have included a very basic colab worksheet illustrating how to load the data and visualise the points on the face.

The simplest approach would be to treat this as either a regular or a cascaded regression problem. To follow this approach you will need to consider what image features are helpful to predict the landmarks and whether some pre-processing is required on the data. One simple way to proceed is to choose a set of locations, either evenly spaced across the image, or in some more useful pattern (think about where in the image you might want to calculate more informations). Using a feature descriptor, such as SIFT, you could predict the landmark locations using linear regression (see FML lecture 13) and the associated lab.

You're not restricted to taking this approach, and for higher marks creativity is very much encouraged. Face alignment has seen a lot of interesting and varied ideas, and if you find some good ideas while reading around the topic that would be great.

5.2 Face Segmentation

We're looking for a simple solution, so anything more than 20 lines of code is too much. The methods we're looking for would involve using the predicted landmark geometry and/or colour. I am intentionally not providing a training set of data for this task. There's some useful code in OpenCV, take a look at cv2.fillPoly.

5.3 Graphical Effects

Think about what simple graphical effects you could perform to a face, given that you know the location of the landmarks and if you do the other extension, all the skin pixels. Exercise your creativity and try and do something amusing, although please note that anything rude or offensive will yield 0 marks.

6 Top Tips for Success

- Remember Occam's razor, complexity should not be added unnecessarily. The more complicated your system the more things to explain/justify etc.
- Start with a simple achievable goal and use that as a baseline to test against. Keep track of early models/results to use as points of comparison.
- Remember that even if it doesn't work well, having a go at the extension tasks is worth a few marks. We're only looking for simple solutions.
- You don't need to work at very high resolution to get accurate results. Particularly when doing initial tests, resize your images to a lower resolution images. Make sure you also transform your training points so they are in the same geometry as the image. For your predicted points, make sure these are all at the same resolution as the original images.
- Think about things that you've learned about in FML as well as Computer Vision. Dimensionality reduction could be helpful. Overfitting and outliers may be an issue, and you should consider using methods to minimise this.

7 Further reading

Face alignment is a reasonably well researched field, and a wide variety of methods have been proposed. Some relatively recent approaches are documented below. [1] is probably a good one to look at, [2] contains a survey of methods, which might give you some ideas and [3] describes the results of a competition. The other references are very much optional reading on other population recent methods.

References

[1] Xiong X, De la Torre F. Supervised descent method and its applications to face alignment. InProceedings of the IEEE conference on computer vision and pattern recognition 2013 (pp. 532-539). Paper link.

[2] Learned-Miller E, Huang GB, RoyChowdhury A, Li H, Hua G. Labeled faces in the wild: A survey. InAdvances in face detection and facial image analysis 2016 (pp. 189-248). Springer, Cham. Paper link.

[3] Sagonas C, Antonakos E, Tzimiropoulos G, Zafeiriou S, Pantic M. 300 faces in-the-wild challenge: Database and results. Image and vision computing. 2016 Mar 1;47:3-18. Paper link.

[4] Cao X, Wei Y, Wen F, Sun J. Face alignment by explicit shape regression. International Journal of Computer Vision. 2014 Apr 1;107(2):177-90. Paper link.

[5] Burgos-Artizzu XP, Perona P, Dollr P. Robust face landmark estimation under occlusion. InProceedings of the IEEE international conference on computer vision 2013 (pp. 1513-1520). Paper link

[6] Zhu S, Li C, Change Loy C, Tang X. Face alignment by coarse-to-fine shape searching. InProceedings of the IEEE conference on computer vision and pattern recognition 2015 (pp. 4998-5006). Paper link