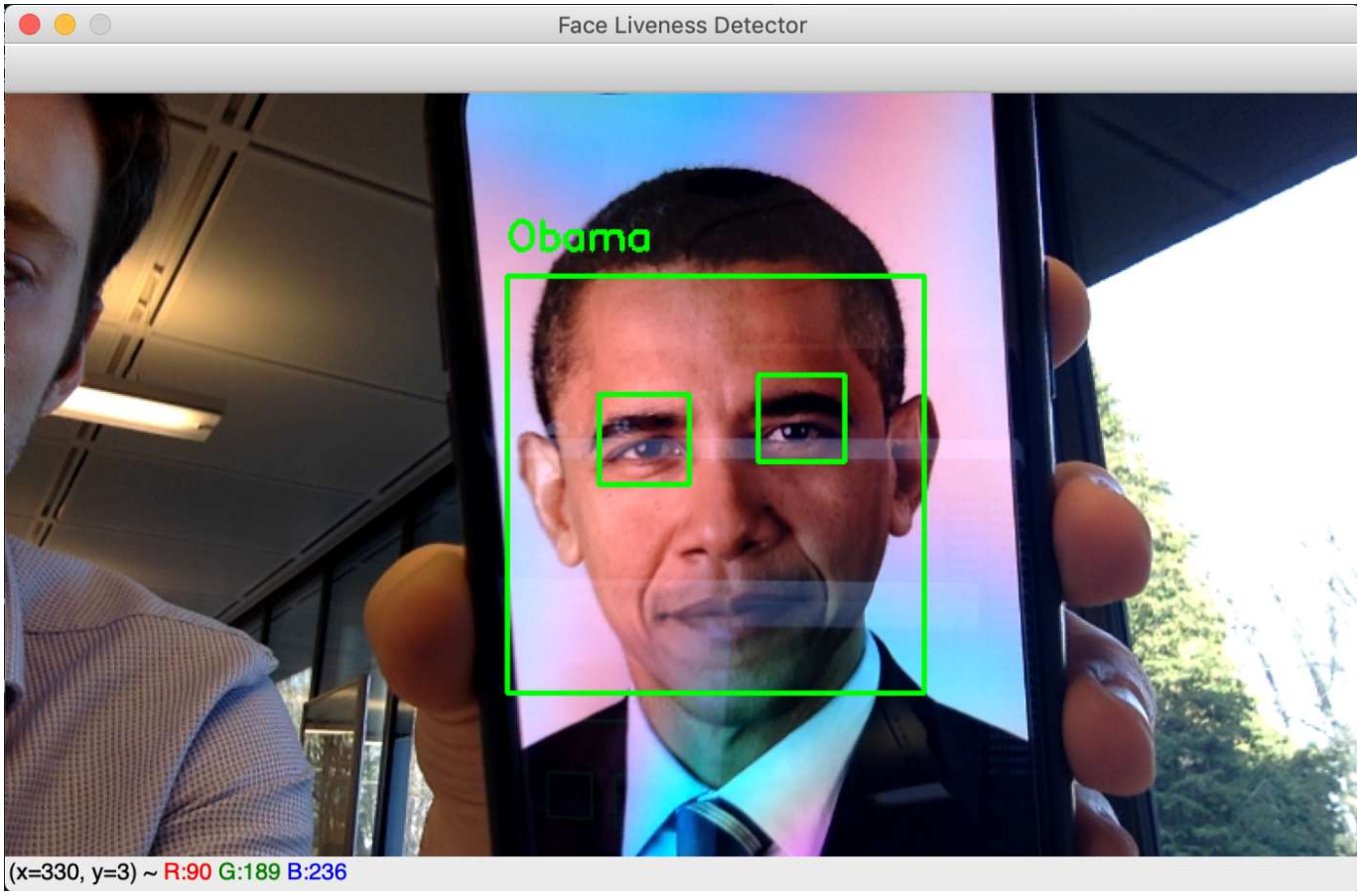
Real-time face liveness detection with Python, Keras and OpenCV

Jordan Van Eetveldt Follow
Mar 4, 2019 · 5 min read ★

Most facial recognition algorithms you find on the internet and research papers suffer from photo attacks. These methods work really well at detecting and recognizing faces on images, videos and video streams from webcam. However they can't distinguish between real life faces and faces on a photo. This inability to recognize faces is due to the fact that these algorithms work on 2D frames.

Now let's imagine we want to implement a facial recognition door opener. The system would work well to distinguish between known faces and unknown faces so that only authorized persons have access. Nonetheless, it would be easy for an ill-intentioned



Example of photo attack with Obama face ✕

This article objective is to implement an eye-blink detection-based face liveness detection algorithm to thwart photo attacks. The algorithm works in real time through a webcam and displays the person's name only if they blinked. In layman's terms, the program runs as follows:

1. Detect faces in each frame generated by the webcam.
2. For each detected face, detect eyes.
3. For each detected eyes, detect if eyes are open or closed.
4. If at some point it was detected that the eyes were open then closed then open, we conclude the person has blinked and the program displays its name (in the case of a facial recognition door opener, we would authorize the person to enter).

most useful. The *face_locations* method can detect faces using two methods*: Histrogram of oriented Gradients (HoG)* and C*onvolutional Neural Network (CNN)*. Due to time constraints the *HoG* method was chosen. The *face_encodings* function is a pre-trained Convolutional Neural Network able to encode an image into a vector of 128 features. This embedding vector should represent enough information to distinguish between two different persons. Finally, the *compare_faces* computes the distance between two embedding vectors. It will allow the algorithm to recognize face extracted from a webcam frame and compare its embedding vector with all encoded faces in our dataset. The closest vectors should correspond to the same person.

1. Known face dataset encoding

In my case, the algorithm is able to recognize myself and Barack Obama. I selected around 10 pictures of each. Below is the code to process and encode our database of known faces.

Now that we know the encodings for each person we want to recognize, we can try to identify and recognize faces through a webcam. However, before moving to this part, we need to distinguish between a face photo and a living person's face.

2. Face liveness detection

As a reminder, the goal is to detect an open-closed-open eye pattern at some point. I trained a Convolutional Neural Network to classify whether an eye is closed or open. The chosen model is the LeNet-5 which has been trained on the *Closed Eyes In The Wild (CEW)* dataset. It is composed of around 4800 eye images in size 24x24.

When evaluating the model, I reached **94%** accuracy.

Each time we detect an eye, we predict its status using our model, and we keep track of the eyes status for each person. Therefore, it becomes really easy to detect an eye blinking thanks to the function below, which tries to find a closed-open-closed pattern in the eyes status history.

We almost have all the elements to set up our "real"-face recognition algorithm. We just need a way to detect faces and eyes in real-time. I used openCV pre-trained Haar-cascade classifier to perfom these tasks. For more information about faces and eyes detection with Haar-cascade I highly recommend you to read this great article from openCV.

The function above is the code used for detecting and recognizing real faces. It takes in arguments:

- model: our open/closed eyes classifier
- video_capture: a stream video
- face_detector: a Haar-cascade face classifier. I used *haarcascade_frontalface_alt.xml*
- open_eyes_detector: a Haar-cascade open eye classifier. I used *haarcascade_eye_tree_eyeglasses.xml*
- left_eye_detector: a Haar-cascade left eye classifier. I used *haarcascade_lefteye_2splits.xml* which can detect open or closed eyes.
- right_eye_detector: a Haar-cascade right eye classifier. I used *haarcascade_righteye_2splits.xml* which can detect open or closed eyes.
- data: a dictionary of known encodings and known names
- eyes_detected: a dictionary containing for each name the eyes status history.

At **Lines 2–4** we grab a frame from the webcam stream and we resize it to speed up computations. At **line 10** we detect faces from the frame , then at **line 21**, we encode them into a 128-d vector. In **line 23–38** we compare this vector with the known face encodings and we determine the person's name by counting the number of matches. The one with the biggest number of matches is selected. Starting at **line 45** we try to detect eyes into face boxes. First, we try to detect open eyes with the *open_eye_detector*. If the

into left and right side for the respective detectors to be classified. Beginning at **line 92,** the eye part is extracted and the trained model predicts whether the eyes are closed. If one closed eye is detected, then both eyes are predicted to be closed and a '0' is added to the eyes status history. Otherwise it's concluded that the eyes are open. Finally at **line 110** the *isBlinking()* function is used to detect eye blinking and if the person has blinked the name is displayed. The whole code can be found on my github account.

Photo attack thwarted using eye-blink detection ✔

References

- https://docs.opencv.org/3.4.3/d7/d8b/tutorial_py_face_detection.html
- https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

Machine Learning Face Recognition Face Liveness Python Opencv