Task Round Documentation

Rohit Raj *

***Abstract*— This documentation specifies the approach and results for ARL Software Team tasks 1, 2 and 3.**

For Task 1, I used minimax algorithm to make optimised system to play Tic-Tac-Toe. For second task, I used Contour detection and OpenCV drawing functions to show ball movements. For third task, I used Haar Cascades for face detection and then utilised simple drawing function for stimulate game environment.

I. INTRODUCTION

1) Task 1 :- First task involves understanding minimax algorithm [1] and implementing it to stimulate an agent which plays against human agent. Understanding minimax algorithm was important part for completing this task and after that it was easy to implement it.
2) Task 2 :- Second task involved stimulating a billiard game in which direction of movement of first ball was given. All conditions were assumed to be ideal while solving this problem i.e. it was assumed that no friction is present and all collisions with walls were assumed to be elastic. To further simplify the problem it was assumed that on collision between two balls whole velocity of one ball is imparted to other and other ball (which was initially moving) comes to rest.
3) Task 3 :- Third task involved using an in-built face detection method to detect face of user and then stimulate a flappy bird game using movement of face i.e. if face moves up then bird moves up and same for all other motions of head.

II. PROBLEM STATEMENT

This should cover one full column of page.

1) Task 1 :- First task involved understanding minimax algorithm and implementing it to stimulate an agent which plays against human agent. Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. In Minimax the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible while the minimizer tries to do the opposite and get the lowest score possible.
2) Task 2 :- Second task involved Second task involved stimulating a billiard game in which direction of movement of first ball was given. Here, in figure red line coming out of center of one of the balls shows direction of movement of that ball and movement of balls is considered as long as number of collisions reaches to count provided by the user.

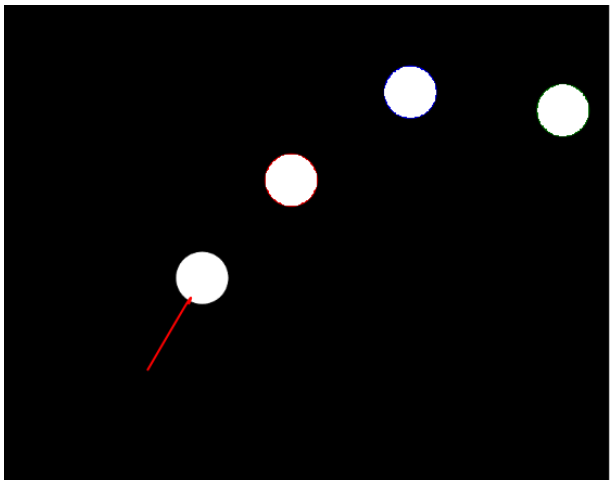

Fig. 1. Initial configuration of board

3) Task 3 :- Third task involved using an in-built face detection method to detect face of user and then stimulate a flappy bird game using movement of face. Flappy bird game was also supposed to be implemented without using any other library except OpenCV.

III. RELATED WORK

1) Task 1 :- Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. In Minimax the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible while the minimizer tries to do the opposite and get the lowest score possible. It is basically a exhaustive search of our game space. Intially I had difficulty in understanding the method of implementing game space but later I figured it out and also modified ideal game space in such a way that our agent tries to minimise the number of steps required to end the game.
2) Task 2 :- Since the balls provided in the figure were not exact circles so Hough circle detection was unable to detect circles in the diagram of board and hence I used contour detection and used concept of moments to find centers and average radius (assuming radius of all balls is approximately same which was visible in figure) of detected contours. To detect collisions I used equation :

$$d = 2 * (r)$$

(where r is radius of ball and d is distance between

centers of nearest balls) and used euclidean formula to calculate distance between center of balls.

3) Task 3 :- Object Detection using Haar feature-based cascade classifiers [2] is an effective object detection method. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it.
   Then task involved creating a flappy bird game using OpenCV functions only in which bird's movement is controlled by movement of head and to accomplish this part I created a circle around detected face and used center of circle to position the bird on screen. To create game environment I used drawing functions like drawrectangle, etc. To detect collision I used this formula :

$$d <= r$$

   where r is radius of falppy bird and d is distance between rectangle edge and center of bird.

IV. INITIAL ATTEMPTS

1) Task 1 :- Initial attempt and final approach is same for major part of solution and this task involved application of minimax algorithm and hence it was very standard approach. A major difference in implementation was in creation of game space as initially I imparted equal score to all wins despite number of steps required to win with that configuration of board. This made code slow and more vulnerable to loss as it was easier for human to figure out a way to win while minimax agent was trying win through longest path.
2) Task 2 :- Initially I used Hough circle detection function to detect balls and motive behind that was hough transform provides good amount of information about circles detected like their centers and radius of detected circles which was later needed to draw circles and detect collisions between balls but since Hough transform was unable to detect the circles I replaced them with contour detection.
3) Task 3 :- Implementation of first part was largely common for initial and final solution since I was supposed to used a built-in face detector to detect face of user. For creating game interface, I thought of using pygame but but since use of no other library was allowed so I decided to use drawing functions implemented in OpenCV to create game interface.

V. FINAL APPROACH

This should cover both columns (full page) excluding images

1) Task 1 :- A minimax algorithm is a recursive algorithm for choosing the next move in an n-player game, usually a two-player game. A value is associated with each position or state of the game. This value is computed by means of a position evaluation function and it indicates how good it would be for a player to reach that position. The player then makes the move that maximizes the minimum value of the position resulting from the opponent's possible following moves.
   For calculating score associated with all possible moves I used built-in function (of environment) to check state of board after a particular move was carried out and then positive increment was made to a move if it eventually ended up with win based on

$$score = 10 - n$$

   where n is number of steps after that move required to win.
   Also negative increment was done to those moves which led to loss of minimax agent and no score change was done if that move finally ended up with a draw. Finally, minimum of all scores (for minimising agent) and maximum of all scores (for maximising agent) was taken to calculate next move by minimax agent.

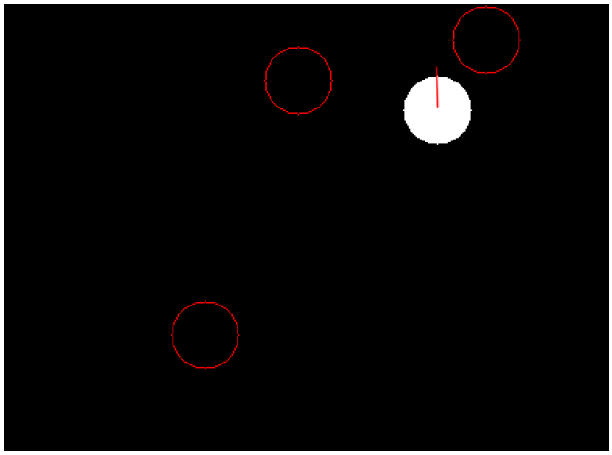

Fig. 2. Output

2) Task 2 :- Contour detection is in-built function in OpenCV used to detect contours in an image and moments are set of scalars that provide an aggregated measure of a set of vectors. OpenCV utilises moments associated with a contour are used to calculate features associated with that contour like its boundary length, area enclosed and center of contour. For completing this task, I used these two features of OpenCV to detect initial position of balls. Then I used Probabilistic hough lines to initial direction of movement of first ball as output of HoughlinesP contains slope of line found in image. Then ball motion was stimulated using standard parametric form of lines i.e.

$$x = x_0 + d * \cos(\theta)$$

$$y = y_0 + d * \sin(\theta)$$

   where x and y are co-ordinates of center of moving ball and θ is slope of line.
   Then while stimulating motion of balls, position of balls were continuously stored in an array as it was

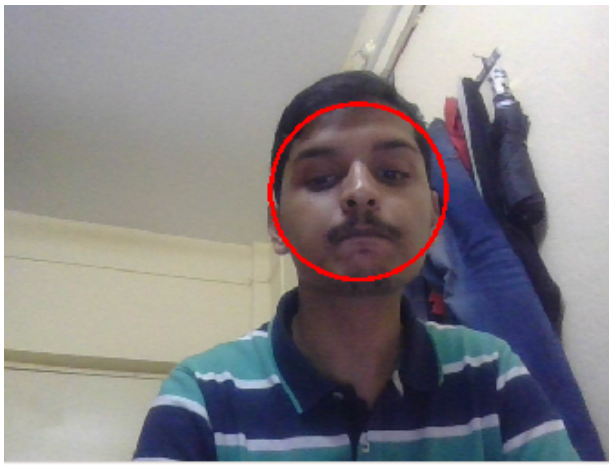

Fig. 3. Output of face detector

Fig. 4. Game interface

needed for balls to retain their position after each collision.

3) Task 3 :- Haar feature-based cascade classifiers [2] is an effective object detection method. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For face detection, training set is included with file of OpenCV and hence training part becomes very easier. To control motion of bird, center of face which was determined by using formula :

$$x = face.x + face.width * 0.5$$

$$y = face.y + face.height * 0.5$$

where face is object returned by face detector. To make game interface, drawing functions provided by OpenCV were used and frequency of obstructions was decided by using counter and a modulo function and height of obstructions was decided by random number generator.

VI. Results and Observation

1) Task 1 :- The algorithm works as expected and is able play against human agent efficiently. However, problem persists with this algorithm as there are certain set of moves which can easily defeat minimax agent and hence this algorithm can be improved by using advanced techniques.
2) Task 2 :- Ball movements were perfectly stimulated using this algorithm but it had issue motion of ball after it collides with a wall as algorithm is supposed to change angle of rotation by 180^0 and hence with some angles of approach two such rotations are required which leads to a weird motion of ball.
3) Task 3 :- Face detection part using Haar cascades work as expected and motion of bird is perfectly simulated by motion of head but due to problems with Haar cascade, face detection stops when even a small part of face goes out of field of view of camera and this lead to collision of bird with obstacle.
   This problem can be resolved by using better algorithm for face detection but since literature used in other algorithms was beyond my understanding, so I used Haar cascades. Also, distance between all rectangles was uniform due to type of implementation used and hence non - uniform distribution of rectangles can also be achieved by using better method.

VII. Future Work

1) Task 1 :- There is a problem with efficiency of minimax algorithm as minimax agent takes considerable amount of time in making its first move (as it has to check for large number of moves) . Also, defeating minimax agent is easy using a particular set of moves and hence it needs to be improved using better methods like genetic algorithm, etc.
2) Task 2 :- This task can be made more realistic by stimulating of motion of balls after considering the condition of oblique collision and also this algorithm can be made efficient by modifying image of balls so that Hough circles can detect them.
3) Task 3 :- As discussed earlier better algorithm for face detection can be used so that accuracy and efficiency can be improved. Also game interface can be made better by using libraries like SDL.

CONCLUSION

Task 1 was relatively easier one proper understanding of minimax algorithm was obtained and this was made even easier by the implementation of environment which contained all necessary functions. Optimisation problems are center of many modern algorithms and this task provided good hands on experience with one such problem.

Task 2 was also easy and it provided me with good experience of object detection which is needed in proper functioning of any autonomous robot.

Task 3 involved learning about face detection algorithms which was pretty good experience and is also useful for

future purposes as it recognition is important subject in field of computer vision.

REFERENCES

[1] E. N. Gilbert , "An optimal minimax algorithm" , Annals of Operations Research volume 4, 1985

[2] Paul Viola and Michael Jones , "Rapid Object Detection using a Boosted Cascade of Simple Features", IEEE computer society conference on computer vision and pattern recognition , 2001.