Tommy Le

Alfredo Weitzenfield

Independent Study in Computer Science

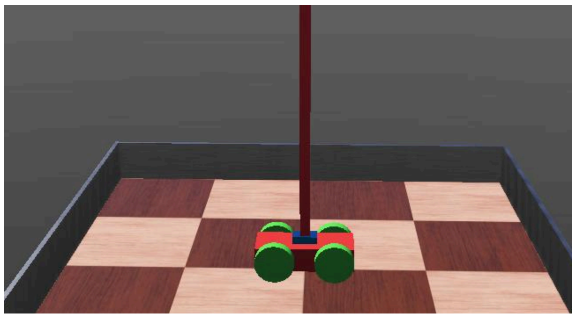
31 July, 2023

Using Reinforcement Learning Robot to Traject a Ball Towards a Goal

During the Summer of 2023, the object was to design a robot to teach itself how to kick a ball to a goal using Reinforcement Learning. The robot would develop a kicking behavior while tracking the ball's trajectory. The current project uses Chance's FAIRIS Robot to take advantage of its sensors and uses Deepbots as the primary Reinforcement Learning Framework.

Going through Tutorial of Deepbots with Cartpole Robot

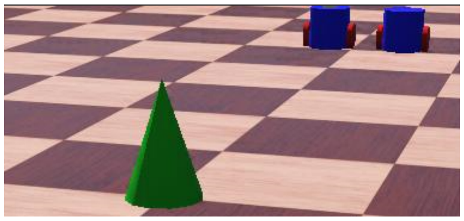
DeepBot is a framework that operates under OpenAI gym's environment but uses webot's robot/supervisor to accept observation and action values. This enables a relational loop between environment and agent for Reinforcement Learning. Inside each project's "controller" folder should include the "PPO_agent.py" and the "utilitles.py". "PPOAgent" accepts a user's observation and action sequence to implement Proximal Policy Optimization in secret, and" Utilities" normalizes the range of numpy observation data when necessary. Overall, the user only modifies the prebuilt Deepbots functions to indirectly implement RL: get_observation(), get_Rewards(), is_Done(), solved(), and apply_action(). Learning through the github tutorial provides a background on how to implement these functions without needing to understand the complexity inside PPO_agent. The Cartpole Robot was a RL robot, which learned how to balance a stick on top by moving backwards and forwards. Inside the "Cartpole" folder and at the bottom section of the "robot_supervisor_controller.py" file, Environment RL runs under a loop, where the Cartpole's observation includes the cart's position, cart's velocity, and the pole angle and the Cartpole's Actions is a space which holds a finite number of actions. Currently the robot is



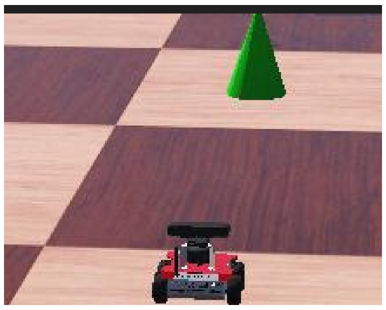
limited to moving backwards and forwards. The Cartpole “apply_action” function would perform a movement from the action_space (forwards or backwards). The Cartpole’s “is_Done” function would end an episode if the robot detects its position being out of bounds relative to the field, or if the pole is too slanted to the point it would fall (based on the pole angle).   The cartpole is rewarded based on the number of episodes. Because the environment doesn’t end until the pole “drops”, the robot will continue to learn and balance the pole and will be rewarded based on the time lapse instead of a standard point system. Because this project was an introduction to Deepbots, the objective was to complete the tutorial and have experience using this framework. Inside the “Video File” folder and “CartPole Robot”, the program runs without any errors.

Implementing Robot FAIRIS Controls and Deepbots Framework

After the CartPole experiment, a two wheeled differential drive robot was made from scratch. The entire robot could move based on the directions of the wheels, and the camera on top can detect objects with “recognition enabled”. This can be seen inside the “Learning Webots” folder. Under “Worlds” should be “First_Robot.wbt” and under “Controllers” should be “drive_my_robot.py” This idea was soon discarded, when it was recommended to use the “FAIRIS-Main” folder as the file included the



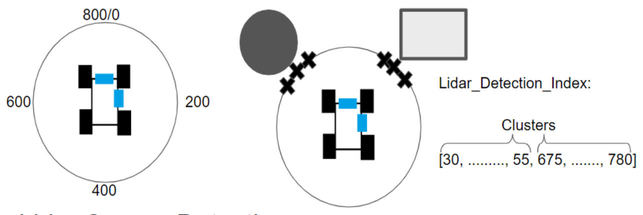
FAIRIS robot program with basic controls and additional supervisor functions which enable the robot to alter its environment. For the soccer project, a user could reset the position of the robot, ball and goal to various locations, enabling the agent to kick more accurately by being given different scenarios or shooting angles. The plan was to implement the Deepbot syntax onto FAIRIS’s Supervisor functions to create a RL environment for FAIRIS robot; however, some of the functions inside Deepbot would interfere with the premade “FAIRIS–MAIN” code. Additionally, there was an issue importing the FAIRIS files because it was incompatible with Visual Studio Code



IDE. The issue was temporarily resolved by creating a new class, c*lass FAIRIS (RobotSupervisorEnv)*, which the program would inherit the Deepbot methods “RobotSupervisorEnv ". The basic motor control, sensor enabling, and additional calculations from the previous project had to be restructured to operate under this class (adding “self” or renaming variables). The next step was to create a new Deepbots RL project that moves toward a detected object. The purpose was to confirm the new syntax of Deepbots/FAIRIS class was operational and to obtain experience with Deepbots, without any specific directions. This can be seen inside the “Learning Webots” folder. Under “Worlds” should be “empty.wbt” and under “Controllers” should be “DeepBot_FAIRIS.py” The primary observation points were the values of the four distance sensors, pointing diagonal from the robot. The selected action space included moving forwards, backward, and rotating left and right. The done function resets the environment when the robot scores more than 100 points or when it has passed 60 minutes. When a robot moves closer to an object, the distance value should be less than before. Whenever the distance sensor gets smaller the robot gets rewarded. If the robot moves towards an object, the sensor would detect a smaller distance value and obtain more rewards overtime. Theoretically, this method would teach the robot to move towards a detected object. In the video “Deepbot_FAIRIS”, the robot either makes turns or moves forward sleights towards the green cone. As sensors are rotated or translated towards the direction of the cone, it would receive more reward and continue these actions. Though the program did print out “Task not solved” with its low RL score, the project was still able to successfully run under this new FAIRIS/Deepbots format, and the environment did reward and learn certain actions. With the current task finished, the focus was to apply this structure for Reinforcement Learning soccer.

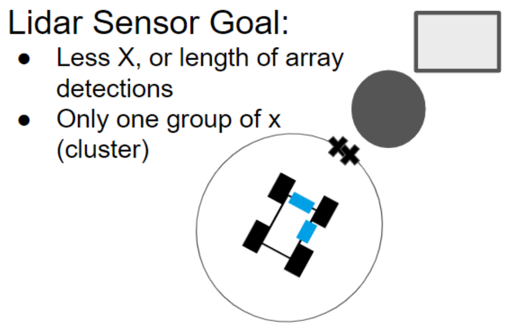
Reinforcement Learning Soccer

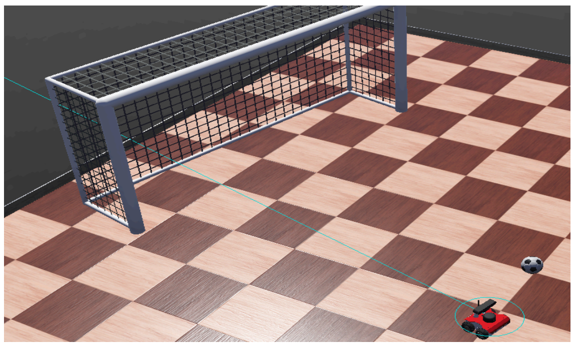
The official project under the “ReinforcmentLearning_Soccer" folder, would utilize the FAIRIS Robot controls from “Basic_FAIRIS_Controls.py" from FAIRS_MAIN, while utilizing the Deepbots Reinforcement Learning Framework from the “Cartpole" and the “DeepBot_FAIRIS" folders. The current objective was teaching the robot how to align itself with the ball and the goal to curve shots because the goal and the ball aren’t positioned at any angles for any curve shots. The main sensor for the robot was the Lidar, enabling the robot to detect distance under an entire radius. Normally the Lidar would print out an array of distance values from index [0,800]; however, by adding a conditional statement to append “indexes” where the object is detected, the sensor would return a smaller array with the “indexes” as the detected



Lidar Sensor Detection

position relative to lidar. As stated before the index ranges from 0 to 800, where the number present represents radial points which the lidar detects an object. From 790-800 and 1-10, would represent an object in front of the robot, while 200-300 on the far right of the robot. “InsideReinforcmentLearning_Soccer” folder, FAIRIS_controller.py”controller, would use the length of the lidar detections and the number of clusters with lidar detections as the Observation values. Because the index represents the spatial positions from object detection, having a greater list could mean a large number of objects spread out the radial field of view. Clusters represent groups of numbers within the lidar range. When the ball and goal are far apart, the lidar would register two groups of numbers instead of one long array. For example: [1,2,3,578,579,580] has two objects/groups from 1-3 (object around top north)and 578-580 (object at the left). The reward system encourages the robot to only detect one cluster and limit the size of the lidar detection. Doing so enables the robot to become more aligned with the ball and goal. Inside the program’s reward function, 1 point is given when the sensor detects less clusters than before and 0 points if unable to; however, once it only detects one cluster, the reward would be based on size of the detection array with more rewards with a smaller detection size. The “done” function prevents the robot from learning any useless action by stopping the robot if it detects more clusters than before. There is an additional stop function to prevent the robot from running an episode for more than 60 minutes. Finally the action space enables the robot to choose 7 different actions: forwards backwards, stop, small left or right turn, and large left and right turns.


Lidar Sensor Goal:
• Less X, or length of array detections
• Only one group of x (cluster)



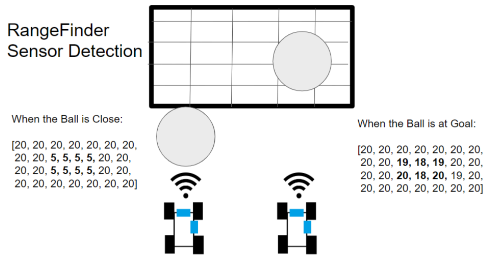
RESULTS

The expectation was for the robot to move throughout the arena and relocate at the most optimal position to realign with the various objects; however, running the “Soccer_Field.wbt” world file with “FAIRIS_Controller.py” python file (both inside “ReinforcmentLearning_Soccer" folder) produced unexpected outcomes. Inside the “Video” folder, Reinforcement Learning 1 and 2, both scenarios show how the robot initially would move backwards then curve in order to move to a spot left from its original position. Though video 2 continues this process, video 1 actually demonstrates how the robot only moves backwards. Inside the same folder, Reinforcement Learning 3 and 4, the robot always remained in one location and rotated during certain episodes. The episodes either received a score of 1 or 13, but did receive a 41 when rotating.

FUTURE RECOMMENDATIONS

For the first scenario, the robot moves backwards initially because it senses less clusters and smaller detection size. Moving backwards limits the field of vision for the ball and the goal. This also limits the gap perceived between the two objects, creating the illusion that the objects are next to each other from a far perspective. The issue could be resolved using a RangeFinder or distance sensor to detect the distance between the robot and the soccer ball. The “done” function can be altered so the experiment could also reset, whenever the robot is too far from the ball, limiting the backwards motion. Additionally, the robot will be restricted to moving a certain area around the ball, this may encourage the robot to rotate around the ball until it aligns with all three objects aligned with each other.
For the second scenario, the robot primarily rotates or stops at one location. In the “FAIRIS_Controller.py ", though the “reward functions” distributes more points whenever the LIDAR only detects one cluster, the robot can still receive a single point if the current cluster count was less than before. If the program doesn’t end the episode, the environment would still reset. If the conditions are met the robot can be continuously given one point. Counteractive actions are getting rewards and are encouraged to repeat during the next step. One bug could be inside the decision making section when the program is already going through and picking the actions with best reward, instead of exploring different actions and discovering the rewards. Because small rewards can develop through multiple episodes, it’s best to have a counter variable, which increments every time the function has run. This variable can be given a condition which prevents the robot from continuing based on its number. So it only runs the small reward system for a finite number of times.

The overall project was to teach the robot how to accurately kick a ball towards a goal; however, the process has only covered its attempt to align itself with the ball and the goal. Aligning itself avoided any possibility of curved shots. If the project were to continue, the next steps were to develop a kicking and tracking RL system to detect if the ball did reach towards the goal. This RL system would be utilized after using the RL for alignment. The design primarily uses the RangeFinder as its Observation Value. The Rangefinder returns all distances, as an array, under its Field of Vision. In the circumstances of the ball and goal, the return values would be an array which includes the distance of the ball (smaller because it's closer) in the middle of the distances of the goal (larger because it is farther). This assumes the ball is in front of the goal. If the ball were to move closer to the goal, the ball ranges would be closer or equal to the goal ranges. After the environment runs the kicking action, the RL would observe the variance of the rangefinder values. The purpose is to reward the function whenever the variance is low and most of the rangefinder values are equivalent. This distance of the ball and the goal are equal, which would mean the ball had reached the goal and the RL was successful. Most of the RL function would be the same, but observations and reward systems place more importance on the rangefinder instead of the Lidar.



CITATIONS

Tsampazis, Kostas. “How Deepbots Works.” *How Deepbots Works - Deepbots Documentation*, deepbots.readthedocs.io/en/latest/how_deepbots_works.html. Accessed 30 July 2023.

Tsampazis, Kostas. “Cartpole Beginner Robot-Supervisor Scheme Tutorial.” *GitHub*, github.com/aidudezzz/deepbots-tutorials/blob/master/robotSupervisorSchemeTutorial/README.md. Accessed 30 July 2023.

Webots Reference Manual. “RangeFinder.” *Cyberbotics*, www.cyberbotics.com/doc/reference/rangefinder?tab-language=python. Accessed 30 July 2023.

Webots Reference Manual. “Lidar.” *Cyberbotics*, https://cyberbotics.com/doc/reference/lidar?tab-language=python#wb_lidar_get_range_image. Accessed 30 July 2023.

Rodney, Justin. “Analyzing Decision-Making in Robot Soccer for Attacking Behaviors.” *Digital Commons @ University of South Florida*, digitalcommons.usf.edu/etd/9448/. Accessed 30 July 2023.

Schwab, Devin, et al. “Learning Skills for Small Size League RoboCup.” *SpringerLink*, 1 Jan. 1970, link.springer.com/chapter/10.1007/978-3-030-27544-0_7.

Ji, Yandong, et al. “Hierarchical Reinforcement Learning for Precise Soccer Shooting Skills Using a Quadrupedal Robot.” *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, https://doi.org/10.1109/iros47612.2022.9981984.

00000000000000000100000011101111

00000000000000000010000011101111