ARK Task Round Documentation

Basic Tasks

Task 2.2 Face Detection

I. INTRODUCTION

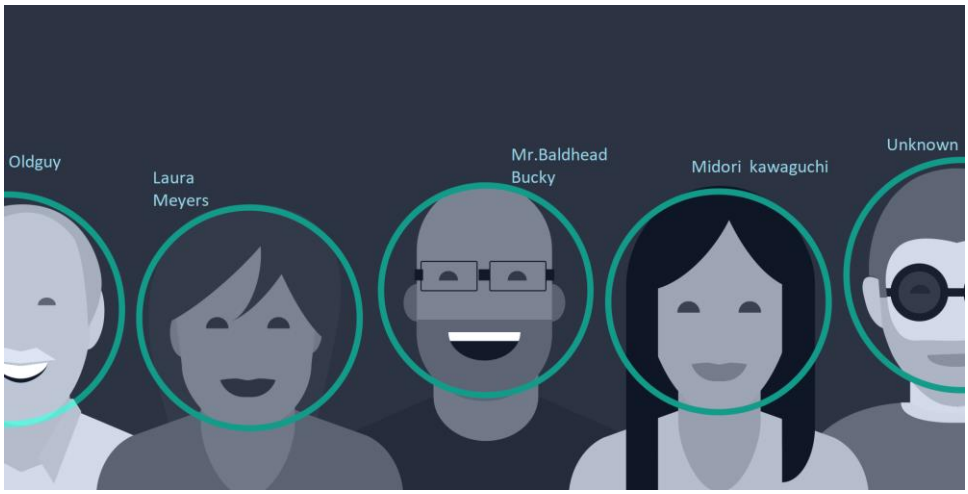
The second task is primarily about face detection. So, the task is about creating a game wherein your face plays the role of a plank or like an object so that the ball bounces off it and does not hit the floor. If the ball hits the floor, then the game is over. The ball is assumed to have perfectly elastic collision and is assumed to be in a gravity free area.

I tried to implement face detection using 'haarcascade'. And I did the bouncing ball code using 'tkinter'.

II. Problem Statement

The task has 2 main components.

The first part is to have a code for detection of face.



For example, in the image shown all the faces have been recognised and a circle is drawn around the faces.

Our here you have to recognise the face and then you could draw an ellipse or a rectangle on the identified face.

The second part was to have a game wherein the ball bounces off walls and ceilings and the game gets over if the ball touches the floor.

The ball is assumed to have a perfectly elastic collision and we assume a gravity free space.

The basic idea we use is that if the ball reaches the width of the screen, that is x+r>width or if x-r<0, that is it undergoes sideways collision then the velocity in x direction gets reversed whereas if y+r >height, that is a collision in the upward direction then the velocity in y direction should get reversed (x and y are the coordinates of the centre of the ball).

Umm perhaps the third part of the task was to have a live video streaming or to access your webcam through python codes.

And lastly you have to combine all these to have a face detection game.

III. Related Work

I have used libraries such as OpenCV, NumPy, Tkinter and PIL. For face detection I have used already defined functions and libraries. To display the game, I have used tkinter. I broke the problem into smaller pieces thus making it easier for debugging as well as for writing code.

IV. Initial Attempts

So firstly, I had used tkinter to have a simple bouncing code. Later on, I added up things like having a plank that moves using WASD keys followed by a moving image. The problem that I was facing was that it was not able to recognise the image properly. As in at times there was no image but it would still bounce off and at times instead off bouncing off from the edge of the image it would bounce off from the centre. Also, to make the image move right and left was a bit difficult, as in it would only move right or only left. I had defined like a bunch of functions and was accessing one function inside another so it had gotten a bit confusing, like the image coordinates got a bit messy. So, I had to get the image coordinates in the function where I was calling the move right () and move left () functions but then I had not done, so I faced run time errors. Another problem was that upon reaching middle of the screen it would not go to the other side. So, I had the image oscillating either between 0 to w/2 or between w/2 and w (I was still not able to rectify this error 😕).

For the face detection part first, initially I had used of images identify faces followed by trying to identify faces on live video. For live video as well first I had used tkinter to just have a streaming video. And then I added the haarcascade files to recognise faces.

I was not able to combine the two things properly though. So, I tried to add the face recognition code to tkinter's live streaming code. It identifies the faces but was not able to display a video. So, it would display a frame and then you press 'q' and then the frame would change.

V. Final Approach

Here is my bouncing ball code. So, when it reaches the edge of the screen it reverses the speed, that is on a horizontal collision speed in x direction gets reversed whereas in a vertical collision the speed in y direction gets reversed.

```
import numpy as np
import random
import tkinter as tk
```

```
WIDTH = 400
HEIGHT = 400
initial_speeds = [-6, -5, -4, 4, 5, 6]
dx, dy = 0, 0
while dx == dy:
    dx, dy = random.choice(initial_speeds), random.choice(initial_speeds)


def bounce():
    global dx, dy
    x0, y0, x1, y1 = canvas.coords(my_ball)
    if x0 <= 0 or x1 >= WIDTH:    # compare to left of ball bounding box
on the left wall, and to the right on the right wall
        dx = -dx
    if y0 <= 0 or y1 >= HEIGHT:   # same for top and bottom walls
        dy = -dy
    canvas.move(my_ball, dx, dy)
    root.after(50, bounce)


if __name__ == '__main__':


    root = tk.Tk()
    root.wm_title("Bouncing Ball")
    canvas = tk.Canvas(root, width=400, height=400, bg="black")
    canvas.pack(expand=True, fill=tk.BOTH)


    size=10
    x = 50
    y = 50
```

```
    my_ball = canvas.create_oval(x-size, y-size, x+size, y+size,
fill="blue")



    bounce()

    root.mainloop()
```

In this code you can move the plank using right and left keys. If the ball touches the floor the game ends whereas if the ball hits the plank the game continues.

```
import tkinter as tk


root = tk.Tk()


width = 900

height = 500


canvas = tk.Canvas(root, bg='white', width=width, height=height)

canvas.pack()


ball = canvas.create_oval(430, 10, 470, 50, fill='green')


platform_y = height - 20

platform = canvas.create_rectangle(width//2-50, platform_y, width//2+50,
platform_y+10, fill='black')
```

```
# ball moving speed
xspeed = yspeed = 2


def move_ball():
  global xspeed, yspeed
  x1, y1, x2, y2 = canvas.coords(ball)
  if x1 <= 0 or x2 >= width:
    # hit wall, reverse x speed
    xspeed = -xspeed
  if y1 <= 0:
    # hit top wall
    yspeed = 2
  elif y2 >= platform_y:
    # calculate center x of the ball
    cx = (x1 + x2) // 2
    # check whether platform is hit
    px1, _, px2, _ = canvas.coords(platform)
    if px1 <= cx <= px2:
      yspeed = -2
    else:
      canvas.create_text(width//2, height//2, text='Game Over', font=('Arial
Bold', 32), fill='red')
      return
  canvas.move(ball, xspeed, yspeed)
  canvas.after(20, move_ball)


def board_right(event):
  x1, y1, x2, y2 = canvas.coords(platform)
  # make sure the platform is not moved beyond right wall
  if x2 < width:
```

```
    dx = 100
    canvas.move(platform, dx, 0)


def board_left(event):
  x1, y1, x2, y2 = canvas.coords(platform)
  # make sure the platform is not moved beyond left wall
  if x1 > 0:
    dx = 100
    canvas.move(platform, -dx, 0)


canvas.bind_all('<Right>', board_right)
canvas.bind_all('<Left>', board_left)



move_ball()


root.mainloop()
```

This is the code where I have replaced the plank with an image. The image moves right and left on its own. If the ball touches the upper edge of the image it rebounds. If the ball touches the floor, then the game gets over.

I have created a canvas(background) and a ellipse(ball) using tkinter functions. I have defined a function to move the ball and bounce it on reaching the upper edge or the right or left edges of the canvas. If the y coordinate of the ball is equal to the y coordinate of the image, then it checks if the x coordinate of the ball is in the range of the x coordinate of the image. If it is so then the ball would rebound else the ball falls on the floor and then the game ends.

To move the image, I have defined the move right and move left functions and another function movement () to call the move right and move left functions periodically to move the image. The movement () keeps on repeating itself and hence the image continues to move.

```
import tkinter as tk
import cv2 as cv
root = tk.Tk()


width = 900
height = 500


canvas = tk.Canvas(root, bg='white', width=width, height=height)
canvas.pack()


ball = canvas.create_oval(430, 10, 470, 50, fill='green')


platform_y = height - 40
img = tk.PhotoImage(file="C:/Research Group/Level1.png")
image = canvas.create_image(width//2-50, platform_y,anchor=tk.N, image=img)
xspeed = yspeed = 2


def move_ball():
  global xspeed, yspeed
  x1, y1, x2, y2 = canvas.coords(ball)
  if x1 <= 0 or x2 >= width:
    # hit wall, reverse x speed
    xspeed = -xspeed
  if y1 <= 0:
    # hit top wall
    yspeed = 2
  elif y2 >= platform_y:
```

```
    # calculate center x of the ball
    cx = (x1 + x2) // 2
    # check whether platform is hit
    px1, px2 = canvas.coords(image)
    w, h = img.width(), img.height()
    if px1 <= cx <=px1+ w:
      yspeed = -2
    else:
      canvas.create_text(width//2, height//2, text='Game Over', font=('Arial
Bold', 32), fill='red')
      return
  canvas.move(ball, xspeed, yspeed)
  canvas.after(20, move_ball)


x1, y1 = canvas.coords(image)
w, h = img.width(), img.height()


flag=0


def right():
  x1, y1 = canvas.coords(image)
  print(x1)
  w, h = img.width(), img.height()
  if (x1+w)>(width+90):
    return
  dx = 6
  canvas.move(image, dx, 0)
  root.after(94, right)
```

```
def left():
  x1, y1 = canvas.coords(image)
  w, h = img.width(), img.height()
  if(x1<0):
    return
  dx = 6
  canvas.move(image, -dx, 0)
  root.after(94, left)


def movement():
  x1, y1 = canvas.coords(image)
  w, h = img.width(), img.height()
  if (x1+w>width+90):
    left()
  else:
    right()
  root.after(5000, movement)


movement()
move_ball()
root.mainloop()
```

Next, I had tried to capture video from the webcam. So, I created a frame where I could stream the video and then used the cv2.VideoCapture to capture from camera. Further I defined a function to constantly stream the video.

```
from tkinter import *
```

```
import tkinter as tk
from PIL import ImageTk, Image
import cv2
import sys


root = tk.Tk()
# Create a frame
app = Frame(root, bg="white")
app.grid()
# Create a label in the frame
lmain = Label(app)
lmain.grid()


# Capture from camera
cap = cv2.VideoCapture(0)


# function for video streaming
def video_stream():
    _, frame = cap.read()
    cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)
    lmain.imgtk = imgtk
    lmain.configure(image=imgtk)
    lmain.after(1, video_stream)


video_stream()
root.mainloop()
```

Then I tried to apply face recognition of an image with several faces in it. For this I used Haar cascade's face cascade function and drew a rectangle around the identified faces.

```
import cv2 as cv


# Read image from your local file system
original_image = cv.imread('C:/Users/admin/Pictures/Screenshots/Screenshot (100).png')


# Convert color image to grayscale for Viola-Jones
grayscale_image = cv.cvtColor(original_image, cv.COLOR_BGR2GRAY)
face_cascade = cv.CascadeClassifier('haarcascade_frontalface_alt.xml')
detected_faces = face_cascade.detectMultiScale(grayscale_image)
for (column, row, width, height) in detected_faces:
    cv.rectangle( original_image,(column, row),(column + width, row + height),(255, 255, 255), 2)
cv.imshow('Image', original_image)
cv.waitKey(0)
cv.destroyAllWindows()
```

Lastly, I tried to do face recognition on a streaming video and drew an ellipse around the recognised face.

```
import cv2
```

```
import sys


cascPath = "haarcascade_frontalface_alt.xml"
faceCascade = cv2.CascadeClassifier(cascPath)


video_capture = cv2.VideoCapture(0)


while True:
    ret, frame = video_capture.read()
    frame = cv2.flip(frame,1)


    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)


    faces = faceCascade.detectMultiScale(frame, 1.1, 4)


    for (x, y, w, h) in faces:
        center = (x + w//2, y + h//2)
        frame = cv2.ellipse(frame, center, (w//2, h//2), 0, 0, 360, (255, 255,
255), 4)


    cv2.imshow('Video', frame)


    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
video_capture.release()
cv2.destroyAllWindows()
```

I tried to do the face recognition using tkinter but I really was not able to do so. Like it was recognising the face but was not displaying live video. When I pressed the key ‘q’ then the frame would change and it would display the new image and an oval around the recognised face.

```
from tkinter import *
```

```
from PIL import ImageTk, Image
import cv2



root = Tk()
# Create a frame
app = Frame(root, bg="white")
app.grid()
# Create a label in the frame
lmain = Label(app)
lmain.grid()


cascPath ="haarcascade_frontalface_alt.xml"
faceCascade = cv2.CascadeClassifier(cascPath)


# Capture from camera
cap = cv2.VideoCapture(0)


# function for video streaming
def video_stream():
    _, frame = cap.read()
    cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)
    lmain.imgtk = imgtk
    lmain.configure(image=imgtk)

    while True:
```

```
     faces =
faceCascade.detectMultiScale(cv2image,scaleFactor=1.1,minNeighbors=5,minSize=(
30, 30),flags=cv2.CASCADE_SCALE_IMAGE)


    # Draw an ellipse around the faces
     for (x, y, w, h) in faces:
        cenx = x + w//2
        ceny = y + h//2
        center = (cenx,ceny)
        cv2.ellipse(frame, center, (w-50, h-50), 0, 0, 360, (255, 255, 0), 4)
        faces = frame[y:y + h, x:x + w]
    # Display the resulting frame
     cv2.imshow('Video', frame)

     lmain.after(1, video_stream)
     if cv2.waitKey(1) & 0xFF == ord('q'):
        break



video_stream()
root.mainloop()
```

The major problem that I faced was that I was not able to combine the codes into a single code. Like I had broken down the code and was able to implement the smaller tasks but was not able to achieve the final code.

The other error I was facing was in the drawing of an ellipse after the face is recognised. Like in one code there was just a small circle on the nose and in another the ellipse perhaps included the neck part as well. And one code drew a tiny circle somewhere on the hairs. A bit of playing around had helped me to rectify these errors.

Another error that I faced was in moving of the image right and left. Also, the ball was not able to recognise the upper edge of the image, at times it would bounce off from the centre of the image. Printing the x and y coordinates helped me understand what was going wrong in my logic and helped me correct the errors. (I was not able to rectify one error though, that is the image stops at the centre and then continues back in the reverse direction. So, like I have an image oscillating in either the left half of the screen or the right half of the screen.

Results and Observation

For face detection, the first step that is carried out is to separate the 'face' from the background. The image is then used to identify distinctive features such as contour of eye sockets, nose and chin. The faces are detected using certain features such as the corner of the eye...

Also, tkinter has a lot of functions for creation of animated videos, games and it was fun playing around.

Drawback of my code was that it was not able to perform the tasks in a single code. Like I wasn't able to merge the code to perform the face detection game.

I chose this method because I had found it simple to implement.

Future Works

So firstly, I feel that to further improvise this task would be to recognise the face and then show only the face or a circle on the output screen. The circle should move in the same way as the face does. This would look neater. Another thing that could be implemented is to have different starting position of the ball each time after which the game gets over or we could have levels in the game wherein if the player is able to not the make the ball touch the

floor for more than a minute then the next round would get loaded where the ball moves with a greater speed and starts from a different position.

The problem that I faced in my algorithm was that I was not able to combine the tkinter live video with face recognition and bouncing ball. Also, initially the code takes a lot of time to switch on the camera.

Conclusion

The overall problem was about face recognition, how to implement it and how it could be used in different fields such as developing games...

Face recognition is useful in several fields such as in preventing crime, finding missing people, helping in investigations, smarter advertisements, attendance in colleges and offices and identifying people on social media platforms.

References

https://medium.com/@ankit.bhadoriya/face-recognition-using-open-cv-1ab1cfb6c29

https://www.analyticsvidhya.com/blog/2018/08/a-simple-introduction-to-facial-recognition-with-python-codes/

https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html