Face Detection Project

Socaci Radu Andrei

December 2019

Contents

1 Bibliographic Study

Face Detection is an AI-based computer technology that can identify and locate the presence of human faces in digital photos and videos. It can be regarded as a special case of object-class detection, where the task is to find the locations and specify the sizes of all the objects that belong to a given class – in this case, faces – within a specific image or images.Face detection applications use algorithms that determine whether images are positive images (i.e. images with a face) or negative images (i.e. images without a face). To be able to do this accurately, the algorithms must be trained on huge data-sets containing hundreds of thousands of face images and non-face images. In the past, these algorithms were machine-learning based, and were heavily affected by factors such as extreme head poses (where the head is rotated far to one side or tilted far up or far down, for example) and varying lighting conditions. Today, however, we can use deep learning methods to carry out accurate face detection in a wide range of scenarios. [2]

OpenCV (Open Source Computer Vision Library) is mostly used for image processing (computer vision). OpenCV is a programming functions developed library, and is available for multiple programming languages, such as Python, C++, etc.

Haar Cascade is a machine learning object detection algorithm proposed by Paul Viola and Michael Jones in their paper "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. The cascade can then be used to detect objects in other images.

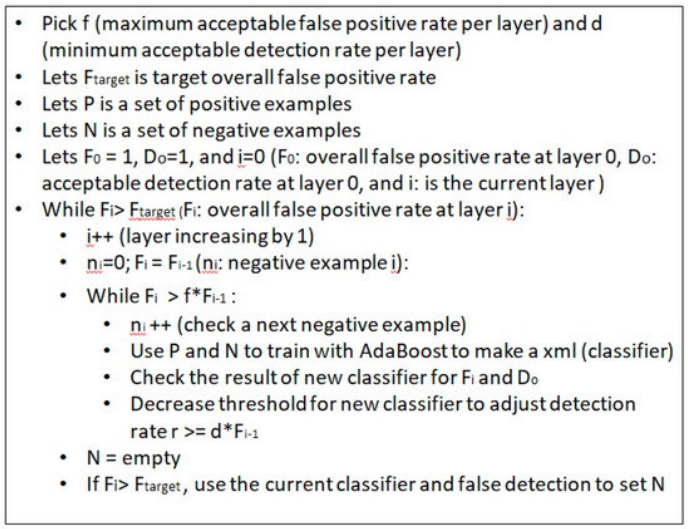

Figure 1: Haar Cascade Algorithm Pseudocode

OpenCV offers pre-trained Haar cascade algorithms, organized into categories (faces, eyes and so forth), depending on the images they have been trained on. [1] The pseudocode is given in figure 1.

2 Analysis and Design

The application enables the user to start a video stream using the webcam. Each frame of this video stream will be processed and the face and eyes of the user will be highlighted (a rectangle will be drawn around them). This entire process works in real time, having a small overhead.

The application can be extended by using already trained XML classifiers from the internet.

The application was implemented using Python 3.8 and OpenCV. Since OpenCV is a "pip install" away, there was not much configuration to be done.

3 Implementation

The implementation is pretty straight forward. Two XML classifiers were used, one for the face detection part and the other for the eyes detection part. Each frame of the video stream is converted to grayscale (most computation in OpenCV are done in grayscale). There are two classifiers needed: one for the face, initialized with *haarcascade_frontalface_default.xml* and one for the eyes, initialized with *haarcascade_eye.xml*.

Firstly, the position of all faces in the frame is determined and highlighted.

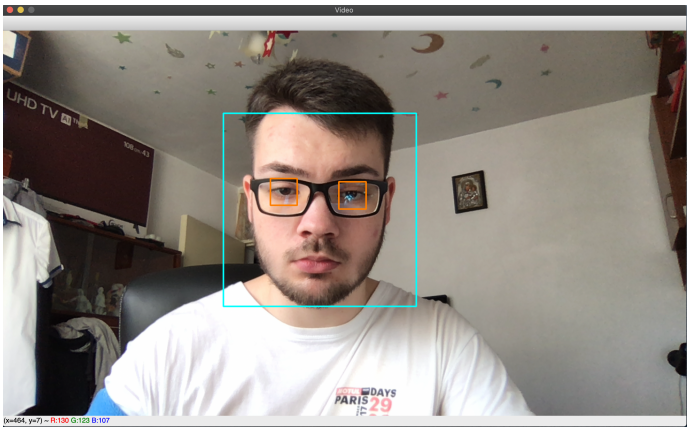

Figure 2: Real Time Face and Features Detection

For each face, the image is cropped (we only take into consideration each rectangle as a standalone picture) and eye detection is applied on these newly obtained pictures. The position of the eyes is also highlighted with rectangles.

The frame is than modified to include all these newly drawn rectangles and displayed on the user's screen. In order to stop the program, the ESC key needs to be pressed.

The application is run directly from the terminal window, and may ask for permission to use the webcam on first use. This is a basic application, but it can be extended to be useful in real life scenarios (for example cropping the picture from the ID card and matching it against a snapshot taken in real time using the webcam to verify the user's identity)

4 Testing and Validation

The testing was done by providing live stream from the web camera using different angles and light intensities. The system can detect faces and features well, but the footage received on the screen is quite laggy and could be improved. In general, faces are detected and highlighted with a really high success rate. However, the eye detection feature is not so consistent and may sometimes fail. This can be improved using better classifiers and/or sensitivity.

References

[1] Mdpi.

[2] Face detection - sightcorp. 2019.