clasificadores en cascada basados ​​en funciones de Haar es un método eficaz de detección de objetos.

Es un enfoque basado en el aprendizaje automático en el que La función en cascada se entrena a partir de muchas imágenes positivas y negativas. Luego se utiliza para detectar objetos en otras imágenes. Inicialmente, el algoritmo necesita muchas imágenes positivas (imágenes de rostros) e imágenes negativas (imágenes sin rostros) para entrenar al clasificador.

from \_\_future\_\_ import print\_function  
import cv2 as cv  
import argparse  
def detectAndDisplay(frame):  
 frame\_gray = cv.cvtColor(frame, cv.COLOR\_BGR2GRAY)  
 frame\_gray = cv.equalizeHist(frame\_gray)  
 #-- Detect faces  
 faces = face\_cascade.detectMultiScale(frame\_gray)  
 for (x,y,w,h) in faces:  
 center = (x + w//2, y + h//2)  
 frame = cv.ellipse(frame, center, (w//2, h//2), 0, 0, 360, (255, 0, 255), 4)  
 faceROI = frame\_gray[y:y+h,x:x+w]  
 #-- In each face, detect eyes  
 eyes = eyes\_cascade.detectMultiScale(faceROI)  
 for (x2,y2,w2,h2) in eyes:  
 eye\_center = (x + x2 + w2//2, y + y2 + h2//2)  
 radius = int(round((w2 + h2)\*0.25))  
 frame = cv.circle(frame, eye\_center, radius, (255, 0, 0 ), 4)  
 cv.imshow('Capture - Face detection', frame)  
parser = argparse.ArgumentParser(description='Code for Cascade Classifier tutorial.')  
parser.add\_argument('--face\_cascade', help='Path to face cascade.', default='data/haarcascades/haarcascade\_frontalface\_alt.xml')  
parser.add\_argument('--eyes\_cascade', help='Path to eyes cascade.', default='data/haarcascades/haarcascade\_eye\_tree\_eyeglasses.xml')  
parser.add\_argument('--camera', help='Camera divide number.', type=int, default=0)  
args = parser.parse\_args()  
face\_cascade\_name = args.face\_cascade  
eyes\_cascade\_name = args.eyes\_cascade  
face\_cascade = cv.CascadeClassifier()  
eyes\_cascade = cv.CascadeClassifier()  
#-- 1. Load the cascades  
if not face\_cascade.load(cv.samples.findFile(face\_cascade\_name)):  
 print('--(!)Error loading face cascade')  
 exit(0)  
if not eyes\_cascade.load(cv.samples.findFile(eyes\_cascade\_name)):  
 print('--(!)Error loading eyes cascade')  
 exit(0)  
camera\_device = args.camera  
#-- 2. Read the video stream  
cap = cv.VideoCapture(camera\_device)  
if not cap.isOpened:  
 print('--(!)Error opening video capture')  
 exit(0)  
while True:  
 ret, frame = cap.read()  
 if frame is None:  
 print('--(!) No captured frame -- Break!')  
 break  
 detectAndDisplay(frame)  
 if cv.waitKey(10) == 27:  
 break