Face Detection Game

Anuj Kakde (20CS30005)

Abstract— **This is a a simple face detection game in python**

To begin with, the program takes input from webcam and displays it on the screen in a separate window. It then inputs a ball inside the frame. The ball is basically an object of class whose name is also object. It has various attributes like size, postion, speed etc..

There are 3 methods of this class namely __init__, insert_object, reflect. The __init__ method initializes the ball with proper physical attributes. The insert_object method is used to insert the ball at a given location in the frame of the window. At last but not the least, reflect method implements the idea of rebounding of ball whenever it undergoes collision. The velocity of ball after rebounding will depend on the line of impact, the angle of collision, the speed of ball but not on the speed of the face. Writing this method was a tough job.

When we run the program, a new window pops up showing the instructions for the player. Pressing the escape key will terminate the program while pressing any other normal key will start the game.

Once the game is started the program starts taking input from the webcam in form of frames and inserts the ball at appropriate location. It then recognizes all the faces present in the frame and makes a circle around it. In each frame, the program checks if the ball is hitting the walls and makes it reflect by reversing the component of velocity perpendicular to the wall. It also sees if the ball is in close vicinity of the face. If yes, then the reflect function is called.

The program uses inbuilt function for face recognition and it is not perfect. Due to this, sometimes the program circles things that are not faces and sometimes it fails to recognise a face. This sometimes lead to unexpected behavior of the program.

I. INTRODUCTION

The Problem statement was to make a simple face detection game using OpenCV. The game has a is webcam based and there is a moving ball in it. The objective of the player is to prevent the ball from touching the floor. The game is made in python only. I have learnt various methods of OpenCV library including face detection methods for this purpose.

The algorithm used for reflecting the ball from any surface after collision required quite a few attempts before finally being successful. Overall this program has been implemented successfully.

The program requires a note.png file and haarcascade_frontalface_default.xml in the same folder while running and the folder needs to be opened in the workbench.

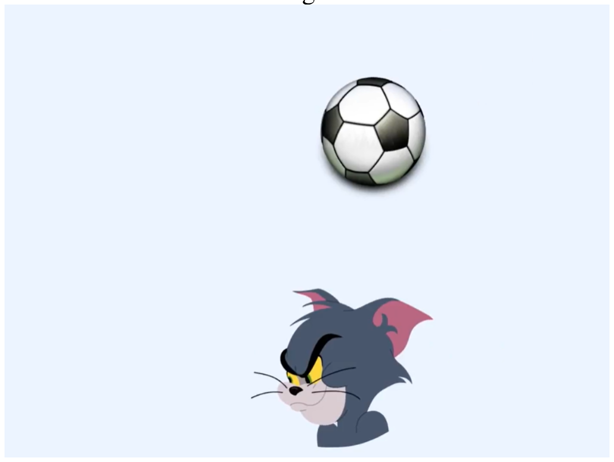
II. PROBLEM STATEMENT

In the task 2.2 ”Face detection game”, we were required to make a basic game environment. We were not supposed to use Unity or pygame and instead we had to come up with a simple UI using OpenCV.

First of all we needed to take live feed from the webcam and display it on screen. Next we had to track the movement of our face because it would serve as the only way of human input during the game. The face could be safely assumed as a circle.

Next we also had to insert a ball like object into the live feed. We had to keep a track of ball as well. The ball should bounce from all the sides of the window (expect the bottom in which case the game would be over) and also from the face. It should also follow basic laws of physics like laws of reflection while rebounding from the walls as well as the face. Collisions could be asssumed as perfectly elastic and gravity could be neglected.

While playing the game, the objective of the user is to prevent the ball from touching the bottom of the window. To achieve this the user can move his face in order to make the ball rebound at the correct angle.



The above image shows how the game would look like. The only difference will be that the user’s face will replace Tom’s face.

III. RELATED WORK

To complete this task, I had to learn to use OpenCV library in python, particularly the methods related to webcam and inserting objects in live feed. Detection of face from the live feed also required learning.

This was the first time I was using numpy arrays. It also took some efforts to get famaliar with.

IV. INITIAL ATTEMPTS

Initially I tried to solve the task using motion detection, but in that implementation the would not rebound if the face is stationary. Also if motion detection would have been implemented than the ball could be reflected from any other moving object in the background like hand, body etc.

Also in the staring I tried to add an png image of a ball into the live feed which could not be implemented properly.

V. FINAL APPROACH

This part has been divided in three subsections namely "General Setup", "Class Object" and "Gameplay

A. General Setup

In the final implementation, I took help of two libraries namely OpenCV and numpy in python. First we need to to load the face cascade that will be used to detect face from the live feed. I did this using cv2.CascadeClassifier method.

Next I capture frame from the webcam and set the window size appropriately.

B. Class Object

I have define a class named object. The ball will be an object of this class. In the class I have defined three methods namely __init__, insert_object and reflect.

In the __init__ method, I have initialized the size of ball to be 50, x and y component of speed (speedx and speedy) to be 10 each. The mag variable represents the magnitude of the net velocity of the ball, while the x and y variable are set to 50 each.

The insert_object method is used for drawing a circle of specified dimensions and coordinates into the live feed.

The reflect function is called whenever the ball undergoes any collision. The variable old_mag stores the initial magnitude of velocity using the following equation:

$$old\_mag = \sqrt{v_x^2 + v_y^2} \tag{1}$$

The variable coldist stores the distance between center of ball and the center of face (approximated as a circle). Variables nx and ny stores the cosine and sine of the angle made by the line of collision with the horizontal.

$$nx = cos\theta \tag{2}$$

$$ny = sin\theta \tag{3}$$

While calculating the rebound velocity of the ball, we will ignore the speed of our face. The variable p calculates and store in it the component of velocity along the line of impact.

$$p = v_x cos\theta + v_y sin\theta \tag{4}$$

Now we assign the new values of velocities (direction only) and then we calculate the new speed of the ball and save it in new_mag.

$$new\_mag = \sqrt{v_x^2 + v_y^2} \tag{5}$$

Next to keep the magnitude of velocity constant before and after collision, we multiply each component of velocity by old_mag and divide by new_mag.

$$v_x = v_x * old\_mag/new\_mag \tag{6}$$

$$v_y = v_y * old\_mag/new\_mag \tag{7}$$

Finally we allocate the new values of x and y component of speed after rounding to speedx and speedy respectively. Rounding is necessary we need our both components of velocity to be integers.

C. Gameplay

In the gameplay we run an infinite loop which can be exited any time by pressing escape key on from the keyboard. Each loop here represents a new game. Firstly an object of class object named ball is created. Then we display a new window and put some text on it for user instructions. Then we wait for users input and act accordingly.

Next again we have an infinite while loop where we are continuously taking input from the webcam as frames and displaying it on the window.

Ball movement is defined here. The ball seems like it is moving as its position is being changed in each frame according to the given equation-

$$x = x + v_x \tag{8}$$

$$y = y + v_y \tag{9}$$

After this I have mentioned the conditions for the ball to rebound from the walls and the top. This has been achieved with simple logic, i.e. if the coordinates of the ball are crossing the window area then the component of velocity perpendicular to the wall/top is reversed. Also each collision with the wall increases the perpendicular component of the velocity by a factor of 1.1. This feature has been added just to increase the difficulty level of the game as time passes by.

The ball is then inserted in the frame at appropriate location using the insert_object method of the class object.

Then the program detects any available face in the frame and draws a circle around it. Face detection is achieved by using the face_cascade.detectMultiScale() function. Each time after we make a circle around a face the program also checks if the ball is in the close vicinity of the face. If yes, then it calls the reflect method of ball and the values of the components of the velocity are changed accordingly.

Then the frame is flipped so as to keep it aligned for the user.

These changes are then reflected on the window.

After the ball has changed its position, if the ball is touching the floor (bottom of the window), then a note is displayed in a new window which reads "Game Over" for 2 seconds and then this window disappears and the user is asked if he/she wants to play again.

Pressing the escape key at any instant of the game will quit the game.

VI. RESULTS AND OBSERVATION

The program runs well according the how it should be. The user interface is also good.

The major drawback of this program is that it uses face detection algorithm which is not perfect. Due to this the program may fail to identify a face due to various factors like low lighting, tilted face, or a face moving fast enough that the algorithm cannot detect it. It may even circle a thing that might look like a face.

This leads to unexpected behaviour and due to this the ball sometimes gets trapped inside the face.

VII. Future Work

The face detection algorithm could be improved by increasing the data provided to the machine learning algorithm that enables face detection.

The user interface can also be improvised with help of platforms like unity and pygame.

The algorithm used for the reflection of ball from the face can be optimized further so that it also takes into consideration the speed at which the user hits the ball.

CONCLUSION

This was an interesting task which taught me the OpenCV library to a good extent. It took me about 2-3 days reading and 2-3 days coding to solve it. While coding I was stuck at the face detection part for quite a bit of time, but after checking out some good websites and github pages, I was finally able to complete it.

All the new things I learnt during it will definitely helpful while working on new projects in ARK as it gave a fundamental idea how the computer is able to ”see” things.

References

[1] https://www.geeksforgeeks.org/opencv-python-tutorial/
[2] https://www.learnpythonwithrune.org/opencv-python-webcam-create-a-simple-game-avoid-the-falling-logo/
[3] https://realpython.com/face-recognition-with-python/