# School of Mathematics, Computer Science and Engineering

## INM460: Computer Vision

## Coursework for PG students: Face recognition

Module leader: Dr. Giacomo Tarroni
Email: giacomo.tarroni@city.ac.uk

### Submission details

You must submit your project (as a single .zip file) to Moodle by **5pm, Thursday 30 April 2020**. The submission must include your project report (PDF format), all the produced source code, a readme.txt file (with the instructions on how to run the implemented methods), the trained models, and a screen capture video (mp4, or any other format executable by the latest version of the VLC player).

Standard lateness penalties (late submission = 0 marks) and extensions policy applies. This assignment constitutes **50% of the total marks for this module**. In practice, the coursework is marked out of 100%, with the final mark scaled to 50%.

### Concept and task

Have you ever been introduced to a number of people, but forget their names? Humans are exceedingly capable of recognising faces previously seen before, but matching names to faces can be difficult. Imagine having an app running on your device (e.g. smartphone, glasses) that recognises people and prompts you with their name, to help you avoid blurting an awkward “I’m sorry but I forgot your name?”.

Person identification is important for a variety of practical tasks, including cybersecurity and authentication. In this project, you will develop a computer vision system to detect and identify individuals using a database of known face images (specifically, students in this module).

### Code specifications

It is recommended you use Matlab (preferably 2018a or later) for the development of the coursework in this module. Matlab is an exceedingly powerful programming language and environment for imaging and computer vision, and includes a variety of toolboxes that provide functionality of interest to this project, including the Computer Vision System Toolbox, Image Processing Toolbox, Optimization Toolbox, Neural Networks and Statistics and Machine Learning Toolbox. As an alternative to Matlab, you are allowed to use Python with relevant libraries. This is however not recommended, since the lecture notes, code samples and lab exercises will be based on Matlab. While some help might be provided, academic support from the lecturers and tutors will be limited to Matlab.

Spend some time familiarising yourself with Matlab, particularly if you haven't used it before. Also, be sure to spend sufficient time working on the lab tutorials and code samples. Look at online sources (documentation, blogs, tutorials, videos) on computer vision in Matlab, including the Computer Vision System Toolbox.

At times you may wish to discuss the methods used with your colleagues. This is encouraged. However, **the coursework is individual**. Plagiarism will be dealt with directly by the department's senior staff and may lead to course disqualification. Your work must be your own and you must cite any and all resources used.

## Deliverables

### Project reporting (30%):

Produce relevant documents and materials to describe how you carried out the coursework. In particular:

- Prepare a project report (**maximum 10 pages**) in PDF format (20%):
  - The report should include an overview of your project followed by a description of your implemented methods (with more emphasis spent on your best performing one). Include a brief block diagram showing the **key** algorithmic components that are part of your best performing method;
  - For each task (i.e. other deliverables), list briefly the requirements that were implemented. Describe your implementation, review and discuss any relevant theory underpinning the methodology. If a requirement was skipped, say so;
  - If you initially tried approaches that failed, mention these briefly, and provide a rationale on why they were ineffective;
  - Provide a quantitative analysis of when the best performing method is successful and conditions when it may fail to achieve the desired result;
  - **Reference any external source code or pretrained models** used;
  - Include a discussion section reflecting on the project. Consider the strengths and weaknesses of what you achieved in the time provided. Also discuss how you might expand the project to improve performance or take it in new directions.
- Provide your commented source code (5%):
  - Follow a logical design, organisation, and coding style;
  - All code has to be commented (at the block level is sufficient);
  - **Reference any external source code** used;
  - Use appropriate naming conventions and display an organised code strategy through the use of functions.
- Provide a screen capture video in which you test the implemented methods on images from the evaluation set (that you will create from the provided datasets) (5%):

- The video must be 3 minutes maximum (anything beyond 3 minutes will not be watched);
- It should show the results produced by the implemented methods by displaying the obtained results on images from the evaluation set;
- The video does not have to be exhaustive, but has to show that your code actually works and provides some results;
- It should contain an audio commentary (recorded by you) providing a short but relevant description of the shown methods as they are executed. Keep it simple!

## Face recognition (55%)

Develop a Matlab function,

```
P = RecogniseFace(I, featureType, classifierType, creativeMode)
```

that returns a matrix **P** describing the student(s) present in an RGB image I. **P** should be a matrix of size N x 3, where N is the number of people detected in the image. The three columns should represent:

- ID: a unique number associated with each person. If the person is present in the database of individual pictures, the ID is the number held by the person;
- x: the x coordinate of centre of the face for the person detected in the image;
- y: the y coordinate of centre of the face for the person detected in the image.

For example, if the function detects two students in the image, with student with ID=03 having his/her face centred at position [144, 153], and student with ID=13 at position [312, 123], the **P** matrix would be

```
P =
    03 144 153
    13 312 123
```

During marking, your function will be tested on images not available in the provided datasets.

- Implement `RecogniseFace` so that it correctly identifies the location of faces **in both individual and group images**. Executing the function must also automatically display the input image with the face locations superimposed as bounding boxes together with the students' IDs (10%);
- The `RecogniseFace` function must have two additional arguments, `featureType` and `classifierType`, which allow different features (e.g. SURF, HOG) or classifiers (e.g. SVMs, MLPs, CNNs) to be used (up to 30%). Use **at least three combinations** of features-classifiers. Different points will be awarded for different combinations:
  - For each general features-classifier combination (5%);
  - For using a CNN classifier (20%). In this case, the content of `featureType` is not used.

  The points scored for this task will be given by the sum of the implemented combinations (e.g. correctly implementing a SURF-SVM, a HOG-SVM as well as a CNN will provide 30%);

- At least for your best performing features-classifier combination, add to your report performance metrics (e.g. percentage of faces correctly identified, confusion matrices) **obtained on the evaluation dataset at least for what concerns the individual images** (15%). Show examples of correct and, if any, incorrect detections on both individual and group images.

## Creative mode (15%)

When the input variable `creativeMode` is set to 1, `RecogniseFace` also alters the faces in a creative way and shows the result in the displayed image. The alteration should be limited to the people's actual faces. Examples include (but are not limited to):

- Cartoonifying the face by reducing the number of colours using a clustering algorithm;
- Replacing the person's head with a different object;
- Augmenting the image by superimposing content (e.g. sunglasses, hats, beards) onto the person's face.

## Implementation notes

- Create a sensible training/evaluation split in the "Coursework Datasets" available on Moodle;
- For the images in your training set, perform face detection to find faces, and save the extracted faces as image samples. Associate each sample with a unique label, which has to be the number held by the person in the individual images. This will produce a labelled dataset of image samples that can be used for training;
- (Optional) The association between faces and labels can be done manually, without the need to apply OCR algorithms to the handheld signs. However, submissions that will implement this feature in their code will be awarded additional points;
- Throughout the project, feel free to use pre-trained models, but carefully reference them in your report and produced code;
- (Optional) `RecogniseFace` should be able to assign as many correct IDs as possible on a previously unseen group image. If some faces do not correspond to any of the ones available in the individual images (i.e. for students who did not have their individual picture taken), it could assign individual "unknown" labels (i.e. "unknown 01", "unknown 02", etc.);
- If no faces are detected, `RecogniseFace` should return an empty matrix;
- Do not panic if some features-classifier combinations do not provide very accurate results: this is expected for some of them. The aim of the coursework is to assess your ability to properly and effectively implement and evaluate the methods you have learned in class, not to necessarily achieve extremely high accuracy in all cases.

## Submission notes

- You should deliver your project as a single .zip file (note this must be less than 200MB) uploaded to Moodle before the deadline. The file should include:
  - The report (PDF format)
  - The screen capture video (preferably mp4, but at least readable by VLC player)
  - All the produced code (commented)
  - The trained models
  - A readme.txt file

- **The readme.txt** file must describe **all the valid ways to call your `RecogniseFace` function** (i.e. arguments to pass in as `featureType` and `classifierType`) to allow easy testing on unseen images;
- If functions/libraries/packages are used beyond those in the standard City, University of London Matlab distribution, make sure that these are sufficiently packaged with the provided source code so that `RecogniseFace` can run on a machine with Matlab without having to set up any complicated file paths;
- Include all trained models so that the implemented function can run autonomously **without the need for re-training: your models will not be trained during marking**;
- Add the code you produced to train your models to your code submission, but **do not include the image datasets used for training**;
- Make sure that your `RecogniseFace` function can properly execute without runtime errors by testing your submission on a different computer (even using the facilities at City, University of London);
- If your models exceed the 200MB limit, you can upload **your models alone** (**report, source code, readme.txt file and screen capture video must be submitted to Moodle**) on a personal cloud space (e.g. Dropbox) and **add a link to them in the readme.txt file** together with an indication of where to store them to allow the seamless execution of your methods.