Task 2 :- Face Detection Game

Srishty Gandhi, roll no. 20CS30052

Abstract— **The task was to create a game environment of a simple game in which you control your piece by moving your head**

The program captures your video cam feed. After that position of your face is detected, the box moves along with your head. We have to move the box such that the ball does not touches the bottom. Detecting face or in general anything else is a very useful technique. We can upload our drones with the object detecting software so, that it can do many tasks independently

I. INTRODUCTION

In this task I have tried to replicate the game environment, as was shown in the animation. I tracked the position of face using Haar cascade filter(haarcascade_frontalface_alt.xml) And then run a loop for the game environment. But the said filter did not recognize the face in every iteration of the loop, as a result in many frames the box was not drawn which resulted in a shaky interface. To resolve this issue I used the coordinates of face detected in previous iteration. But after this too the environment created was not smooth, the block used to get stuck when it didn't detect a face.

To solve this issue i recorded the relative of position of the face instead of absolute position. i.e. when the face is on left side of the screen the block moves towards the left, when it is on the right side, the box moves towards right, and when it is in centre, the box stops moving.

II. PROBLEM STATEMENT

Problem statement of the task was to watch an animation and create a similar game environment where one can play the game.

To make the game environment we had to assume certain things

- We have to assume that our face was a perfect circle (I made a minor change to it, I assumed that the face was a square instead of circle, as writing equations of reflection from a circular surface was becoming a messy, sorry for that)
- The balls follow the laws of physics, and all the collisions that it makes are perfectly elastic.
- We do not have to consider the effects of gravity, i.e. the ball come down only when it collides with the upper wall.
- The game ends when the ball drops to floor.

There are some additional things that i have added in the game

1) I have added a score counter on the top left hand corner
2) The velocity of the ball and the box increases as the score increases
3) I added guiding lines to help us recognize where are face is according to camera

III. RELATED WORK

A lot of work is being done in the industry related to face detection. Most common one is to combine it with machine learning techniques to store the data of face and then recognize it later on.

IV. INITIAL ATTEMPTS

Attempt 1

The initial attempt was to detect the coordinates of face and draw a square in its place. We could only control the x co-ordinate of the box by moving our face to various parts of the screen, but the y coordinate remain fixed, as the box was supposed to slide only along a particular line.

The major drawback of this method was that it does not recognize the face in every frame, due to various reasons, such as, noisy background, etc.

So when in any frame the filter was not able to detect the face, the frame went blank.

Attempt 2

To resolve the above said problem I started storing the last recorded value of X in a separate variable, so, whenever the blank frame came I made the box with help of the previous recorded value of x. As a result the box was drawn in every frame but whenever a blank frame was encountered the block stopped moving for that fraction of time, and then jumped again to new position when it started detecting the face again.

The jumping action occurred here because in the fraction of time when the face was not detected, the box stopped to move and kept its initial position, but the your face continued to move. So, when the face was detected again, there was some difference in the position of face and the previous recorded position, which results in sudden jumps. All in all again was not at all a nice experience while playing.

To resolve this issue gave a velocity to the box to, which is explained in section V. Final Approach.

V. FINAL APPROACH

If we take a close look at the code, we will realise that only a few things are going on in the code in an infinite loop. And that loops break only when we loose or when we press the key 'q' on our keyboard.

In the infinite loop the video feed is converted into a series of images and we deal with one single image over one iteration.

Step 1:-

The image that is read from the camera is stored in the form of numpy array, if no image is read from the camera in that particular image then we pass a continue statement, due to which all the further commands of the loop are ignored and the next iteration starts.

Step 2:-

After successfully reading the image we convert it to black and white mode, as it is easier to detect faces in black and white mode. This is so because the Haar cascade filter uses edge detection techniques to recognize the faces, and edges are better detected in black and white mode.

Step 3:- After that we use the Haar cascade filter to detect the faces, and store there data in a list

Step 4:-

After that the border conditions for the ball is checked. That if its touching any of the three borders, or the top face of the box, then the velocity is changed.

If it is touching the horizontal surface then vertical velocity is reversed. and if it is touching any vertical surface then horizontal velocity is reversed.

If the ball touches the floor then the loop is stopped and the game is over.

Step 5:- After the velocity is updated we update the position of both the ball and the box.

Updating the position of ball is simple, we just have to add the velocity of the ball to its x and y co- ordinates.

We update the position of the box according to the relative position of last recorded x- coordinate of the face. If the x is in left side then we decrease the x-coordinate of the box, If the last recorded x- coordinate is on the right side then we increase the x-coordinate of the box, if it is in the middle, we do not do any changes to the position of the box

Step 6:-

Then we draw all the necessary props namely a white rectangle for background, a red circle, and a blue box at desired locations

Step 7:-

After drawing these props we update the velocity of the box and the ball according to the score, to increase the difficulty level of the game with time.

That's all whats happening inside our game loop.

VI. Results and Observation

Result of this task is a decent enough game.

VII. Future Work

1) we can add some graphic effects in the game
2) we can add warnings and some rewards in game, such as
   - if we hit the ball from the middle of the box then we get a bonus point
   - if we hit the ball from almost the edge of the box then it should prompt warning.
3) Some other power-ups can also be added, suppose we drop a small clock figure from the top of the screen, and if we collect it with the block, then it will result in a slow motion.

References

[1] some youtube videos from where i learned how to use the haar cascade filter and draw basic shapes such as a rectangle and a circle.

VIII. Some stills from the game