Lab 2B: Face Recognition

CM30080

Dr Matthew Brown

Introduction

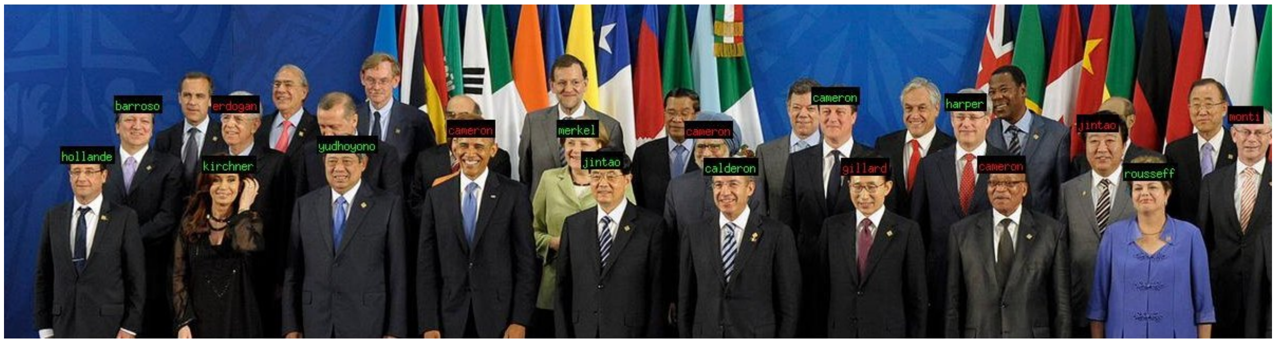

Figure 1: Face Recognition in the G20. Correct classifications shown green.

In this lab you will implement an algorithm that can recognise faces in images. This will be split into two parts, 1) Face Detection: where you will find bounding rectangles for face candidates in the image, and 2) Face Classification: where you will use a training dataset to identify the person present. A dataset of faces for the G20 leaders is provided for you (see Figure 3). To start, revise the lecture notes and Szeliski's book [Sze11] chapters 14.1 and 14.2. You should also have completed the MATLAB skills lab 2A.

Report and Hand-in

You should create a report that documents your method, including source code and relevant comments, and example output. You should order your report by Requirement number (Req #) as detailed in the sections below. Marks are given beside each Req #, out of a total of 100.

Your report and all source files should be handed in via Moodle. You should submit two files: *lastname_firstname.pdf* containing your report in pdf format, and *lastname_firstname.zip* containing all of your source code. The **deadline** for coursework hand-in is **Monday 4th May** at 4pm. The coursework is worth **25%** of the overall mark for the course.

Skeleton Code

Skeleton and helper code for the face detector and classifier are included in `FaceDetect.m` and `FaceClassify.m`. Note that Requirements #3 and #4 do not depend on #1 and #2

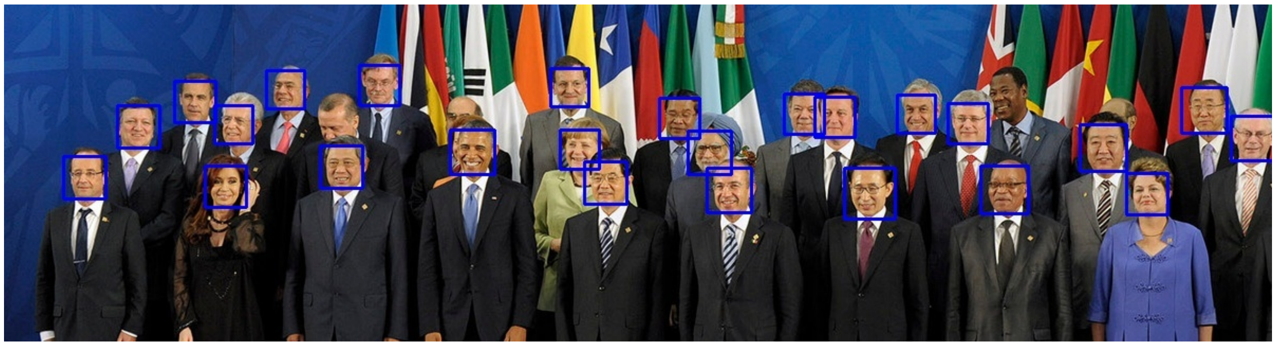
Figure 2: Faces detected using the Viola-Jones algorithm[VJ01]

so you can tackle these parts independently.

Req #1: Correlation Detector [25 marks]

For this part you will build a face detector based on correlation. To do this, you will select a template based on part of the image or a pre-existing face. The first thing to try is a simple correlation of the template with the image, using the `filter2` function in MATLAB (see Lab 2A).

Implement the correlation where shown in `FaceDetect.m` and then run this script. This will pass your correlated image to the function `FindLocalMaxima` which uses non-maximal suppression to find local maxima. It will then show the detections on the image, and report the percentage overlap with detections from the Viola-Jones face detector [VJ01] as a measure of success.

Test the code with a few different correlation windows and the supplied template to see how the matching works. What sort of errors (false positives and false negatives) are common?

Next you will try to improve your detector using normalised correlation. The simple correlation above could be seen as computing the dot product $\mathbf{x}^T\mathbf{y}$ between the vectorised template $\mathbf{x}$ and the portion of image beneath $\mathbf{y}$. To use the normalised correlation you will instead need to compute

$$r = \frac{(\mathbf{x} - m_x\mathbf{1})^T(\mathbf{y} - m_y\mathbf{1})}{|\mathbf{x} - m_x\mathbf{1}||\mathbf{y} - m_y\mathbf{1}|}$$

where m_x, m_y are the means of the vectorised template and image, $\mathbf{x}$ and $\mathbf{y}$, and $\mathbf{1}$ is a vector of 1's. What issues can you see with implementing this directly? Simplify the above expression and try to devise an efficient way to compute it using correlations. Include a description of your method and commented source code for your report.

Req #2: Improved Detector [25 marks]

In this part you will try out your own ideas to improve the detector. You should clearly describe the ideas that you have tried, including justification for the methods you chose,

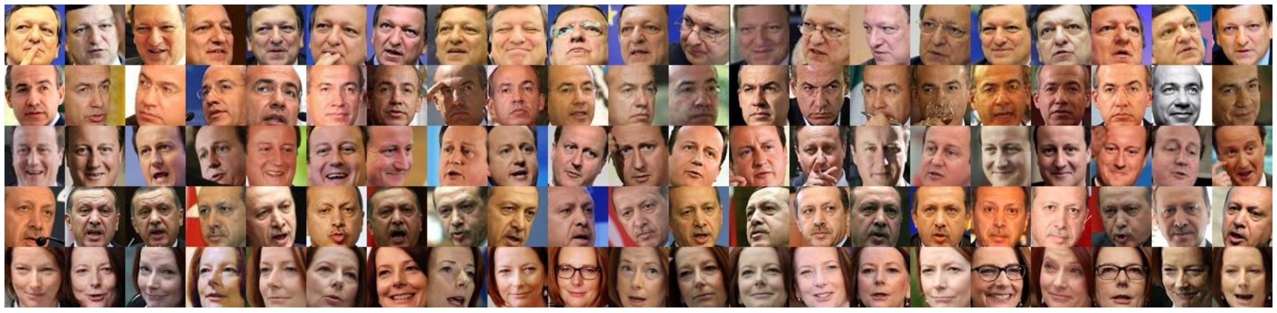

Figure 3: Face Dataset: this shows a subset of the G20 face dataset (the full dataset has 32 images each for 20 leaders)

and results of testing different configurations. You can experiment with the template design and feature extraction, using multiple templates, or any other idea of your choice. In your report, summarise the results of your experiments and explain your final detector design. Note that you will need to correctly handle edge effects in convolutions when using the fixed template from the face database (see Lab 2A).

Req #3: Nearest-Neighbour Classifier [25 marks]

In this part you will build a nearest-neighbour face classifier using the training data provided in `facedata.png`. Extend the skeleton code in `FaceClassify.m`, which uses detections from the Viola-Jones face detector [VJ01]. Your code should compute euclidean distances between the Viola-Jones faces and each face in the database, and return the category of the face with the smallest distance. For efficiency, you should resize the images to 32×32 greyscale for this part. What is the success rate of the simple nearest-neighbour classifier?

Req #4: Improved Classifier [25 marks]

In this part you will try out your own ideas to improve the face classifier. You can experiment with feature extraction, more advanced classifiers, or a combination of both. For example, you could use a Support Vector Machines (SVM) [CL11], Logistic Regression, or another classifier of your choice. You should clearly describe the methods you choose and the results of your experiments.

To tune your classifier, allocate a *training set* and a *validation set* from the face database. For example, you could use 24 images per person as training and 8 for validation. You can then tune the parameters of your classifier/features over this validation set. For example, with `libsvm` [CL11] you can experiment with the kernel function (`−t` option) and associated kernel parameters (e.g., `−g gamma` for the RBF kernel), and the misclassification cost parameter (`−c`).

When you are happy with the results on the validation set, run your final classifier on the G20 group test image and report the results.

References

[CL11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[Sze11] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2011.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.