Project Development Phase

Sprint 3-Python Coding

Date	12 Nov 2022
Team ID	PNT2022TMID11425
Project Name	Virtual Eye - Life Guard for Swimming Pools To Detect Active Drowning
Maximum Marks	4Marks

Init.py

```
from .object_detection import detect_common_objects
```

app.py

```
import cv2
import os
import numpy as np
#from utils import download_file
import cvlib as cv
from cvlib.object_detection import draw_bbox
import time
from playsound import playsound
import requests
from cloudant.client import Cloudant
from flask import Flask, flash, redirect, render_template, request, url_for,
Response
from werkzeug.utils import secure_filename
#import detect
UPLOAD_FOLDER = "static/uploads/"
RESULTS_FOLDER = "static/results/"
app=Flask(__name__,template_folder='template')
app.secret_key = "secret-key"
app.config["UPLOAD_FOLDER"] = UPLOAD_FOLDER
from cloudant.client import Cloudant
```

```
client=Cloudant.iam('e80322c6-5b15-4385-ba6a-c587c3471e4b-
bluemix','Kf6tBtrDrpQZtYfredJ-rkYky1lX39giPycwe0lhCmyj',connect=True)
@app.route("/")
def index():
   return render_template("index.html")
@app.route("/register", methods=["GET", "POST"])
def register():
   if request.method == "POST":
   # Get the form data
      try:
         uname = request.args.get('name')
         username = request.args.get('email')
         psw = request.args.get('psw')
         print(list(request.form.values()))

          #email = request.form["email"]
         #password = request.form["password"]
         # Create a database using an initialized client
         my_database = client['my_db']
         # Check that the database doesn't already exist
         if my_database.exists():
            print(f"'{my_database}' successfully created.")
         # Create a JSON document
         json_document = {
         "_id": email,
         "name": uname,
         "email": email,
         "psw": psw,
         }
         if email in my_database:
            return render_template("register.html", msg="Email already exists")
         else:
         # Create a document using the Database API
            new_document = my_database.create_document(json_document)
            return render_template("register.html", msg="Account created
successfully!")
        except Exception as e:
          return render_template("register.html", msg="Something went wrong!
Please try again")
    if request.method == "GET":
        return render_template("register.html")
@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        username = request.args.get('email')
        psw = request.args.get('psw')
        print (username, psw)
        # Create a database using an initialized client
        my_database = client['my_db']
        query = {'email': {'$eq': username}}
        docs = my_database.get_query_result(query)
        print(docs)
        my_database.get_query_result(query)
        print(len(docs.all()))
        if(len(docs.all())==0):
          return render_template("register.html", prediction="The username is
not found.")
        else:
          if((username==docs[0][0]['email'] and psw==docs[0][0]['psw'])):
            return redirect(url_for("predict"))
          else:
            return render_template("login.html", msg="Invalid credentials!")
    if request.method == "GET":
        return render_template("login.html")
@app.route("/predict", methods=["GET", "POST"])
def predict():
    if request.method == "POST":
        webcam = cv2.VideoCapture("drowning.mp4")
        if not webcam.isOpened():
```

```
        print("Could not open webcam")
        exit()

    t0 = time.time()  # gives time in seconds after 1970
    # variable dcount stands for how many seconds the person has been
standing still for
    centre0 = np.zeros(2)
    isDrowning = False

    # this loop happens approximately every 1 second, so if a person doesn't
move,
    # or moves very little for 10seconds, we can say they are drowning
    # loop through frames
    t0 = time.time()  # gives time in seconds after 1970

    # variable dcount stands for how many seconds the person has been
standing still for
    centre0 = np.zeros(2)
    isDrowning = False

    # this loop happens approximately every 1 second, so if a person doesn't
move,
    # or moves very little for 10seconds, we can say they are drowning

    # loop through frames
    while webcam.isOpened():
        # read frame from webcam
        status, frame = webcam.read()

        if not status:
            print("Could not read frame")
            exit()

        # apply object detection
        bbox, label, conf = cv.detect_common_objects(frame)
```

```
# simplifying for only 1 person

# s = (len(bbox), 2)
print(bbox)
if len(bbox) > 0:
    bbox0 = bbox[0]
    # centre = np.zeros(s)
    centre = [0, 0]

    # for i in range(0, len(bbox)):
    # centre[i] =[(bbox[i][0]+bbox[i][2])/2,(bbox[i][1]+bbox[i][3])/2 ]

    centre = [(bbox0[0] + bbox0[2]) / 2, (bbox0[1] + bbox0[3]) / 2]

    # make vertical and horizontal movement variables
    hmov = abs(centre[0] - centre0[0])
    vmov = abs(centre[1] - centre0[1])
    # there is still need to tweek the threshold
    # this threshold is for checking how much the centre has moved

    x = time.time()

    threshold = 30
    if hmov > threshold or vmov > threshold:
        print(x - t0, "s")
        t0 = time.time()
        isDrowning = False

    else:

        print(x - t0, "s")
        if (time.time() - t0) > 5:
            isDrowning = True
```

```
                # print('bounding box: ', bbox, 'label: ' label ,'confidence: ' conf[0],
'centre: ', centre)
                # print(bbox,label ,conf, centre)
                print("bbox: ", bbox, "centre:", centre, "centre0:", centre0)
                print("Is he drowning: ", isDrowning)

                centre0 = centre
                # draw bounding box over detected objects

            out = draw_bbox(frame, bbox, label, conf, isDrowning)

            # print('Seconds since last epoch: ', time.time()-t0)

            # display output
            cv2.imshow("Real-time object detection", out)
            print(isDrowning)
            if isDrowning == True:

                playsound("alarm.mp3")

            # press "Q" to stop
            if cv2.waitKey(1) & 0xFF == ord("q"):
                break

        # release resources
        webcam.release()
        cv2.destroyAllWindows()

        if isDrowning == True:
            return render_template("prediction.html",prediction='Emergency!!! The
person is drowning')
        else:
            return render_template("logout.html")
        return render_template("logout.html")
    if request.method == "GET":
        return render_template("prediction.html")
@app.route("/logout", methods=["GET"])
def logout():
    return render_template("logout.html")

if __name__ == "__main__":
    app.run(port=4000,debug=True)
```

object_detection.py

```
#import necessary packages
import cv2
import os
import numpy as np
from cvlib.utils import download_file

initialize = True
net = None
dest_dir = os.path.expanduser('~') + os.path.sep + '.cvlib' + os.path.sep +
'object_detection' + os.path.sep + 'yolo' + os.path.sep + 'yolov4'
classes = None
#colors are BGR instead of RGB in python
COLORS = [0,0,255], [255,0,0]

def populate_class_labels():

    #we are using a pre existent classifier which is more reliable and more
efficient than one
    #we could make using only a laptop
    #The classifier should be downloaded automatically when you run this script
    class_file_name = 'yolov3_classes.txt'
    class_file_abs_path = dest_dir + os.path.sep + class_file_name
    url = 'https://github.com/Nico31415/Drowning-
Detector/raw/master/yolov3.txt'
    if not os.path.exists(class_file_abs_path):
        download_file(url=url, file_name=class_file_name, dest_dir=dest_dir)
    f = open(class_file_abs_path, 'r')
    classes = [line.strip() for line in f.readlines()]
    return classes
def get_output_layers(net):
    #the number of output layers in a neural network is the number of possible
    #things the network can detect, such as a person, a dog, a tie, a phone...
    layer_names = net.getLayerNames()
    layer_names = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
    return layer_names


def draw_bbox(img, bbox, labels, confidence, Drowning, write_conf=False):

    global COLORS
    global classes

    if classes is None:
        classes = populate_class_labels()

    for i, label in enumerate(labels):

        #if the person is drowning, the box will be drawn red instead of blue
        if label == 'person' and Drowning:
            color = COLORS[0]
            label = 'DROWNING'
        else:
            color = COLORS[1]

        if write_conf:
            label += ' ' + str(format(confidence[i] * 100, '.2f')) + '%'



        #you only need to points (the opposite corners) to draw a rectangle. These
points
        #are stored in the variable bbox
        cv2.rectangle(img, (bbox[i][0],bbox[i][1]), (bbox[i][2],bbox[i][3]), color, 2)

        cv2.putText(img, label, (bbox[i][0],bbox[i][1]-10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    return img

def detect_common_objects(image, confidence=0.5, nms_thresh=0.3):

    Height, Width = image.shape[:2]
    scale = 0.00392

    global classes
    global dest_dir

    #all the weights and the neural network algorithm are already preconfigured
    #as we are using YOLO

    #this part of the script just downloads the YOLO files
    config_file_name = 'yolov4.cfg'
    config_file_abs_path = dest_dir + os.path.sep + config_file_name

    weights_file_name = 'yolov4.weights'
    weights_file_abs_path = dest_dir + os.path.sep + weights_file_name

    url = 'https://github.com/Nico31415/Drowning-
Detector/raw/master/yolov3.cfg'

    if not os.path.exists(config_file_abs_path):
        download_file(url=url, file_name=config_file_name, dest_dir=dest_dir)

    url = 'https://pjreddie.com/media/files/yolov3.weights'

    if not os.path.exists(weights_file_abs_path):
        download_file(url=url, file_name=weights_file_name, dest_dir=dest_dir)
    global initialize
    global net

    if initialize:
        classes = populate_class_labels()
        net = cv2.dnn.readNet(weights_file_abs_path, config_file_abs_path)
        initialize = False
    blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True,
crop=False)
    net.setInput(blob)
    outs = net.forward(get_output_layers(net))
    class_ids = []
    confidences = []
    boxes = []

    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            max_conf = scores[class_id]
            if max_conf > confidence:
                center_x = int(detection[0] * Width)
                center_y = int(detection[1] * Height)
                w = int(detection[2] * Width)
                h = int(detection[3] * Height)
                x = center_x - w / 2
                y = center_y - h / 2
                class_ids.append(class_id)
                confidences.append(float(max_conf))
                boxes.append([x, y, w, h])
    indices = cv2.dnn.NMSBoxes(boxes, confidences, confidence, nms_thresh)

    bbox = []
    label = []
    conf = []
    for i in indices:
        i = i
        box = boxes[i]
        x = box[0]
        y = box[1]
        w = box[2]
        h = box[3]
        bbox.append([round(x), round(y), round(x+w), round(y+h)])
        label.append(str(classes[class_ids[i]]))
        conf.append(confidences[i])

    return bbox, label, conf
```

utils.py

```
import requests
import progressbar as pb
import os

def download_file(url, file_name, dest_dir):

    if not os.path.exists(dest_dir):
        os.makedirs(dest_dir)

    full_path_to_file = dest_dir + os.path.sep + file_name

    if os.path.exists(dest_dir + os.path.sep + file_name):
        return full_path_to_file

    print("Downloading " + file_name + " from " + url)

    try:
        r = requests.get(url, allow_redirects=True, stream=True)
    except:
        print("Could not establish connection. Download failed")
        return None

    file_size = int(r.headers['Content-Length'])
    chunk_size = 1024
    num_bars = round(file_size / chunk_size)

    bar = pb.ProgressBar(maxval=num_bars).start()

    if r.status_code != requests.codes.ok:
        print("Error occurred while downloading file")
        return None

    count = 0

    with open(full_path_to_file, 'wb') as file:
        for chunk in  r.iter_content(chunk_size=chunk_size):
            file.write(chunk)
            bar.update(count)
            count +=1

    return full_path_to_file
```