Removal of Background in a Selfie (for Face ID Picture) using face detection technique

1. **Problem Definition**

The growing popularity of smartphones and other handy devices with advanced camera technology, allows us to take high-quality pictures that we can use as ID pictures or even just share on social platforms like Facebook, Instagram, etc. However, **GOLA: these pictures may need to get processed before we start using them as ID pictures such as removing the background or any other unwanted objects that showing on our picture.** Therefore, this project addresses this issue where we want to extract our faces from the picture.

2. **Suggested Solution**

Extracting the faces/objects from the pictures is the first step to start developing a reliable application that can take your capture and transform it into a high-quality ID picture. Therefore, this project suggests using face detection algorithms in order to extract the faces from the picture.

3. **Project Scope**

Image detection is one of the most widespread topics in computer vision nowadays. One of the popular image detection applications is face detection where a computer can process the image and detects faces in it. They are found in different fields like healthcare, marketing, transportation, or even in personal use and that is what the project will propose. The scope of the project is building a program that can be able to detect the faces from the picture.

4. **Methodology**

using a pre-trained haar cascade model that's provided by the OpenCV library

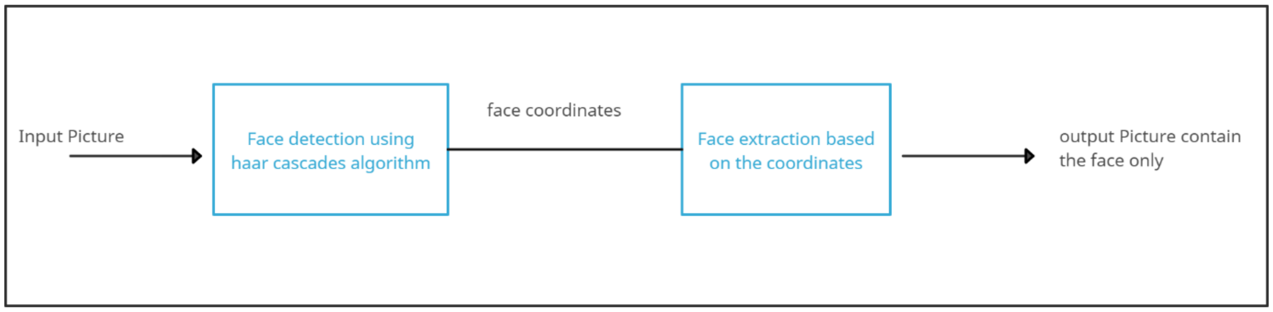
5. **Project Objectives**

Reviewing similar applications and articles shows that in order to remove the background or any unwanted objects from the picture while you want to keep the faces you have to do first the face detection process and then perform an extraction algorithm. The main functionalities of the proposed project include the following:

1. Take pictures an input.
2. Detect all faces in the loaded picture automatically using an image detection algorithm.
3. Extract the faces from the picture.
4. Apply the faces on a plain background.

However, in this project, I will focus on the face detection stage.

6. Proposed Design


Input Picture
Face detection using haar cascades algorithm
face coordinates
Face extraction based on the coordinates
output Picture contain the face only

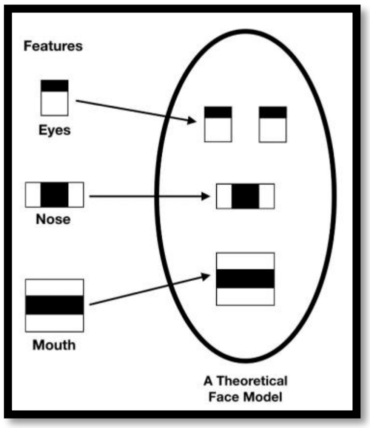
7. Finding

Discussion, analysis, and More Detecting Faces with Haar Cascade: Limitations In terms of deep object detection, Haar Cascade cannot work. There are a few Limitations of the Haar Cascade Algorithm. Compared to modern object detectors, this detector has a lower accuracy, The detection of false positives is high. It is necessary to manually adjust parameters and it is not easy to train Hamar cascades for custom objects.

8. Implementation

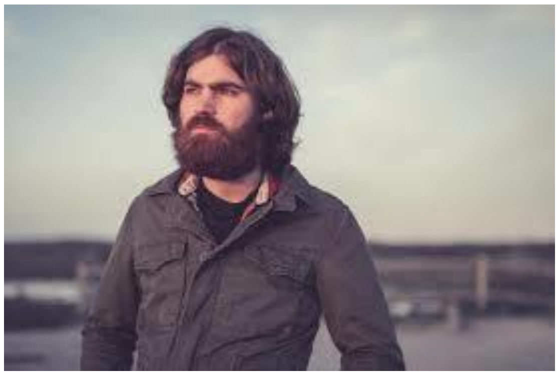
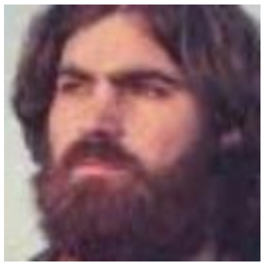
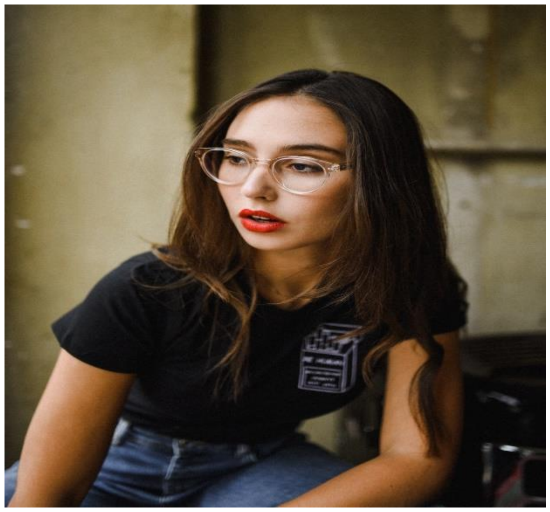
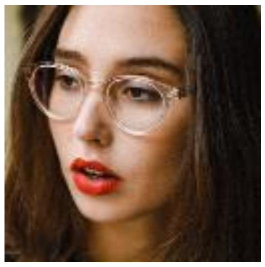
8.1. Face detection algorithm

A computer program that analyses a photo into two results: positive photo (photo has faced) or negative photo (photo has non-face) is called a classifier. OpenCV provides pre-trained and ready-to-use face detection classifiers one of them known as the Haar-like features classifier is a machine learning-based approach, an algorithm generated by Paul Viola and Michael Jones (ramviyas, 2017).



Face detection using OpenCV

8.2. Result

Input	Output
	
	

***I attached the video (of running code) in a separate file**

9. Conclusion

The Haar Cascade Detection algorithm is one of the most powerful and oldest face detection algorithms available. It had been there for a long, long time before Deep Learning became famous. There was no limitation to Haar Features because they could also detect eyes, lips, license plates, and more.

Code:

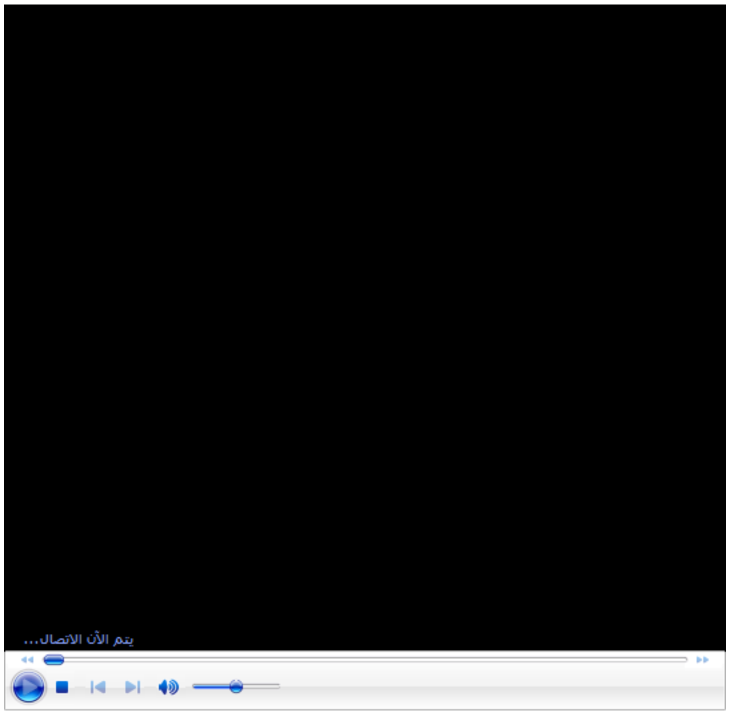
```
#import the libraries
import numpy
import cv2
import glob
from google.colab.patches import cv2_imshow
#import the model using openCV library
face_cascade=cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_fr
ontalface_default.xml")

img = cv2.imread("/content/Image2.jpg")
gray_img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces=face_cascade.detectMultiScale(gray_img, scaleFactor=1.3,minNeighbors
=5)
for x, y, w, h in faces:
    #img=cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,0),3)
    roi_color = img[y:y+h+100, x:x+w+100]

resized=cv2.resize(roi_color, (128,128))
cv2.imwrite("/content/After.jpg", resized)
cv2_imshow(resized)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Additional demo



Source

References

Ramviyas (2017). ”Face Detection using Haar Cascades”. Aug 4, 2017. Retrieved from https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html