import cv2

import time

import datetime

import ibm\_boto3

from ibm\_botocore.client import Config, ClientError

from ibmcloudant.cloudant\_v1 import CloudantV1

from ibmcloudant import CouchDbSessionAuthenticator

from ibm\_cloud\_sdk\_core.authenticators import BasicAuthenticator

import wiotp.sdk.device

# Constants for IBM COS values

COS\_ENDPOINT = "https://manohar.s3.jp-tok.cloud-object-storage.appdomain.cloud" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints

COS\_API\_KEY\_ID = "67MBwu0H49L21goGQBII-SufzagQ-j-Lc-1xIRZOyqat" # eg "W00YixxxxxxxxxxMB-odB-2ySfTrFBIQQWanc--P3byk"

COS\_INSTANCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/94a9058de7634a75b86a721e2524a404:9eaeda1b-4ff4-4fa1-94a8-9917e79c2fc3::"

# Create resource

cos = ibm\_boto3.resource("s3",

ibm\_api\_key\_id=COS\_API\_KEY\_ID,

ibm\_service\_instance\_id=COS\_INSTANCE\_CRN,

config=Config(signature\_version="oauth"),

endpoint\_url=COS\_ENDPOINT

)

authenticator = BasicAuthenticator('apikey-v2-2y8twpk3cni02ngsc297oqatoulogdt961768upuw79q', '43c3fef46c4ca6560b7359d05c4c3d57')

service = CloudantV1(authenticator=authenticator)

service.set\_service\_url('https://apikey-v2-2y8twpk3cni02ngsc297oqatoulogdt961768upuw79q:43c3fef46c4ca6560b7359d05c4c3d57@7b88ba8a-383b-49c5-ba00-9a03115de98a-bluemix.cloudantnosqldb.appdomain.cloud')

def myCommandCallback(cmd):

print("Command received: %s" % cmd.data)

myConfig = {

"identity": {

"orgId": "sn7dm1",

"typeId": "ESP32",

"deviceId": "1234599"

},

"auth": {

"token": "9390569334"

}

}

client = wiotp.sdk.device.DeviceClient(config=myConfig, logHandlers=None)

client.connect()

bucket = "manohar"

def multi\_part\_upload(bucket\_name, item\_name, file\_path):

try:

print("Starting file transfer for {0} to bucket: {1}\n".format(item\_name, bucket\_name))

# set 5 MB chunks

part\_size = 1024 \* 1024 \* 5

# set threadhold to 15 MB

file\_threshold = 1024 \* 1024 \* 15

# set the transfer threshold and chunk size

transfer\_config = ibm\_boto3.s3.transfer.TransferConfig(

multipart\_threshold=file\_threshold,

multipart\_chunksize=part\_size

)

# the upload\_fileobj method will automatically execute a multi-part upload

# in 5 MB chunks for all files over 15 MB

with open(file\_path, "rb") as file\_data:

cos.Object(bucket\_name, item\_name).upload\_fileobj(

Fileobj=file\_data,

Config=transfer\_config

)

print("Transfer for {0} Complete!\n".format(item\_name))

except ClientError as be:

print("CLIENT ERROR: {0}\n".format(be))

except Exception as e:

print("Unable to complete multi-part upload: {0}".format(e))

face\_classifier=cv2.CascadeClassifier("haarcascade\_frontalface\_default.xml")

eye\_classifier=cv2.CascadeClassifier("haarcascade\_eye.xml")

#It will read the first frame/image of the video

video=cv2.VideoCapture("worker.mp4")

while True:

#capture the first frame

check,frame=video.read()

frame = cv2.resize(frame, (600,400))

gray=cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

detect = False

#detect the faces from the video using detectMultiScale function

faces=face\_classifier.detectMultiScale(gray,1.3,5)

eyes=eye\_classifier.detectMultiScale(gray,1.3,5)

print(faces)

#drawing rectangle boundries for the detected face

for(x,y,w,h) in faces:

print(x,y,w,h)

#print(type(x+h/2))

detect = True

cv2.circle(frame, (int(x+h/2),int(y+w/2)), int(w/2), (0,0,255), 2)

cv2.imshow('Face detection', frame)

picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M")

cv2.imwrite(picname+".jpg",frame)

multi\_part\_upload(bucket, picname+'.jpg', picname+'.jpg')

json\_document={"link":COS\_ENDPOINT+'/'+bucket+'/'+picname+'.jpg'}

response = service.post\_document(db="access", document=json\_document).get\_result()

print(response)

#drawing rectangle boundries for the detected eyes

for(ex,ey,ew,eh) in eyes:

cv2.rectangle(frame, (ex,ey), (ex+ew,ey+eh), (0,0,255), 2)

cv2.imshow('Face detection', frame)

myData={'Face\_detect': detect}

client.publishEvent(eventId="status", msgFormat="json", data=myData, qos=0, onPublish=None)

print("Data published to IBM IoT platfrom: ",myData)

client.commandCallback = myCommandCallback

time.sleep(2)

#waitKey(1)- for every 1 millisecond new frame will be captured

Key=cv2.waitKey(1)

if Key==ord('q'):

#release the camera

video.release()

#destroy all windows

cv2.destroyAllWindows()

break

