ELEC 474

Project – Fake Face Detection

Project – Fake Face Detection

Ian Maquignaz, Nic Merz, Yangzheng Wu and Michael Greenspan

Contents

1. Background

Machine vision based face recognition is becoming an increasingly popular method of biometrically authenticating users, whether it's a cell phone, personal computer, building access, or (eventually) access to one's bank account. One of the inherent risks of using such authentication is the potential for abuse by cyber thieves attempting to break into someone's account. The simplest way to attempt this would be to present a picture of someone's face to the camera, and thereby trick the system to believe that the real person is present before it.

2. Objective

The objective of this project is to develop a machine vision system that can differentiate between a set of images of a real face, and a set of images of a picture of a face.

To accomplish this, you will develop a machine vision system that takes as input a set of at least three images of a face (e.g. your face). These images can be acquired by a webcam, cell phone, or any other camera that you have available. Each image will have a slightly different pose of the face (e.g. look forward, look left, look right, etc.). One set of images will be of your actual face, and another set of images will be of a picture of your face, as illustrated in Figures 1 and 2.


a) left


b) front


c) right

Figure 1 – Real Image Set


a) left


b) front


c) right

Figure 2 – Fake Image Set

3. Rules and Constraints

3.1 Method

You can use whatever method you like, under the following constraints:

- You can use any existing OpenCV methods, in either the core or contrib repositories;
- All of the code that you develop has to be your own, i.e. other than OpenCV, matplotlib and numpy, you cannot use any external code or python libraries.
- You cannot use any external data sets for training the system.

3.2 Cameras

You may use only intensity (RGB) cameras, either monocular or stereo. Do not use range sensors or RGBD cameras (e.g. Microsoft Kinect or Intel Realsense).

3.3 Groups

This project is meant to be done individually, or in groups of 2. Groups larger than 2 are not allowed.

3.4 Datasets

Test the effectiveness of your system on the provided data set. When you submit your code, it will be tested against a different data set of a different face. It is therefore a good idea to test the generality of your method by trying it on one or more different data sets, other than the provided data set (e.g. use your face & possibly more). To do so, you should supplement the provided data set with at least 10 sets of real and 10 sets of fake images. If each set has 3 images (front, left and right), then your training dataset will contain 60 images, i.e. 30 real and 30 fake.

3.5 Evaluation Metrics

Your code should classify each image set as either "real" or "fake", and, while running through the training data, track and report the four possible classification outcomes as a confusion matrix:

- True Positive (TP): A real image set is classified as real;
- False Positive (FP): A false image set is classified as real;
- True Negative (TN): A false image set is classified as false;
- False Negative (FN): A real image set is classified as false.

The Precision and Recall metrics of your system are then calculated as:

$$\text{Precision} = N_{TP}/(N_{TP} + N_{FP})$$

$$\text{Recall} = N_{TP}/(N_{TP} + N_{FN})$$

where N_{TP} is the number of images that were true positives, etc. The confusion matrix is then:

N_{TN}	N_{FP}
N_{FN}	N_{TP}

Table 1 - Confusion Matrix

4. Hints

4.1 Haar Cascades

There's a good set of methods in OpenCV to detect the bounding box (Region of Interest) of faces, using Haar Cascades, which can be helpful. Take a look at:

https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

4.2 Plane Face

The fake faces (e.g. printed on a piece of paper, or displayed from a cell phone or tablet) are planar surfaces. Real faces are not planar.

5. Deliverables

The deliverables for this project are a report, and the code. For privacy reasons, you are not required to submit the additional dataset that you collected.

5.1 Report

Write a brief (no more than 5 page) report describing your solution and results. Include the following information:

- Your name and your student number. If you're working in a pair, include both partners' names and student numbers. (The order that they appear does not matter.)

- A declaration of originality, indicating the extent that you are the author of the submitted solution. For example, if you are working alone, you might declare that you developed the solution and wrote 100% of the submitted source code. Alternately, if you're working in a pair,

then you might declare that the two partners jointly developed the code, with partner A focussing on step 1 and partner B focussing on step 2, (e.g. a 75%/25% split).

- A description of the method. Describe the rationale for the approach, and why you believe it should work well.

- A description of the implementation. Include a description of any specialized data structures and algorithms that you developed and applied, any key OpenCV methods that you relied upon, as well as any further relevant details of the developed software (optimization methods, data types of interest, etc.).

- A description of the tests that you executed and the results that you obtained. Report a separate confusion matrix, Precision and Recall value (see Section 3.5) for each dataset that you tested. Also include the time performance of the method (e.g. how long did it take to execute for each tested image set?).

- A discussion of the correctness and effectiveness of your solution. Include here a declaration of the success of your system, i.e. the extent that you considered your solution to solve the stated task. Include any limitations that you observed, and indicate if there were any computationally expensive bottlenecks to the processing

- A proposal on how you would improve your solution, if you had more time to work on it (e.g. 3 months).

5.2 Code

Your code should include a .zip of:

- The .ipynb or .py files that automatically (and without user prompts) executes your experiment on the entire dataset, and outputs the Confusion Matrix, Precision and Recall scores.

- The provided dataset, correctly linked to your code, and with images stored respectively in folders labelled “/real” and “/fake” off of the main project directory. This will allow your code to be easily executed on a novel (different) dataset, by just renaming these two folders accordingly.

The marking rubric is as follows:

Section	mark
Approach	5
Implementation	3
Experimentation	2
Report	4
Correct submission format	1
Total:	**15**