CPTS 483 Final Project Proposal

By Kris Hahn and John Lindquist

Introduction

We are undertaking the challenge of programming a robotic arm to play Tic-Tac-Toe. This project serves as a practical learning experience for our team of Mechanical engineers, allowing us to dive into the world of robotics programming and ROS2. Tic-Tac-Toe, a classic game of strategy played on a 3x3 grid, provides a clear and manageable objective. Our 'Problem Formulation' outlines a conservative yet scalable approach, acknowledging our initial learning curve while ensuring a solid foundation for future development.

Problem Formulation

Our goal is with the Franka FR3 arm move Tic-Tac-Toe pieces on and off a grid so that two users can play Tic-Tac-Toe. The primary goal is to first make an interface for a user to interact with to tell the robot where a piece needs to go. The second goal is to convert the play locations to a 3d spot that an arm can go to. The final goal is to set up the motion paths/ planning for picking up and placing down the pieces. By doing this we hope to learn how to use ROS2 more effectively and to play Tic-Tac-Toe robotically.

Method for completion

To try and complete this project we used Python as our main scripting language due to our familiarity with it. To create the interface to select where a play is made, we used a package called “tkinter” any it ended up Showing what cells were selected and, in the background, we used an array to show where the elements are sitting. After we had this basic interface, we then started to adapt it to give us the locations on an x y grid of where to pick up and place down an element. When placing an element into the “Play grid” we pull the first element from the “Standby grid”, and we pull from the grid in a specific order to pull an “X” then and “O” respectively. By pulling from this grid in a specific order we know how we leave this grid so when it is time to reset, we know the first spot is “X” then “O” etc. we also are keeping track of how many plays have been made and by doing this we are able to know how many pieces need to be placed back. When placing the pieces back we first search the “Play grid” array for and “X” and when we find one, we then tell the arm to grab it and place it in the start grind in position 1. Next, we search the “Play grid” array for an

“O” then go and grab it place into the “Standby grid” rinse and repeat. By using this method, we can reset the “Play grid” from any state or order and return to a known state making it ready for another play. So, once we knew which piece needed to move by using the grid locations, we next needed to know the “X” and “Y” locations relative to the robot’s base. To do this we created an if statement that when you are pulling an element it gets the position ID and gives, and “X” value based on it and a “Y” Value. The “X” and “Y” values are in mm and are calculated by using a fixed offset to the center of the grid. We also know that the center of one element to another is 66mm so with the “X”, “Y”, and the offset we can find the position to pick up and place down the elements in mm. by doing this it also allows for the grids to be placed in a different place and all that needs to be changed is the center of the grid for the arm to get to it. Because we know in “X” and “Y” where the elements needed to go, we then are handing it off to a ROS2 Package called Moveit2 to do the Kinematics of finding the joint angles that can then move the Franka FR3 in simulation.

Experiment

We have developed a two-player Tic-Tac-Toe program with the following features: players alternate turns selecting a tile on the game board, which is then marked with their chosen symbol (X or O). The program tracked the x and y coordinates of the selected tile relative to the base of a robotic arm. Game pieces were sourced from a secondary storage board and placed onto the main playing board. The program incorporated win condition checks, announcing the winner when a player achieved three consecutive symbols in a row. In the event that all spaces on the board were filled without a winner, the game was declared a draw. Upon completion of a game, the program returned the game pieces back to the secondary storage board

In addition to the program, we had a simulation of the arm as well as a 3D-printed Tic-Tac-Toe game board and 'X' and 'O' pieces with the hope of using the real arm. The locations on the game board were defined by their x and y coordinates relative to the robotic arm's base. We successfully visualized the robotic arm in RViz and possessed the capability to control its joints for movement. However, we experienced difficulties when attempting to import the game elements into Rviz

However, we faced challenges in integrating the Tic-Tac-Toe program with the robotic arm, specifically in commanding the arm to move to the determined X and Y coordinates on the game board. We attribute this problem to the moveit2 library not being updated for the current iteration of ROS2 in python. The theory is that because we have the

x and y coordinates, we should just be able to feed them into the moveit2 library and it would do all the motion planning to move the pieces to the desired location.

GitHub Link: https://github.com/Engineerboy02/CPTS483_FinalProject

Works Cited

"Examples." *Examples - MoveIt Documentation: Rolling Documentation*, moveit.picknik.ai/main/doc/examples/examples.html#using-moveit-directly-through-the-python-api. Accessed 29 Apr. 2025.

"Franka_ros2." *Franka_ros2 - Franka Control Interface (FCI) Documentation*, frankaemika.github.io/docs/franka_ros2.html. Accessed 29 Apr. 2025.

Importing an STL File into Rviz - Ros Answers Archive, answers.ros.org/question/318307/. Accessed 29 Apr. 2025.

"Moveit Quickstart in Rviz." *MoveIt Quickstart in RViz - MoveIt Documentation: Rolling Documentation*, moveit.picknik.ai/main/doc/tutorials/quickstart_in_rviz/quickstart_in_rviz_tutorial.html. Accessed 29 Apr. 2025.

"Moveit Setup Assistant." *MoveIt Setup Assistant - MoveIt Documentation: Rolling Documentation*, moveit.picknik.ai/main/doc/examples/setup_assistant/setup_assistant_tutorial.html. Accessed 29 Apr. 2025.

"Ros 2 Documentation." *ROS 2 Documentation - ROS 2 Documentation: Humble Documentation*, docs.ros.org/en/humble/. Accessed 29 Apr. 2025.

“Ubuntu (Deb Packages)□.” *Ubuntu (Deb Packages) - ROS 2 Documentation: Humble Documentation*, docs.ros.org/en/humble/Installation/Ubuntu-Install-Debs.html. Accessed 29 Apr. 2025.

“Using Turtlesim, ROS2, and RQT□.” *Using Turtlesim, Ros2, and Rqt - ROS 2 Documentation: Humble Documentation*, docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Introducing-Turtlesim/Introducing-Turtlesim.html. Accessed 29 Apr. 2025.

Wiredworkers, www.wiredworkers.io/wp-content/uploads/2019/12/Panda_FrankaEmika_ENG.pdf. Accessed 30 Apr. 2025.