Aerial Robotics Kharagpur Documentation Template

Neha Dalmia

***Abstract*— In this task, I have created a game which uses face detection and tracking to control a ball around a frame. The facial movements are translated to the movement of the ball. I used haar_cascades to detect the face and used meanshift to track it's movement.**

Object detection is breaking into a wide range of industries, with use cases ranging from personal security to productivity in the workplace. Object detection and recognition is applied in many areas of computer vision, including image retrieval, security, surveillance, automated vehicle systems and machine inspection. 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.The main objective of this task is to assist the human operators, by implementing intelligent visual surveillance systems which help in detecting and tracking suspicious or unusual events in the video sequence. The visual surveillance system will require fast and robust methods of detecting and tracking moving objects. In this task, I have investigated methods for detecting and tracking objects from drones. Moving objects are detected using adaptive background subtraction technique successfully and these detected objects are tracked by Mean-Shift tracking based techniques. The simulation results show the efficacy of these techniques in detecting and tracking moving objects in the video sequences acquired by the UAV.

I. INTRODUCTION

For this task we were supposed to implement algorithms to detect a face and to track its motion wirhour using OpenCV's inbuilt function for tracking faces. however 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.

I have used **Haar-Cascades** for face detection. Once the face has been detected, I am making the bounding rectangle the region of interest and I am then using histogram back-projection to create a mask like image where the maximum pixel density is kept track of using **Mean Shift** face tracking algorithm. Earlier I was using centroid tracking to track the face but it was reduntant as it had to do multiple face detections and could not recognize all positions of one's face, thus making it pointless to be used in a game. Mean shift too uses centroid tracking but as it tracks the face using one's complexion, it's much more effective in tracking different configuration of one's face. I then used arrays to create and rotate obstacles and detected collisions via coordinates.

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 fiirst detect our face and track it. This tracking of facial movement wiill 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 functinalities 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

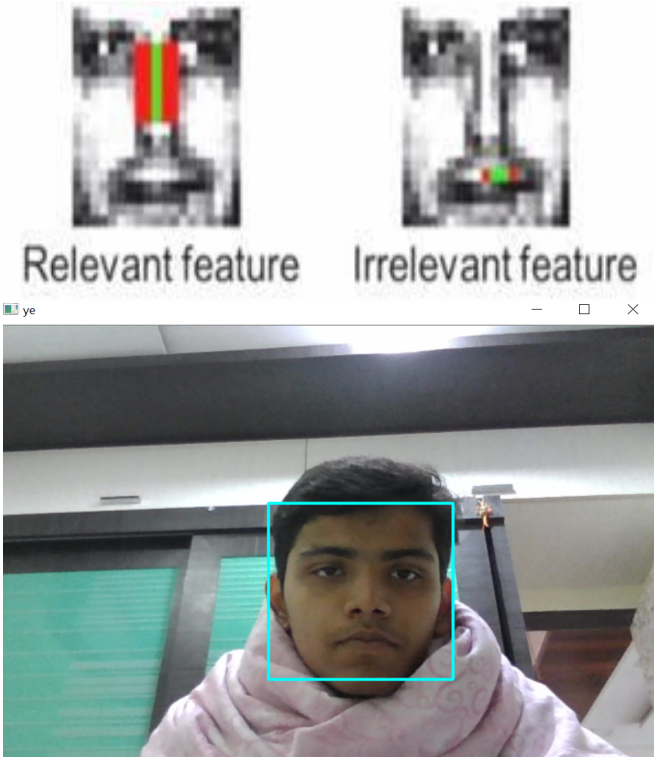
Other methods used for face tracking used are **face tracking using contours** and **Template Matching**. The advantage of using contour matching is that the structure of the face is strongly represented in its description along with its algorithmic and computational simplicity that makes it suitable 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 I used a very primitive way to track the face. I used **centroid tracking** which involved tracking multiple faces in a given video feed but has to detect faces in each frame. Hence it was neither very accurate not very fast and had to be discarded. Then I tried using **Cam Shift** which failed again due to improper adjustment resulting in a very small rectangular area being marked as the face. Then I finally used **Mean Shift** which worked for various facial positions and was quick as well. I made the game using another black frame and drawing rectangles on it using **cv2.rectangles()** method, and used rand to randomise it. For collision, initially I tried to check if a circle of radius one unit larger than my game unit circle had a pixel whose colour was not black thereby indicating a collision has occured using the **ellipse2poly()** method. However, this did not work due to the time it takes to traverse 360 points around a circle and late detections when the ball had gone too deep into the obstacle. Initially, a lot of impossible cases were also propping up with respect to the obstacles coming up as I had not used screen height and width for adding restrictions to the rectangle length.

V. Final Approach

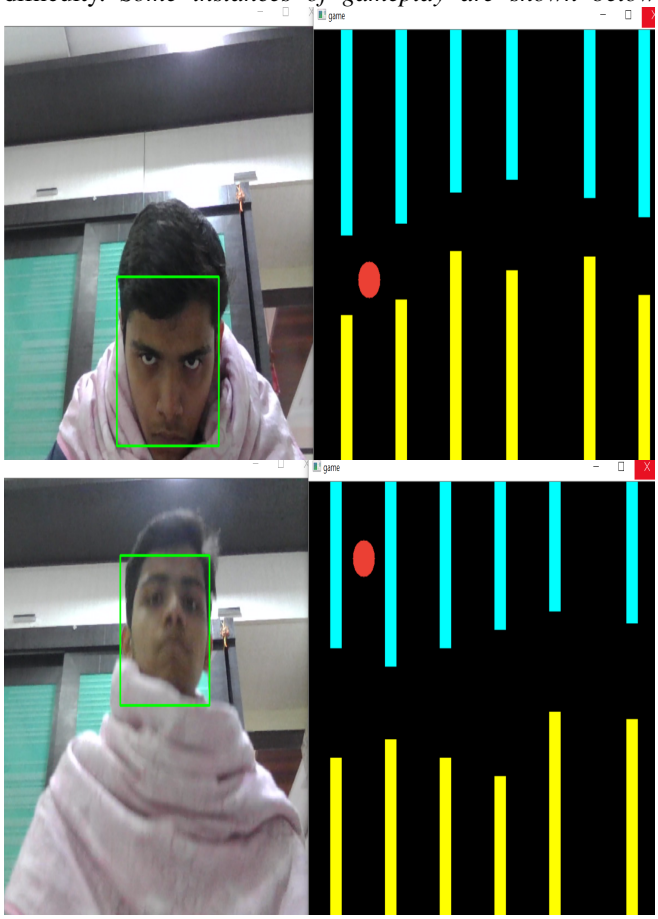
Face Detection using Haar Casacdes: Haar Cascades use machine learning techniques in which a function is trained from a lot of positive and negative images. This process in the algorithm is feature extraction. The training data used in this project is an XML file called: haarcascade_frontalface_default.xml. Haar Cascade classifier is based on the Haar Wavelet technique to analyse pixels in the image into squares by function. This uses "integral image" concepts to compute the "features" detected. 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. As Haar cascades cannot be relied upon to detect the face at the first instance every time I ran a loop to give it more time to detect the face. Once the face has been detected, it's coordiates are stored as x,y,w and z.

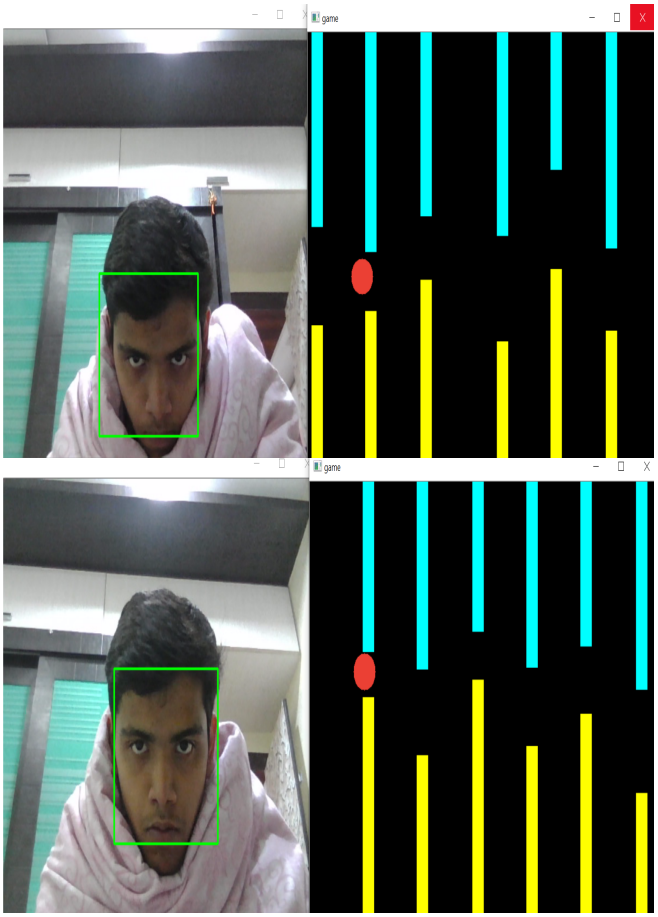


Using Meanshift: Meanshift is a clustering algorithm that assigns the datapoints to the clusters iteratively by shifting points towards the mode. The mode can be understood as the highest density of datapoints (in the region, in the context of the Meanshift). As such, it is also known as the mode-seeking algorithm. Meanshift algorithm has applications in the field of image processing and computer vision.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. 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 duraation 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

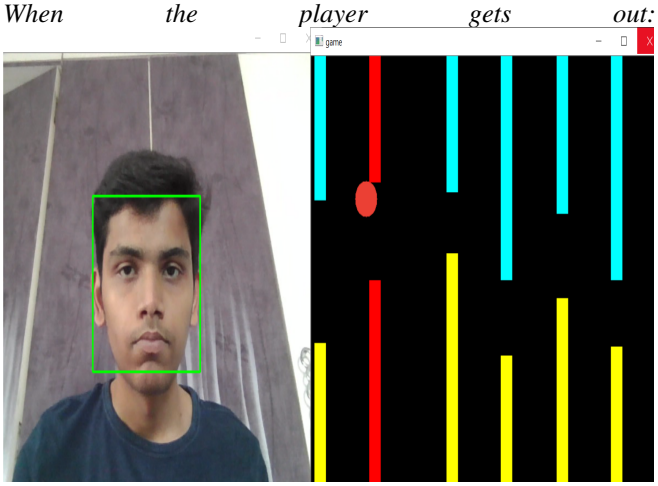
individual is no longer in front of the screen. Meanshift is extremely useful for object tracking especially when a stark differencee arises between the object to be tracked and the the background and when the object can be perceived to be of not many colours, an exampled of this can be tracking players in a football field using their jersey colours againt the green colour of the field.

The Gameplay: The gameplay involves using 4 lists to generate the different rectangular obstacles which store their height width and start and end coordinates. The topleft corner coordinates of the roi obtained from the face tracking is made the center of the ball which moves arounf the black board. This is because the top left corner is closest to a persons eye as the task required us to do in the beginning but tracking a person's eye using meanshift and haar cascade combination is more unreliable than face detection. Thus the ball moves moves around the screen. To detect the collisions we assume the circle is circumscribed by a square and using the coordinates of the edges of the square and the coordinates of the obstacles stored in the array we calculate if an intersection between them occurs using several conditions to cover all possible cases. Once an intersection occurs that partcular obstacles becomes red and the gameplay ends. Initially the obstacles have an easy, preset pattern to get the player accustomed to the environment. After a few seconds, random onstacles re generated thus increasing the difficulty. *Some instances of gameplay are shown below*





When the player gets out:



VI. Results and Observation

I could have used **contours** to track faces instead of the complexion of the face. Contours would have been independant of the position of the face and would not have got mixed up with the background due to similarities in the complexion of the face and the background colour. The algorithm would have involved finding the face contours in the region of interest and the n tracking the face on the basis of similar contour lines in the the subsequent video feed. However, due to time constraint and having already implemented meanshift I was unable to shift to using contours t solve this problem.

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. We can improve on this by somehow segementing the face to make it more distinct from the background present. I can add **Instance Segmentation** to the above image to make the face more distinct from the surrounding which will be blurred everytime the face is recognized so that the chances of confusing the face with the blurred background in the next frame will be reduced considerably.

VII. Future Work

The main limitation in testing this algorithm is that the background cannot have anything of a colour similar to one's compplexion as that will result in unexpected outcomes and the inability to continue with the game.

CONCLUSION

The problem involved using and understanding opencv really well and was a nice introduction to object tracking in general. The ability to track an object and separate it from its surroundings is a very important skill. Opencv usage to make the game environment enabled me to explore a lot more aspects of opencv and evern making a histogram back projectiona and generating the mask of the image increased my understanding of opencv greatly. Overall this task was a good test of a combination of skills.

References

[1] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K Jain. Face detection in color images. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 24(5):696–706, 2002.

[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 I–511. IEEE, 2001.

[3] Karl Schwerdt and James L Crowley. Robust face tracking using color. In Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on, pages 90–95. IEEE, 2000