EXPT.NO:9

DATE: 23.4.2024

IMPLEMENTATION OF IRIS DETECTION USING PYTHON AND MATLAB

AIM:

To implement the Iris detection using PYTHON and MATLAB.

SOFTWARE USED:

- PyCharm and Webcam
- MATLAB 2014a

THEORY:

Iris detection is a biometric technology that involves identifying individuals based on the unique patterns found in their iris, the colored part of the eye that surrounds the pupil. The algorithm uses Haar Cascade method and image segmentation for detection of the Iris.

Unique Patterns:

The iris contains unique patterns that are formed randomly during the embryonic stage and remain stable throughout a person's life. These patterns include furrows, freckles, and other features that are used for identification.

Iris Recognition:

Iris detection typically involves capturing an image of the iris using a specialized camera. The image is then processed to extract the iris pattern, which is often represented as a series of mathematical values.

Feature Extraction:

Feature extraction algorithms are used to identify key characteristics of the iris pattern, such as the location and shape of the pupil, the structure of the iris fibers, and the presence of any unique features like freckles or furrows.

Template Creation:

Based on the extracted features, a unique template or code is created to represent the iris pattern. This template is typically stored in a database for future comparison.

Matching:

When a person needs to be identified, a new image of the iris is captured and processed to create a template. This template is then compared to the templates stored in the database to find a match.

Webcam:

Iris detection using a webcam involves capturing images of the iris, processing them to extract unique patterns, and using these patterns for identification. While webcam quality may affect accuracy, it's still possible to achieve reliable results with proper image capture and processing.

Application:

Iris detection is used in various applications, including access control, border security, and identity verification. It is considered one of the most accurate biometric technologies due to the complexity and uniqueness of the iris patterns.

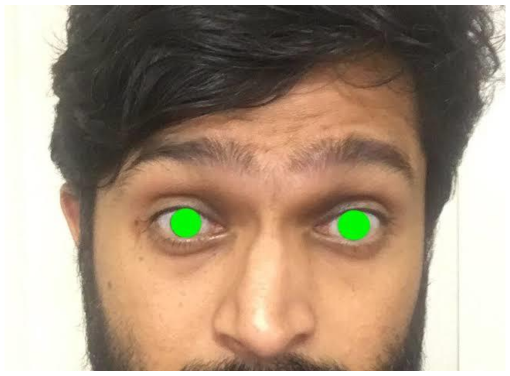
PROGRAM:

```
import cv2
import numpy as np
##Import xml files for face and eye detection
eye = cv2.CascadeClassifier('haarcascade_eye.xml')
face = cv2.CascadeClassifier('haarcascade_frontalface_alt.xml')
Kernal = np.ones((3, 3), np.uint8)      #Declare kernal for morphology
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)       #Set camera resolution
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
while 1:
    ret, frame = cap.read()                   #Read image frame
    frame = cv2.flip(frame, +1)
  #Convert image to grayscale
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    detect_face = face.detectMultiScale(gray, 1.2, 1)  #Detect Face
    detect_eye = eye.detectMultiScale(gray, 1.2, 1)
    for(face_x, face_y, face_z, face_h) in detect_face:
        img2 = gray[face_y:face_y+face_h, face_x:face_x+face_z]
        detect_eye = eye.detectMultiScale(img2, 1.2, 1)
        for (eye_x, eye_y, eye_z, eye_h) in detect_eye:
            # cv2.rectangle(frame, (face_x+eye_x, face_y+eye_y), (face_x+eye_x+eye_z,
face_y+eye_y+eye_h),
            #               (0, 255, 0), 2)
            eye1 = gray[face_y+eye_y:face_y+eye_y+eye_h,
face_x+eye_x:face_x+eye_x+eye_z]
            ret, binary = cv2.threshold(eye1, 60, 255, cv2.THRESH_BINARY_INV)
            # cv2.imshow('Binary', binary)
            width, height = binary.shape
            binary = binary[int(0.4 * height):height, :]    #Crop top 40%of the image
            opening = cv2.morphologyEx(binary, cv2.MORPH_OPEN, Kernal)
            dilate = cv2.morphologyEx(opening, cv2.MORPH_DILATE, Kernal)
            # cv2.imshow('Dilated', dilate)
            contours, hierarchy = cv2.findContours(dilate, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)
            if len(contours) != 0:
                cnt = contours[0]
                M1 = cv2.moments(cnt)
                Cx1 = int(M1['m10'] / M1['m00'])       #Find center of the contour
                Cy1 = int(M1['m01'] / M1['m00'])
                croppedImagePixelLength = int(0.4*height)
                center1 = (int(Cx1+face_x+eye_x), int(Cy1+face_y + eye_y +
croppedImagePixelLength))              #Center coordinates
                cv2.circle(frame, center1, 2, (0, 255, 0), 2)
    if not ret:                               #If frame is not read then exit
        break
    if cv2.waitKey(1) == ord('s'):          #While loop exit condition
        break
    cv2.imshow('Frame Image', frame)        #Show original Image
cap.release()
cv2.destroyAllWindows()
```

MATLAB CODE:

```
clc
clear all
i=imread('images(1).jpeg');
i2=rgb2gray(i);
imshow(12)
iedge=edge(i2,'canny',0.5);
imshow(iedge)
%d=imdistline:
[c, r]=imfindcircles(iedge,[30, 50]);
subplot  211
imshow(i)
hold on
viscircles(c ,r,'color','r');
title('Extract  the Iris of eye')
%%
[c1,r1]=imfindcircles(iedge,[6,18]);
subplot  212
imshow(i)
hold on
viscircles(c ,r,'color','r');
title('Extract  the pupil of eye')
```

OUTPUT:



RESULT:

Thus the implementation of Iris detection is done through Python and matlab software and the output is verified.