

PHOTO CAPTURE

Presented by V B V Satish
(Reg No: 37110848)
Guided by Dr. S. Prince Mary

OBJECTIVE



THIS PROJECT AIMS TO DETECT THE NUMBER OF FACES AND CAPTURE THE IMAGES OF THE FACES THAT COMES BEFORE THE CAMERA.



THE OBJECTIVE OF OUR PROJECT IS TO DESIGN SOFTWARE THAT CAN DETECT HUMAN FACES.



IMPLEMENT THE PROJECT USING PYTHON WITH NECESSARY LIBRARIES

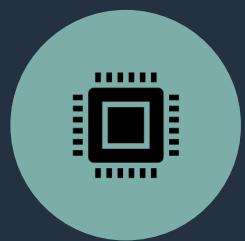
SOFTWARE REQUIREMENTS



ANY OS



HAAR CASCades
DATA FILE



RAM (MINIMUM
4GB)



ANACONDA
NAVIGATOR



OPENCV-
PYTHON

OpenCV

- CV2 OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library.
- It will also support Windows, Linux, Mac OS and even the Android operating systems.
- It supports a some of the programming languages namely: Python, Java, C and C++.
- Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.
- We are going to use the "DetectMultiscale" module from OpenCV.

Features of OpenCV Library



Read and write images.



Capture and save videos.



Process images (filter, transform)



Detect specific objects such as faces, eyes in the videos or images.



Analyse the video, i.e. estimate the motion in it, subtract the background, and track objects in it.

Parametres

Parameter Name	Description
Cascade	It can be loaded from XML or YAML file using Load().
Image	Matrix of the type CV_8U containing an image where objects are detected
Objects	Vector of rectangles where each rectangle contains the detected object the rectangles may be partially outside the original image.
numDetections	Parameter specifying how much the image size is reduced at each image scale
ScaleFactor	Parameter specifying how much the image size is reduced at each image scale
minSize	Minimum possible object size. Objects smaller than that are ignored.
maxSize	Maximum possible object size. Objects larger than that are ignored.

Code Snippet

The screenshot shows the Spyder Python IDE interface. The main window displays a Python script named `photocapture.py`. The code uses OpenCV's Haar cascade classifiers to detect faces and eyes in a video feed from the webcam. It prints the number of faces detected and saves each face to a file. The `detect` function processes a grayscale frame, finds faces, and then processes a ROI (Region of Interest) for eye detection. The `while True` loop reads frames from the camera, applies the `detect` function, and displays the resulting video.

```
5 import cv2
6
7 # Loading the cascades
8 face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
9 eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')
10
11 # Defining a function that will do the detections
12 def detect(gray, frame):
13     faces = face_cascade.detectMultiScale(gray, 1.3, 5,0)
14     for (x, y, w, h) in faces:
15         print (faces.shape)
16         print ("Number of faces detected: " + str(faces.shape[0]))
17         print("Data Found")
18         cv2.putText(frame, "Number of faces detected: " + str(faces.shape[0]), (10, 30),
19                     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
20         cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
21         roi_gray = gray[y:y+h, x:x+w]
22         roi_color = frame[y:y+h, x:x+w]
23         FaceFileName = "face_" + str(y) + ".jpg"
24         cv2.imwrite(FaceFileName, roi_color)
25         eyes = eye_cascade.detectMultiScale(roi_gray, 1.1, 3)
26         for (ex, ey, ew, eh) in eyes:
27             cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
28     return frame
29
30 # Doing some Face Recognition with the webcam
31 photo_capture = cv2.VideoCapture(0)
32 while True:
33     _,frame = photo_capture.read()
34     gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
35     canvas = detect(gray, frame)
36     cv2.imshow('Video', canvas)
37     if cv2.waitKey(1) & 0xFF == ord('q'):
38         break
39 photo_capture.release()
40 cv2.destroyAllWindows()
```

The right side of the interface shows the `Console` and `IPython console` panes. The `Console` pane displays the output of the script, which includes the number of faces detected and the status of the `Data Found` variable. The `IPython console` pane is currently empty. A floating `Usage` help box provides information on how to use the `Ctrl+I` keyboard shortcut to get help for objects.

Results and Discussion

The image displays two side-by-side screenshots of the Spyder Python IDE interface, illustrating the performance of a face detection script named `photocapture.py`.

Screenshot 1: Shows a video feed of a single person with a mustache. A single blue bounding box surrounds the person's head, and three green bounding boxes highlight the eyes. Red text at the top of the video window reads "Number of faces detected: 1". The code in the editor shows a loop that processes frames from a camera.

```
# -*- # Face detection -*-  
# Import  
import cv2  
# Load  
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')  
# Detect  
def detect(gray, frame):  
    # Do something  
    photo_capture = cv2.VideoCapture(0)  
    while True:  
        _,frame = photo_capture.read()  
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
        canvas = detect(gray, frame)  
        cv2.imshow('Video', canvas)
```

Screenshot 2: Shows a video feed of three people. Three blue bounding boxes surround the heads of the three individuals. Red text at the top of the video window reads "Number of faces detected: 3". The code in the editor is identical to the first screenshot.

```
# -*- # Face detection -*-  
# Import  
import cv2  
# Load  
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')  
# Detect  
def detect(gray, frame):  
    # Do something  
    photo_capture = cv2.VideoCapture(0)  
    while True:  
        _,frame = photo_capture.read()  
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
        canvas = detect(gray, frame)  
        cv2.imshow('Video', canvas)
```

The right side of each screenshot shows the Spyder interface with the console output, which displays the number of faces found in each frame, and a usage help panel.

References

Git Repo Link:

<https://github.com/satish2902/photocapture>