Aerial Robotics Kharagpur Task-3 Documentation

Rishabh Agarwal

***Abstract*— In this task I had to create a game in which an object navigates through an obstacle course, and the ball's motion depends on the motion of our faces. The motion of our faces are tracked and the object is moved correspondingly.**

Object tracking has a variety of uses, some of which are surveillance and security, traffic monitoring, video communication, robot vision and animation. Object tracking by drones can extend the range of these by manifold. It can be used by our law-enforcement system to monitor suspects, track wanted criminals or for the security monitoring of large areas. It can be used in agriculture to find infected plant species in acres of land, in mapping of cities and districts and in other similar fields. In this task I've started with face detection, i.e determining whether a face is present in the live feed from the webcam, and then moved on to face tracking. Various algorithms for face detction and tracking have been investigated by me.

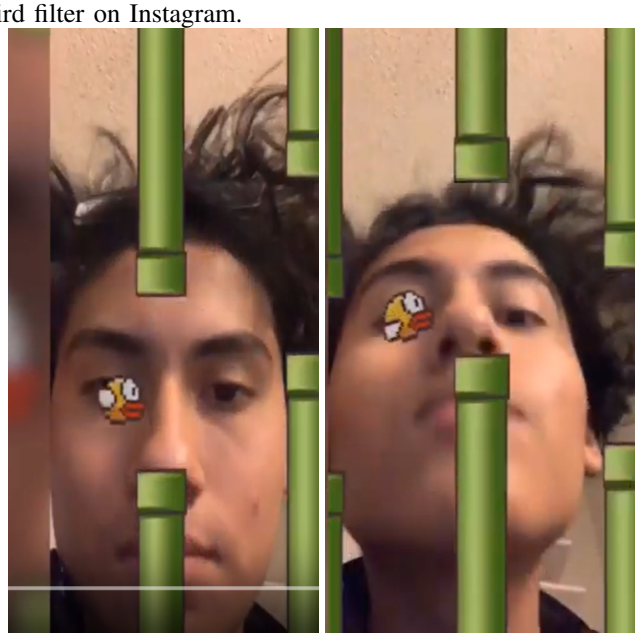
I. INTRODUCTION

For this task we were supposed to implement algorithms to detect a face and to track its motion wirhout using OpenCVs inbuilt function for tracking faces. Once we have detected and have working face tracking methods, we have to implement a game where the user has to navigate his face through obstacles.

To detect faces I have used the **Haar-Cascades Classifiers** available in OpenCV. Once the face has been detected, the rectangular region of the face is made the region of interest and then the hue values of this region are extracted and then a histogram is formed using those values. This histogram is used as a reference and for any new image the pixel value in it are replaced from the value of the reference histogram corresponding to that pixel value. This technique is called **histogram back projection**. **Mean shift** is then applied on this back projected image. Once the detection and tracking was up and running I have implemented my game by creating obstacles in the form of an array and detecting collisions by simply checking if the distance is less than the radius.

II. PROBLEM STATEMENT

The problem statement required us to implement a face detection and tracking mechanism and use it in a obstacle course game. The task requires us to first detect our face and track it. This tracking of facial movement will be used to control a ball around a game space. Collisions with obstacles in the game space will result in the game ending. We were allowed to use OpenCv's inbuilt functionalities for face detection so I used Haar Cascades for frontal face detection to begin the game. Once the face has been detected, we were required to track the face without using any functions predefined in opencv. For this part of the task I used Mean Shift face tracking algorithm. The problem requires the face to be tracked in every possible position with every possible facial expression so repeated face detection involved in tracking will not work, hence I chose a hue based tracking mechanism. The final part of the problem is to control a game unit, in my case a ball via our facial movements , through the face tracking we had implemented prior to this portion. The movement of the face will be translated to the movement of the ball on the screen. The screen will have obstacles of varying dimensions which appear randomly. Through our facial movements, we have to move the ball around this game frame. Our code has to be capable of detecting collisions of the ball with these randomised obstacles and once the ball collides the game ends. The game is similar to the flappy bird filter on Instagram.



III. RELATED WORK

Other methods used for face tracking used are **face tracking using contours** and **Template Matching**. The advantage of using contour matching is that the structure of the face is strongly represented in its description along with its algorithmic and computational simplicity that makes it suitable for both hardware and software implementation. In template matching the initial template is based on edge detection with some deviations being allowed.

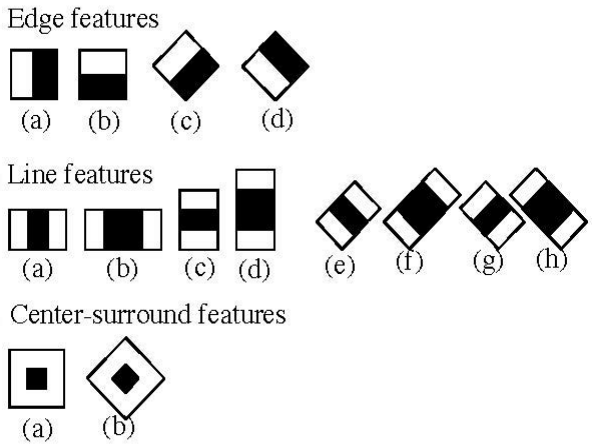
IV. INITIAL ATTEMPTS

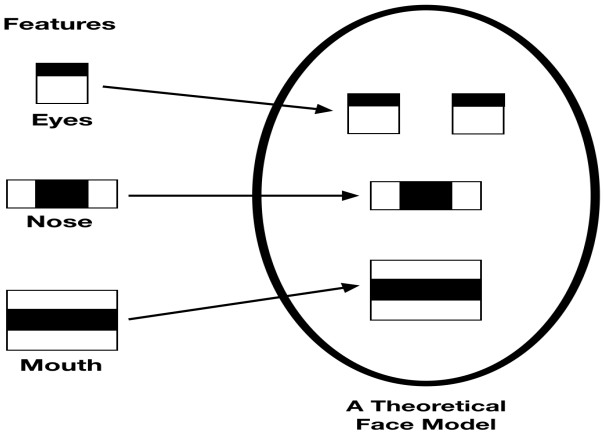
Initially instead of tracking the face, I was detecting the face for each and evry frame, which is a very computationally expensive process. Also this was not a very reliable process

as the face isn't detected if the facial expressions change. Then I looked up various object tracking algorithms, and found MeanShift and CAMShift as quite useful ones. I decided to implement the MeanShift algorithm as I thought that there was no need to use CAMShift as CAMShift is generally used when the distance of the object from the lens is changing, which was not the case in our situation. Also in CAMShift the orientation of the rectangle bordering the face kept changing which was complicating things unnecessarily. I implemented the obstacle course by storing the coordinates in an array. I had implemented the game in a vertical manner, i.e the obtsacles would be vertical and the ball would move up and down with the face. But this was not a good idea as the up and down movement of the head is restricted and also there was less space for up and down movement due to the aspect ratio of the screen.

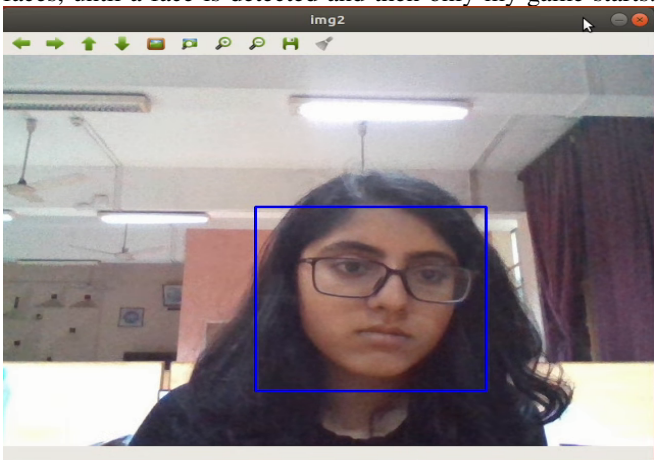
V. Final Approach

Face Detection: Haar Cascades use machine learning techniques in which a function is trained from a lot of positive and negative images. This process in the algorithm is feature extraction. The training data used in this project is an XML file called: haarcascade_frontalface_default.xml. Haar-like features are based on the concept of Haar wavelets. The features below show a box with a light side and a dark side, which is how the machine determines what the feature is. Sometimes one side will be lighter than the other, as in an edge of an eyebrow. Sometimes the middle portion may be shinier than the surrounding boxes, which can be interpreted as a nose. Additionally, when the images are inspected, each feature has a value of its own. Its calculated by subtracting the white area from the black area. The image is converted into an **integral image** to calculate the values of the features.





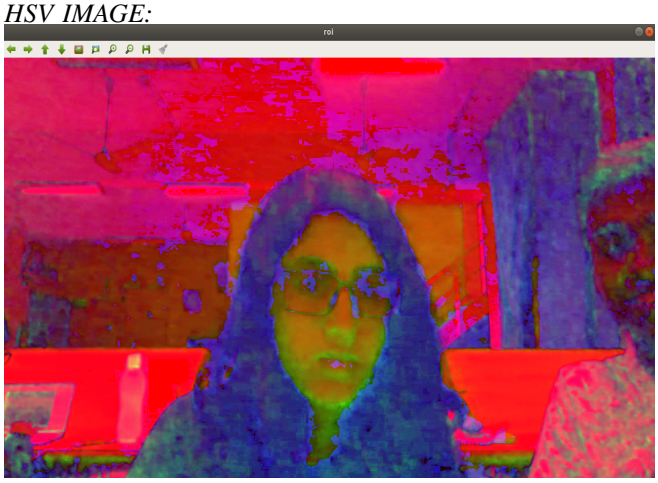
Now we train the machine to identify these features. The algorithm shrinks the image to 24 x 24 and looks for the trained features within the image. It needs a lot of facial image data to be able to see features in the different and varying forms. That's why we need to supply lots of facial image data to the algorithm so it can be trained. Haar Cascades use the Ada-boost learning algorithm which selects a small number of important features from a large set to give an efficient result of classifiers then use cascading techniques to detect face in a image. Haar cascade classifier is based on Viola Jones detection algorithm which is trained in given some input faces and non-faces and training a classifier which identifies a face. I used Haar Cascades to detect the face initially. I have run a loop and kept on detecting faces, until a face is detected and then only my game starts.



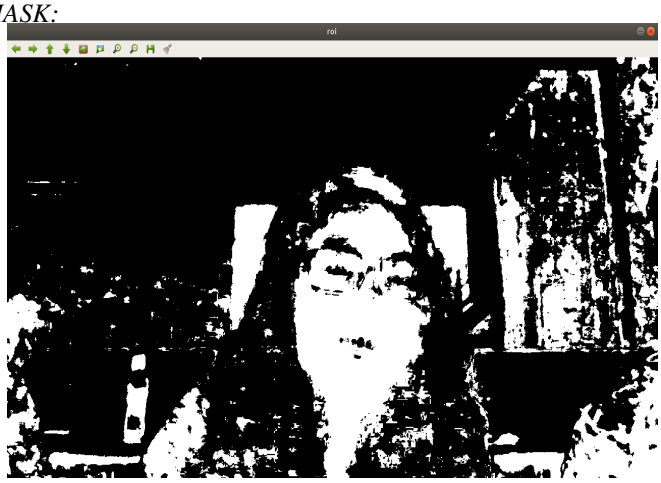
Using Meanshift: Meanshift is a clustering algorithm that assigns the datapoints to the clusters iteratively by shifting points towards the mode. The mode can be understood as the highest density of datapoints. As such, it is also known as the mode-seeking algorithm. Given a set of datapoints, the algorithm iteratively assign each datapoint towards the closest cluster centroid. The direction to the closest cluster centroid is determined by where most of the points nearby are at. So each iteration each data point will move closer to where the most points are at, which is or will lead to the cluster center. When the algorithm stops, each point is assigned to a cluster.The region of interest is now the region of the video feed which was bounded by our x y

w z coordinates. We then use histogram back-projection to replace each pixel in the roi by its mean value. Now a mask is generated in which each pixel having a value close to this mean value is of much higher intensity (appears as white in the mask) than the pixels not having a value close this value(which appear as black in the mask). We then take the video feed and generate this mask using the pixel hue we calculated above. The movement of the face is demarcated by the movement of this high intensity region.

HSV IMAGE:



MASK:

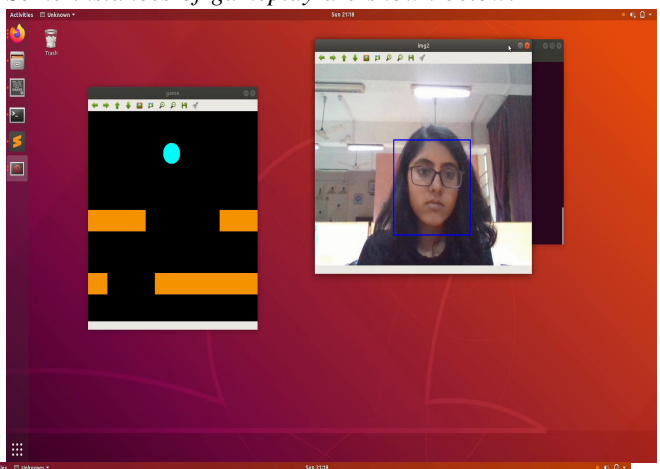


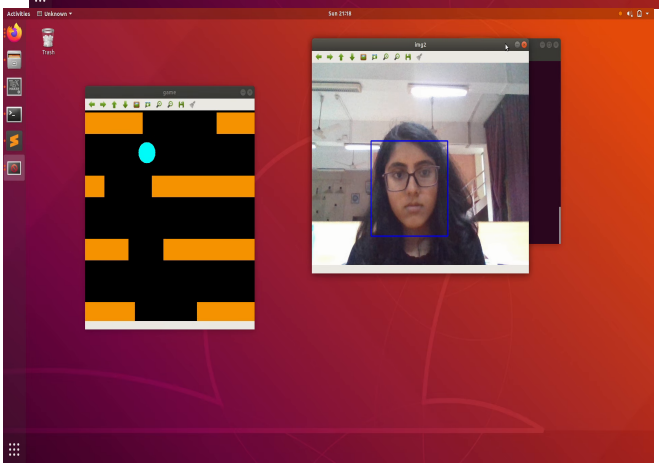
HISTOGRAM BACK PROJECTION:

To track the movement of the face we use the top left coordinates of the bounding box rectangle surrounding the region of interest which is the face in our case. This part assumes that the background colour is significantly different from the complexion of the individual playing the game and that the face remains in the screen for the entire duration of the gameplay. The second one seems like a reasonable assumption and the game works such that the game will come to and end if the individual is no longer in front of the screen. Meanshift is extremely useful for object tracking especially when a stark differencee arises between the object to be tracked and the the background and when the object can be perceived to be of not many colours, an exampled of this can be tracking players in a football field using their jersey colours againt the green colour of the field.

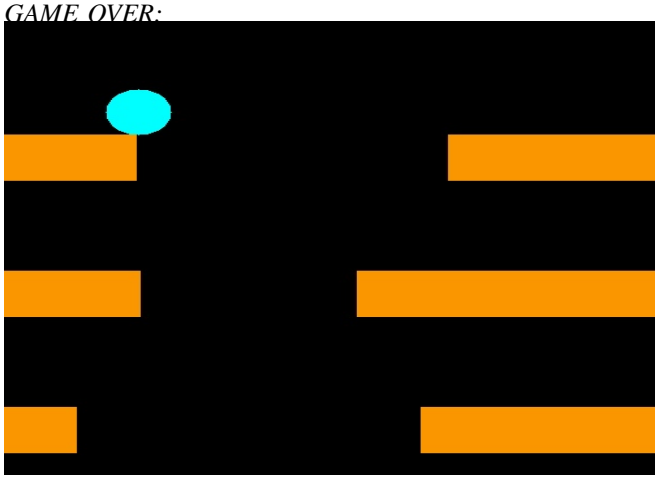
The Gameplay: The gameplay involves using a 2X5 array to store the upper and lower ends of the 5 obstacles that are generated randomly. I have kept the ball in the middle axis and any movement of the face deflects it to the corresponding direction. Thus the ball moves moves around the screen. To detect the collisions I have found the distance between the centre and the lateral sides as well as the upper and lower ends of the first obstacle. Also the ball only moves in the horizontal direction while the obstacles move vertically. If a collision occurs, the game comes to an end and the final frame is stored in 'gameover.png'. Initially the obstacles start at the end of the screen to get the user accustomed to the movement and I have also made sure that the obstacles are never too close that it is impossible to get through.

Some instances of gameplay are shown below:





GAME OVER:



VI. Results and Observation

I could have implemented some other tracking algorithm instead of MeanShift, which wouldn't have been dependent on the complexion of the face. By looking at the contours in the background and finding out hge contours which were altered due to the moving face I could've implemented a **Contour based detection algorithm**. However, due to time constraint and having already implemented mean-shift I was unable to implement this algorithm.

The sole error thar arises in this algorithm is the confusion of the facial complexion withh the background and hence resulting in the bounding rectangle getting stuck in some part of the background. A somewhat solution to this can be that when the player while playing the game feels that if the tracking is getting stuck the user can enter a key to invoke face detection again. This is not an exact solution but a makeshift one.

VII. Future Work

Face detection works pretty well but the main problem is tracking. The limitation associated with the MeanShift algorithm is that it tracks the faces on the basis of complexion. If the background has a colour similar to one's complexion, the tracking gives inaccurate results. I would like to implement some other tracking algorithms based on contours in the background in the future.

CONCLUSION

The problem with the face detection is that the parameters passed to the function needs to be changed if the no of faces changes. We do not get consistent results, they will all depend on the parameter values set for scaleFactor, minNeighbors and minSize. There is no exact value to detecting the number of faces. The user will have to use arbitrary values to come up with face detection. The code can detect faces, but it would still require verification from the user. This is therefore not a fully intelligent system since it requires interaction from the user. The problem involved using and understanding opencv really well and was a nice introduction to object tracking in general. Also implementing the game was a test of our combinational skills- image processing as well as general coding.

References

[1] https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177

[2] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, volume 1, pages I511. IEEE, 2001.

[3] https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999