

Drone Facial Recognition System

NAME – OJAS GUPTA

MENTOR – DR.VIVEK KANHANGAD

TITLE – FACIAL RECOGNISATION FOR DRONE

PROGRAMMING LANGUAGE - PYTHON

EXTERNAL RESOURCES - [haarcascade_frontalface_default.xml](#)

Introduction

Drones with facial recognition is a concept that combines the features of both face recognition and drone surveillance. Developing drones with robust facial recognition that could identify people from a crowded place was a hurdle for security agencies and researchers.

Face recognition incorporated in drones clearly creates a unique and valuable combination, mainly in the security arena. The use of this powerful application will become widespread as more decision makers will realise its advantages. As technology continues to progress, we can also expect to see an increase in the use of drones in the civic-commercial market.

Application

Image database investigations: Searching image databases of licensed drivers, benefit recipients, missing children, immigrants and police bookings.

General identity verification: Electoral registration, banking, electronic commerce, identifying newborns, national IDs, passports, employee IDs.

Surveillance: Like security applications in public places, surveillance by face recognition systems has a low user satisfaction level, if not lower.

Security: Today more than ever, security is a primary concern at airports and for airline staff office and passengers. Airport protection systems that use face recognition technology have been implemented at many airports around the world.

```

import cv2
import numpy as np

w, h = 360, 240
fbRange=[6200,6800]
pid = [0.4, 0.4, 0]
pError = 0

def findFace(img):    # function to detect and track face and we have
to find it in an image so we are passing that img
    faceCascade =
cv2.CascadeClassifier("Resources/haarcascade_frontalface_default.xml")
# here we are referring to haarcascade file for object detection using
parameter and methods that is available on open cv
    imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(imgGray,1.2,8)

    myFaceListC = []
    myFaceArea = []

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,225),2)
        cx = x + w//2
        cy = y + h//2
        area = w*h
        cv2.circle(img ,(cx,cy),5,(0,255,0),cv2.FILLED)
        myFaceListC.append([cx,cy])
        myFaceArea.append(area)

    if len(myFaceArea) != 0:
        i = myFaceArea.index(max(myFaceArea))
        return img, [myFaceListC[i], myFaceArea[i]]
    else:
        return img, [[0, 0], 0]

def trackFace(info, w, pid, pError):
    area = info[1]

    x, y = info[0]

    fb = 0

    error = x - w // 2

    speed = pid[0] * error + pid[1] * (error - pError)

```

```

speed = int(np.clip(speed, -100, 100))
if area > fbRange[0] and area < fbRange[1]:
    fb = 0
elif area > fbRange[1]:
    fb = -20
elif area < fbRange[0] and area != 0:
    fb = 20
if x == 0:
    speed = 0

    error = 0

print(speed, fb)

return error

```

```

cap = cv2.VideoCapture(0)
while True:
    _, img = cap.read()
    img = cv2.resize(img, (w, h))
    img, info = findFace(img)
    pError = trackFace(info, w, pid, pError)
    print("Center", info[0], "Area", info[1])
    cv2.imshow("Output", img)
    cv2.waitKey(10)

```

Code

```

import cv2
import numpy as np

```

```

w, h = 360, 240
fbRange=[6200,6800]
pid = [0.4, 0.4, 0]
pError = 0

```

```

def findFace(img):    # function to detect and track face and we have
to find it in an image so we are passing that img
    faceCascade =
cv2.CascadeClassifier("Resources/haarcascade_frontalface_default.xml")
# here we are refering to haarcascade file for object detection using

```

```

parameter and methods that is available on open cv
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(imgGray,1.2,8)

myFaceListC = []
myFaceArea = []

for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,225),2)
    cx = x + w//2
    cy = y + h//2
    area = w*h
    cv2.circle(img ,(cx,cy),5,(0,255,0),cv2.FILLED)
    myFaceListC.append([cx,cy])
    myFaceArea.append(area)

if len(myFaceArea) != 0:
    i = myFaceArea.index(max(myFaceArea))
    return img, [myFaceListC[i], myFaceArea[i]]
else:
    return img, [[0, 0], 0]

def trackFace(info, w, pid, pError):
    area = info[1]

    x, y = info[0]

    fb = 0

    error = x - w // 2

    speed = pid[0] * error + pid[1] * (error - pError)

    speed = int(np.clip(speed, -100, 100))
    if area > fbRange[0] and area < fbRange[1]:
        fb = 0
    elif area > fbRange[1]:
        fb = -20
    elif area < fbRange[0] and area != 0:
        fb = 20
    if x == 0:
        speed = 0

    error = 0

    print(speed, fb)

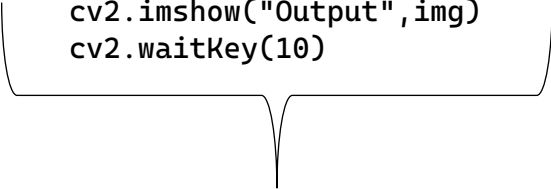
```

```
return error
```

```
cap = cv2.VideoCapture(0)
while True:
    _, img = cap.read()
    img = cv2.resize(img, (w, h))
    img, info = findFace(img)
    pError = trackFace(info, w, pid, pError)
    print("Center", info[0], "Area", info[1])
    cv2.imshow("Output",img)
    cv2.waitKey(10)
```


Code Explanation

```
cap = cv2.VideoCapture(0)
while True:
    b, img = cap.read()
    findface(img)
    cv2.imshow("Output",img)
    cv2.waitKey(10)
```



Using `cv2.VideoCapture()` to get a video capture object for the camera. Set up an infinite while loop and use the `read()` method to read the frames using the above created object. calling `cv2.imshow()` method to show the frames in the video. A `waitkey` is also called for the delay.

```
def findface(img);
    faceCascade =
cv2.CascadeClassifier("Resources/haarcascade_frontalface_default.xml")
#
    imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(imgGray,1.2,8)
```




OpenCV comes with lots of pre-trained classifiers. Those XML files can be loaded by cascadeClassifier method of the cv2 module. Here we are going to use haarcascade_frontalface_default.xml for detecting faces. Initially, the image is a three-layer image (i.e., RGB), So It is converted to a one-layer image (i.e., grayscale). This is done using the cv2.CascadeClassifier::detectMultiScale method, which returns boundary rectangles for the detected faces (i.e., x, y, w, h). It takes two parameters namely, scaleFactor and minNeighbors.

```
myFaceListC = []
myFaceArea = []

for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,225),2)
    cx = x + w//2
    cy = y + h//2
    area = w*h
    cv2.circle(img ,(cx,cy),5,(0,255,0),cv2.FILLED)
    myFaceListC.append([cx,cy])
    myFaceArea.append(area)

if len(myFaceArea) != 0:
    i = myFaceArea.index(max(myFaceArea))
    return img, [myFaceListC[i], myFaceArea[i]]
else:
    return img, [[0, 0], 0]
```



here we are collecting all the faces and the area of in the lists myFaceListC and myFaceArea. After declaring the for loop in faces which gives us parameters x , y , w , h and after we are have created the rectangle around the largest face we are calculating the centre point circle and filled it. Finally we are appending the list. We are checking whether the list empty or not , if it's not empty then we are calculating the index of the largest area and max value. Based on the area we are going to adjust our drone position foreword or backwards and the centre values will help in adjusting the rotation.

```
def trackFace(info, w, pid, pError):
    area = info[1]
```

```

x, y = info[0]

fb = 0

error = x - w // 2

speed = pid[0] * error + pid[1] * (error - pError)

speed = int(np.clip(speed, -100, 100))

if area > fbRange[0] and area < fbRange[1]:

    fb = 0

elif area > fbRange[1]:

    fb = -20

elif area < fbRange[0] and area != 0:

    fb = 20


if x == 0:
    speed = 0

    error = 0

print(speed, fb)

return error

```



For going forward and backward we are creating a range of [6200,6800]. In the first if section we are commanding drone to stay stationary if it's in the perfect zone , move away if it's too close and area is too big and get closer if object is too far.

The error declared in the function gives us the deviation from the centre. The value of Proportional integral derivative is set to [0.4,0.4,0]. By using Proportional and derivative we have calculated the speed and if we do not get any input the we have to stop for that we can set the value of speed and error to 0