Tic Tac Toe with MyCobot

Karteek Menda
School Of Manufacturing Systems and Network, Ira A Fulton School, Arizona State University

Article Info

Article history:

Received December 1, 2023
Revised December 3, 2023

Keywords:

mycobot
Atom
Color recognition
Tic Tac Toe
Minimax Algorithm
Pick and Place

ABSTRACT

The myCobot 280 m5 is a cobot which I use in this project for making the cobot play the tic tac toe. Tic-tac-toe is a classic two-player game played on a 3x3 grid. The cobot with its capabilities plays the tic tac toe with users. The project is all about employing the best move so that the cobot wins and this can be achieved by employing a decision-making algorithm (minimax algorithm) which is generally used in two player turn games. Overall, this algorithm is used in game theory to minimize the loss in the worst-case scenario. The cobot with its capabilities needs to be integrated with the necessary logic of the above algorithm so that it can mimic the decision taken by the algorithm in its turn.

Corresponding Author:

Karteek Menda
School Of Manufacturing Systems and Network, Arizona State University
Tempe, Arizona, United States
Email: kmenda@asu.edu

1. INTRODUCTION

The objective of the project is to develop a logic that allows the users to play the tic tac toe game with the help of cobot capabilities i.e. color recognition, pick and place. A 3x3 grid is used to play the standard two-player game tic tac toe. Every player marks a square in turn, usually with a single letter "X" for one player and the letter the letter "O" for the other. The goal is to earn three of those points in a row—vertically, in the horizontal direction, or diagonally—before your rival does so.

The camera module is open which scans the camera zone area for the aruco markers and then displays the 3x3 grid. The X registers the algorithm(computer) move which will be passed to the cobot for the move to be done physically. And the "O" registers my move. So here the most important is the "X" move which is nothing but the decision made by the minimax algorithm.

The process involves a combination of usage of AI algorithm, Cobot capabilities, OpenCV, and etc. which will combinedly help us to execute this project. The below steps clearly explain in detailed report of how all these are integrated with each other to get the task done.

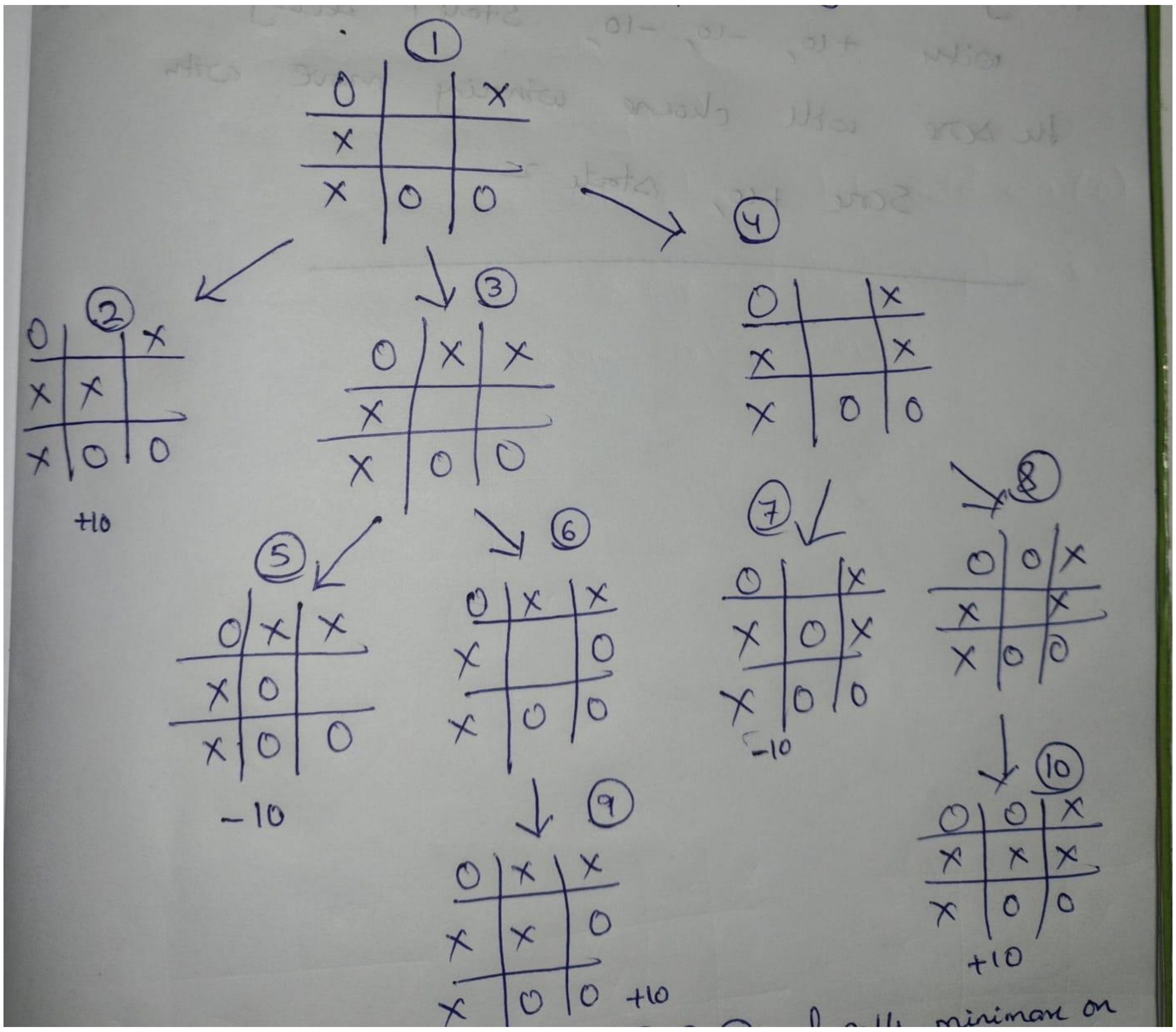
2. METHOD

Setup:

Installing Pymycobot and the necessary Python software is the first step. You must install myCobot for Linux after installing Python. Next, you must download and burn the transponder while the base is linked to the display. Next, open myStudio, download Atom, and flash it. To stop the robot from hitting the camera module, simply install the required software and then set all motor settings to 0 degrees. After that, use the calibration code to calibrate. In addition to it, install the opencv necessary packages as well. Make sure the environment is properly set for the error free execution. Remember this project is nothing but establishing a nice integration of Robotics, Artificial intelligence, and Computer Vision.

Minimax Algorithm:

A back and forth between both players, along with the player deciding whose "turn it is" it is, is crucial to the Minimax algorithm to be able to determine which player has the best result. The other player then selects which of its possible actions has the lowest possible score, calculating the scores for each of the possible actions. The player taking turns subsequently attempts to maximize its score, and so on along the move tree into an end state, deciding the scores for the moves of the opposing players.



Walking through the algorithm execution.

1. In state 1, it is now the X's turn. X runs minimax on states 2, 3, and 4 upon generating them.
2. Since the match is at its final state, the second state gives the score of +10 to the state 1's score list.
3. Without being in the end states, states 3 and 4 generate states 5 and 6 and use minimax on them, whereas states 4 create states 7 and 8 and use minimax on them.
4. State 5 places a score of -10 on the rating list of state 3, while state 7 performs the same, giving a score of -10 on the score list of state 4.
5. States 6 and 8 add a rating of +10 to the move lists of states 3 and 4 as they generate the only feasible moves, which are end states.
6. As it is O's turn in states 3 and 4, O is going to attempt to find the smallest score; given the option, the two states 3 and 4 will result in a score of -10.
7. State 1 will choose the winner's move with scoring +10 in a bid to best utilise the score, while state 2 will fill up the score list for states 2, 3, and 4 with +10, -10, and -10, accordingly.

To summarize, the minimax algorithm works on game tree representations, backward recusriosn(minimax), and decision making. While playing a game like tic tac toe, the algorithm evaluates all of potential moves, gives a numerical value for every game state, and chooses the move that gives the participant the best outcome at the time after taking the opponent's best response into consideration.

The logic encoded in the script consists of some methods related to minimax algorithm which are integrated.

a. Creation of tic tac toe board
   1. board_initialization(): as I set the columns and rows to 3 each. There by I want to create a board of a 3x3 matrix shape which represents each cell in the game.
   2. draw_board(): takes an image as input and draws a tic-tac-toe board on it. The board is divided into a grid with horizontal and vertical lines. The black colour lines are used with a line thickness of 3 pixels.
   3. print_board(): displays the tic tac toe board in the console.
b. Centers for the blocks in the game
   1. block_centers() got these coordinates by collecting all the 9 block coordinates and their centers.
c. Getting the pickup places and the other relevant coordinates.
   1. pickup_place_cords() : Giving the coordinates to the cobot to pickup the block for its(Computer) move. And placing the blocks at the mentioned centers which I declared in the block_centers().
d. Minimax algorithm.
   1. minimax() : function is a recursive implementation of the minimax algorithm for a Tic-Tac-Toe game. The function evaluates the game state and returns a score based on whether the 'O' player or the 'X' player is maximizing their score.

      Base Cases:

      If the 'O' player has won, the function returns 1.

      If the 'X' player has won, the function returns -1.

      If the board is full and there's no winner, the function returns 0.

      Recursive Cases:

      If it's the turn for the maximizing player (currently 'O'), the function iterates through available pickup_place_cordss, simulates each pickup_place_cords, and calls itself recursively with the updated board and the next player's turn ('X'). It then updates max_eval with the maximum value obtained from these recursive calls.

      If it's the turn for the minimizing player (currently 'X'), the function does the same but minimizes the values by updating min_eval with the minimum value obtained from recursive calls.

      Undoing pickup_place_cordss:

      After each pickup_place_cords simulation, the board is reverted to its original state by resetting the cell to an empty space.
   2. find_best_pickup_place_cords() : AI algorithm for playing Tic-Tac-Toe. It uses the minimax algorithm to evaluate and choose the best pickup_place_cords for the 'O' player.
   3. cross_check_winner() : Checks which player has won the game. In order to do that, need to check the rows and columns and also the diagonals.

4. RESULTS AND DISCUSSION

As a part of this project and as per the explained logic above, the Aruco's (markers) were detected upon which the grids (tic tac toe game board) get activated and displayed on the screen. Place the green blocks in the bin from which the cobot picks up. Apart from it, I need to have the yellow cubes. So, as a part of the first move, I need to put the yellow block in one of the 9 grids, and the camera module identifies the block and based on the minimax algorithm's next decision of making a move, the cobot pickups the green block from the bin and puts it in the grid as guided by the minimax algorithm. Now, it is my turn to pace yellow block in the grid which ever I want. And then the cobot takes it move and this process continues on until an end result is achieved. In my demonstration, the game was won by the computer as it followed the decisions from the Minimax algorithm.

Cobot Serial Number : ER28001202200478

Raw Video Link : (https://www.youtube.com/watch?v=HL6Okvrt6iw)
Video presentation: (https://www.youtube.com/watch?v=x1cC3XHAWkw)

5. CONCLUSION

The robot advances to the green block and picks up the green block and put it in the respective coordinates as directed by the logic by the minimax algorithm. In my turn I need to keep the yellow block in the grid which ever I want to place. The process continues on. This demonstrates the cobot's capabilities by using the color recognition. The cobot's capabilities, decision making algorithm (AI Move), and the computer vision all combinedly comes together to get this project done. Thus, we have met all the objectives declared at the beginning.

ACKNOWLEDGEMENTS

I thank Prof Redkar, teaching assistants for taking the demonstrations.

REFERENCES

The main references are
[1]https://www.hackster.io/Elephant-Robotics-Official/newly-upgraded-and-user-friendly-ai-kit-2023-a23669
[2]https://docs.elephantrobotics.com/docs/gitbook-en/2-serialproduct/2.10-AIkit2023en_320/3-color_recognition.html
[3] https://www.geeksforgeeks.org/minimax-algorithm-in-game-theory-set-1-introduction/
[4] https://www.javatpoint.com/mini-max-algorithm-in-ai
[5] https://www.youtube.com/watch?v=KU9Ch59-4vw
[6] https://www.youtube.com/watch?v=SLgZhpDsrfc

BIOGRAPHY OF AUTHOR

Karteek Menda is a Robotics GRAD Student at ASU, Ex - Machine Learning Engineer, and a Machine Learning Blogger. Experienced data science professional with extensive knowledge of building data-intensive applications and overcoming complex architectural and scalability challenges. Proficient in predictive modelling, data processing, and data mining algorithms, as well as scripting languages, including Python and C++. A passion for Artificial Intelligence coupled with a comprehensive understanding of machine learning concepts and other related technologies. Through the design, development, testing, and deployment of highly adaptive diverse services, able to transform business and functional qualifications into substantial deliverables. I can be contacted at email: kmenda@asu.edu