Aerial Robotics Kharagpur Documentation

Agni Keyoor Purani *

***Abstract*— A series of task related to computer vision, path planning and general programming was to be solved and this documentation specifies the approach for the taks. The task consisted of designing a Minimax agent, face detection and tracking, a basic physics simulation, path planning and error correction. These task given by Aerial Robotics Kharagpur where given to test our abilities and see how we come up with the most efficient and novel solution in limited amount of time (around 3 weeks). Some gave a really fun time and a great motivation to join the team. The task were very practical and the work done on it can be scaled for other works or can we used for ARK purposes too.**

I. INTRODUCTION

Aerial Robotics Kharagpur is a research group in Indian Institute of Technology, Kharagpur. In the process of selections for the freshers they give them certain tasks to complete as mentioned in the problem statement section. The problem statement consisted of five problems which had been classified into basic tasks and advanced tasks. The tasks hovered around the topics of algorithm design, computer vision and path planning. The tasks in general were very interesting and quite intimidating at first glance but while attempting them they seemed to be easy and fun. The basic tasks consisted of Tic Tac Toe AI, collision simulation and face tracking. Whereas the advanced task was to create a path planning algorithm for a 2D platformer and a way to detect true positives and correctly display a data with errors and without anomalies. Twenty six days were given to complete as many tasks as possible.

II. PROBLEM STATEMENT

The problem statement consisted in total of five tasks which were divided into 3 basic and 2 advanced tasks respectively. The tasks are as follows:

A. *Task 1.1*

We had to write an agent for a particular Tic-Tac-Toe environment called gym-tictactoe that works on the Minimax algorithm. If time permits then implementation of Q-value algorithm or Genetic Algorithms can be used.

B. *Task 1.2*

Given images of a billiard pool table with the cue and ball (Figure 1) to be hit indicated find pairs of collisions assuming perfectly elastic ball-ball collisions and ball-table collisions. To reduce the complexity of the problem it was assumed the the oblique ball to ball collisions can be ignored i.e. the colliding ball would come to rest and the other one would move with the same velocity.

*Ashwary Anand, Archit Rungta

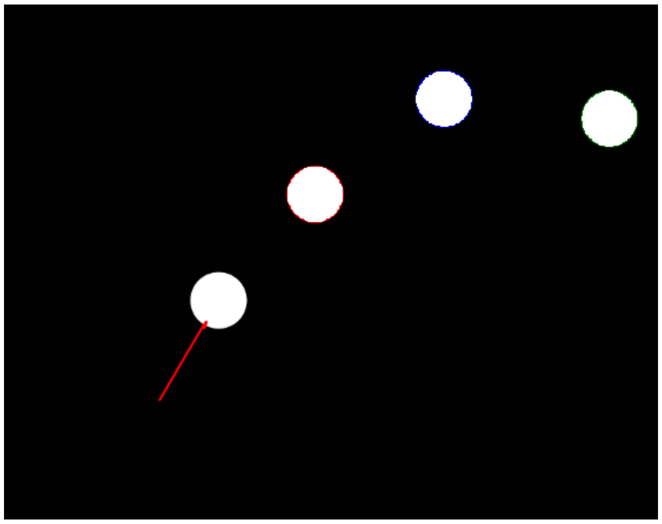


Fig. 1. An example input

C. *Task 1.3*

In this task the objective was to make an obstacle avoiding game using face detection and tracking. For the purpose of tracking we where not allowed to use the inbuilt functions available in OpenCV or any such computer vision libraries but we can do so for detection of faces. An example of the game(only the first 10 seconds are relevant).

D. *Task 2.1*

Given a game environment (Figure 2) and a basic strategy for it, we had to write a path-planning algorithm that makes the agent navigate and pick up a weapon and then fight back its opponent. The game a 2D platformer type game which consists of weapons, health packs, mines and ladder with gravity.

E. *Task 2.2*

In the final task we had to write a code for a military aircraft radar. In this we are given a list of detections in chronological in a .csv file, which is filled with false positives and missing true detection such as a bird or a civillian aircraft could be cosidered as an enemy aircraft. The work was to find out, in spite of all the incorrect detections the number of flight paths that exists and report the true trajectories.

III. RELATED WORK

There have been a lot of work on the tasks given to us especially in anomaly detection. As detection of anomaly is really though as there can be a true negatives or false

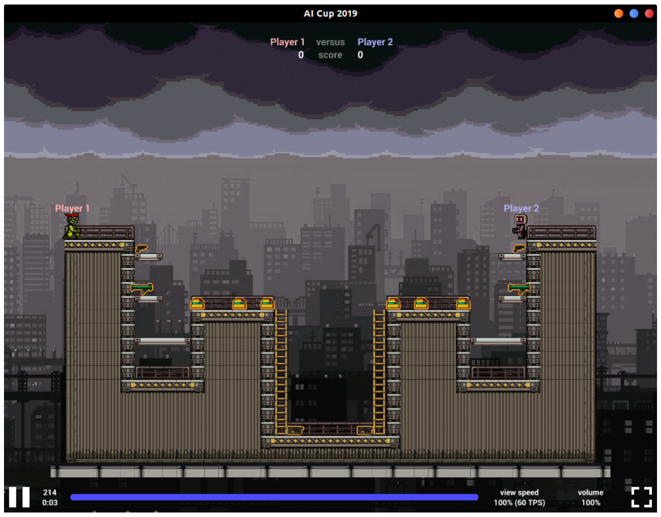


Fig. 2. The Game Environment

positives. These anomalies can cause a lot of problem and is common and solutions can be the use of some statistical filters like the Kalman filter[1][2]. Also there have been a lot of work on finding pathfinding algorithm and making our general grid based algorithms work in restricted environments like IDA* and A*[3]. The heavy restriction on the y-axis for this makes it harder to implement it.

Also, there has been a lot of work done for he improvement and designs of Minimax algorithm and finding the best way to implement it[4][5]. There also have been a lot of work on identification and training of human faces. Human face detection is hard to implement directly without training a classifier and in this case we would be using the Haar cascade classifier[6] and Object tracking is a very important tool used a lot especially in the area of aerial robotics and unmanned drones to follow a certain object.

Detection of cirlces and lines can be done using a lot of methonds and one of the most commonly used one is the Hough transform. Any a lot of work is done in identificaation of various shapes and all these can be extented to any shape. A Hough transform is a very robust way to detect shape even if some points are measured and works well in a lot of situations[7].

IV. Initial Attempts

A. Task 1.1

The first task was pretty straight forward in implementing the agent for the game of Tic Tac Toe. The first attempt was to implement a way in which the agent can respond when it is his turn on any remaining position. Later on the Minimax algorithm could be implemented. A Minimax agent works by finding out all the possibilities and playing against itself and then making a move depending on which result has the maximum score. A recursive implementation was the best way.

B. Task 1.2

For the second task, the first step is to identify the balls and the first ball to move with its velocity. This could be done with Hough Transforms for line and circles respectively and after their detection we have to give the first ball the velocity as the same direction and then apply a collision detection algorithm that works by seeing if the circles have touched or not and apply velocity.

C. Task 1.3

The first job is to figure out where the face is by detecting it by using a pretrained classifier, the Haar Cascade classifier and then track the face. The first step for tracking was to take the detected face as a separate image and then searching for that face by template matching. This worked by failed very easily as either high threshold for the template matching would not find the face or a low threshold would some times get confused between a real face and some other unwanted object in the frame. This could be fixed by updating the face tracked every frame so each frame would have a face at least similar to that of the original face.

D. Task 1.4

To navigate in any game environment and pickup a weapon and fight, one must implement a path finding or planning algorithm in any manner, since a lot of places adopt A* and IDA* algorithm my first attempt revolved in implementing an A*. But immediately a problem was encountered knew that a simple A* would fail as the game environment wasn't a simple graph that a general A* is used for and had to work around this problem.

E. Task 1.5

The initial attempt on the final task was to try to plot the data given without and=y change too see how corrupted the data is or how many false information is there. This seemed really similar to anomaly correction and had an idea to implement a statistical filter like the Kalman filter.

V. Final Approach

A. Task 1.1

The implementation a a very simple Minimax agent in the first task will be what is to be done. Since, Tic Tac Toe is a very simple game, the Minimax agent will always either draw the match or never lose. The recursive Minimax algorithm is very simple.

B. Task 1.2

In the second task, the first step was to identify the cue and the balls correctly by using Hough transforms using the OpenCV's built-in **HoughLinesP()** and **HoughCircles()** functions which gives a list of start and end coordinates of lines and a list of centers and radius of circles respectively (Figure 3). Then using the coordinates of the line we can find out the nearest circle to the cue and then give it a velocity of magnitude one in the direction of the cue which can be figured out by the coordinates of the cue.

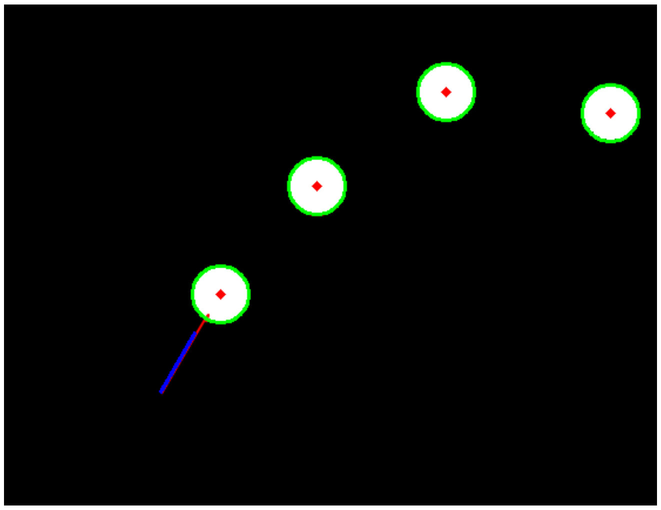


Fig. 3. Detection of cue and balls

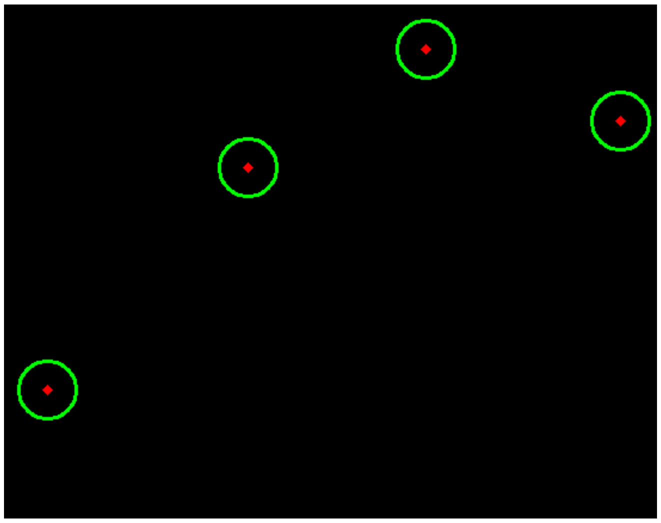
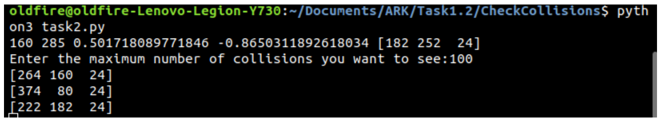


Fig. 4. Simulation in progress

Using this we can increase the X and Y coordinate of the ball undergoing motion by the Velocity in the X and Y directions respectively which causes the motion of the ball. The X and Y coordinates of the ball undergoing the motion is assumed to be **float** and will be later type-casted to **int** or else the ball may not move. For the collision check, we can simply find the distances between every circle with respect to the moving cirlce and see if the distance is less than or equal to two times the radius. If it is then they have collided and the collided circle's X and Y position will be incremented by the same velocity as the incident ball. For the collision of circle with the boundary, we can see if the distance between the center of circle and the edge is equal to the center of the circle if so then the velocity direction of that axis is reversed. The simulation is shown in Figure 4.

The terminal initially outputs the nearest end of the cue to the nearest ball, the velocity of the ball and the coordinate



```
oldfire@oldfire-Lenovo-Legion-Y730:~/Documents/ARK/Task1.2/CheckCollisions$ pyth
on3 task2.py
160 285 0.501718089771846 -0.8650311892618034 [182 252  24]
Enter the maximum number of collisions you want to see:100
[264 160  24]
[374  80  24]
[222 182  24]
```

Fig. 5. Terminal Output

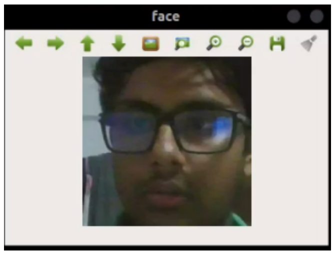


Fig. 6. The ROI for template matching

and the radius of the first ball. Then takes an input from the user about the maximum collisions he wants to see. For every collision the striking balls coordinate is displayed at the moment of striking as shown in Figure 5.

C. Task 1.3

For the third task, a pre-trained classifier called Haar cascade classifier was used for the detection of face. Once a rectangular region of interest(which will be refereed to ROI from now) Figure 6 *i.e,* the face area was stored separately and was used for the purpose of tracking. The ROI is then used for template matching and the region with the maximum probability is then taken as the center of the face and a new ROI for the same size is taken as the new ROI to prevent loss of tracking as small changes can evolve on. In addition to this, to confirm that we are tracking a face we compare the new ROI with the original ROI to figure out whether they are not drastically different. If they are drastically different then detection if performed again and the process continues with the newly obtained ROI (Figure 7).

The game (Figure 8) in general is very simplistic in nature. It consist of rectangles of cyan colour appearing horizontally with gaps of fixed size appearing at random position. The player is a orange circle whose center is determined by the the center of the ROI as tracked. The objective of the game is to move your head such that you have to avoid hitting the rectangular bar and score the highest. The score is calculated by the number of bar that appeared on the screen and not the number of bars you pass through. The main reason for this is that the player can exploit the degree of freedom of moving up and down to score as much as he can and so the implemented score is a better way to measure in this case.

D. Task 1.4

For the 2D platformer game our objective is to pickup a weapon in the game (preferably the one that does the most damage in the least amount of time and has a blast so that the enemy is damaged even if we miss by a little bit) and then fight an enemy while surviving. So, in the beginning our

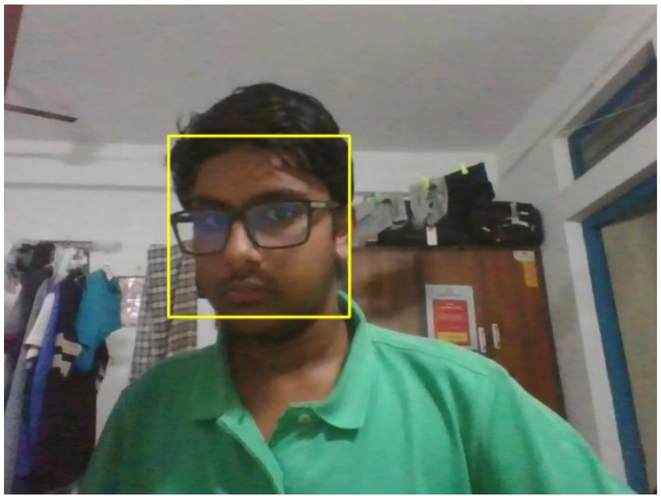


Fig. 7. Face being tracked

Fig. 8. The Game

character needs a weapon and then has to find the enemy. If the characters health is low he has to find a health pack and if a mine is near him, it is always a good idea to pick it up and place it somewhere. In all the above case we need to figure out a path planning algorithm where the target or the goal could be the position of a weapon, the enemy, health pack or a mine.

The path planning algorithm that was being planned to use was the A* algorithm. This game having gravity makes the implementation for such algorithm hard but there is always a work around. Instead of having a 2D grid of possible nodes we can have a 3D grid of possible states. The X axis of the grid can represent all the possible paths that the character can take in the X direction or the horizontal direction. Now, let us consider the character can jump up to a maximum of 4 bloc direction then the Z axis with represent all the nodes in the X-Y plane when the character has jumped 1 block high or is 1 block higher then the normal level. And the Y axis represent the possible Y coordinates that the character can go. This was we have solved the problem of the jump in the game. The reason of not choosing and making the RRT work like this is because for every branch we create in this 3D grid it is difficult to map the branches when going in the Z - direction. Though the implementation was not fully completed the final approach seems to be correct and should work

E. *Task 1.5*

Task 1.5 is a task that seemed the most difficult out of all the given tasks. In order to figure out the actual flights and interpolated the missing data for the fight a lot of mathematics has to be involved. The idea of using a Kalman filter was a possible approach for the solution. The Kalman filter is an efficient recursive filter that estimates the internal state of a linear dynamic system from a series of noisy measurements. After obtaining the final trajectory, the final data can be plotted on a 3D plot to see the trajectories.

VI. Results and Observation

1) The Minimax algorithm or idea is such a simple idea in game theory but can be used in a very powerful way and can help us make agents that can mimic thinking. In this case, when tested against a lot of real people several times, it has never lost.
2) The collision detection program was tested with five different input images and worked perfectly for four of them and was unable to detect the cue for the final one but was able to detect the circles.
3) The template matching idea worked really well and gave very little or no mismatching over the period of testing. While testing out the program with no tracking and only detection, the stream seemed to lag and slow down. This shows that detection takes up a lot of processing power and hence we should you tracking once the object has been detected.

VII. Future Work

A. *Task 1.1*

For the first task, instead of implementing the Minimmax algorithm we can implement more advanced methods like the Q-value or the any genetic algorithms as they are more faster than the Minimax algorithm as the Minimax tries to explore all the possible game states.

B. *Task 1.2*

In this task, there was an assumption that when the balls collide with one another the colliding ball stops and the ball with which it collided moves on with a constant speed. But, in the real world the physics don't work out like that and so the next work is to implement real life oblique collision physics and simulate it. Futher, instead of assuming perfectly elastic collisions we can take in the coefficient of restitution as an input ans then simulate the model. Adding friction and be the final work that can be done to simulate the entire model.

C. Task 1.3

Since the implementation of the face tracking was supposed to be implemented by not using any of the OpenCV's built-in method and due to the time restriction imposed, the tracking algorithm may not be the most efficient and accurate method that exist and can be tweaked a little more to perform better than its current state. The second thing that can be worked on is the game UI and its aesthetics as it is not visually appealing.

D. Task 1.4

The future work is to fully implement the modified A* algorithm and test it to see how it behaves against humans and other models that currently exists.

E. Task 1.5

The anomaly correction implementation has to be finalized and implemented. After the successful implementation of the correction a 3D plot of the flight trajectories can be plotted to see the path followed for different aircrafts that appeared on the radar.

CONCLUSION

Each part of the task, even though it is implemented for a specific task, it can be generalized and used every where. The Minimax agent can we used for any sort of game or work that requires us to take up a choice that can result in a good or a bad outcome. The detection of circles and lines can be extended and can be used to identify symbols from the sky. In case of building a drone that follows a human or any object autonomously we can implement tracking algorithms like the one designed in this case. Also, path planning is essential for automatizing and robot or drone as it has to figure out how it has to navigate through the space. Finally, in any robot, drone or machinery there always could be a fault in the output of sensors, that could lead to inconsistent performance or outputs and hence anomaly corrections and statistical filters are absolutely necessary.

REFERENCES

[1] Bin Feng, Yuan Gao, *Development of Strategy Software and Algorithm Simulators for Multi-Agent System in Dynamic Environments*, Link (2013)

[2] Augustin Soule, Kavé, Nina Taft, *Combining Filtering ans Statistical Methods for Anomaly Detection*, Internet Measurement Conference pg. 331 (2005)

[3] Peter Innes, *Effects of Multiple Gravitational Fields on Pathfinding in a Simple Two Dimensional Platformer*, Research in Game Behaviour Vol. 1 No. 1 (2014)

[4] Vardi, A., *New minimax algorithm*, J Optim Theory Appl 75, 613–634 (1992)

[5] Igor Roizen, Judea Pearl, *A minimax algorithm better than alpha-beta? Yes and No*, Artificial Intelligence Vol. 21, Issues 1–2, pg. 199-220 (March 1983)

[6] R. Padilla, C. F. F. Costa Filho, M. G. F. Costa, *Evaluation of Haar Cascade Classifiers Designed for Face Detection*, International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:6, No:4 (2012)

[7] J. Illingworth, J. Kittler, *A Survey of the Hough Transform*, COMPUTER VISION, GRAPHICS, AND IMAGE PROCESSING 44, 87-116 (1988)