Robotic Air Hockey

Diogo Filipe Dores
Master in Informatics and Computing Engineering
Faculty of Engineering of the University of Porto
Porto, Portugal
diogo.dores@ieee.org

José Pedro Borges
Master in Informatics and Computing Engineering
Faculty of Engineering of the University of Porto
Porto, Portugal
up201503603@fe.up.pt

***Abstract*—The development of recreational robots that mimic a human part in a real environment has been increasing throughout the years. The project's goal is to create a robot that could play a game of air hockey against a human player. The development of this project went through 42 different iterations and resulted in an agent that could, although not perfectly, fulfill its initial design.**

***Index Terms*—Air Hockey, Robotics, Reinforcement Learning, Unity, Machine Learning**

I. Introduction

Nowadays, there is a plethora of different software that allows anyone to create and develop artificial intelligence to control a robot. The easy access to these technologies has resulted in an increase in various types of robots. One area that is constantly being presented with new creations is entertainment. The project's theme resides within this area.

The aim of this project was to develop a robot that had the ability to play a game of air hockey against a human player. Air hockey is a game where two players compete against each other on a low-friction table. Air hockey requires an air-hockey table, two player-held strikers, and a puck. The developed robot is an intelligent agent that can observe its environment and decide on the best course of action using those observations. It was trained using Reinforcement Learning, an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize some notion of cumulative reward.

For the development of the robot, Unity and, more specifically, Unity's Machine Learning Toolkit were utilized.

This article will thoroughly describe every step and method used during the development of this agent and illustrate the final results obtained.

II. System Description

A Unity environment in 3D was created, and its main components are the robot agent, a player, a puck, two goals, and an academy. The academy is an empty game object with a script that configures the training of the robot.

The camera is on a top-down perspective into the air hockey match, in order to visualize the entire field while playing.

A. Robot Characteristics

The robot is a cylinder with a scaled-down height. It has a mass of 0.4 kg and has a restriction that prohibits it from rotating around the Y-axis.

The robot is also equipped with 3D ray perception rays, which are an alternative system for the agent to provide observations based on the physical environment.

During observations, several rays are cast into the physics world, and the objects that are hit determine the observation vector that is produced. There are a total of 9 rays with an angle of 15 degrees between each one, starting at 30º and ending at 150º. 0º corresponds to the robot's right side, 90º to the robot's front and 180º to the robot's left. The rays have a length of 12 meters and the objects it can detect are the walls and the puck. In pursuance of making decisions, the robot must observe its environment in order to infer the state of the world. **Vector Observation** is a feature vector consisting of an array of floating-point numbers. The target position and robot position vectors, as well as the velocity in x and velocity in z of the robot, were added to the vector observations.

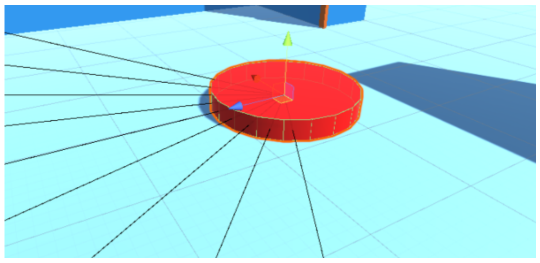

Fig. 1. The robot

B. Robot Movement

After observing the environment surrounding the itself, the robot makes a decision on where it should move and it acts based on that decision. The agent environment is a continuous action space. As such, whenever an agent decides to take any action, an array of Control Signals is created and sent to the Agent Action function. A new control signal, which is a Vector3, is created with an x, y and z properties. Since the Y-axis shouldn't be used on a game of air hockey, as the robot should not move upwards or downwards, only the x and z properties of the control signal are used in the action function. These coordinates are determined based on the vector action that the function received. Whenever the robot decides to move, he moves based on the control signal and a speed parameter.

C. Puck Characteristics

The puck is also a cylinder with the Y property scaled-down and has a mass of 0.1kg. It has a physics material associated with it which makes it have low friction and a high bounciness, in order to resemble the disks in a real air hockey table. A maximum speed of **7 m/s** has been added in order to prevent the puck from reaching a uncontrollable velocities. The object is unable to rotate and move in the Y-axis, in order to prevent it from flying inside the hockey table.

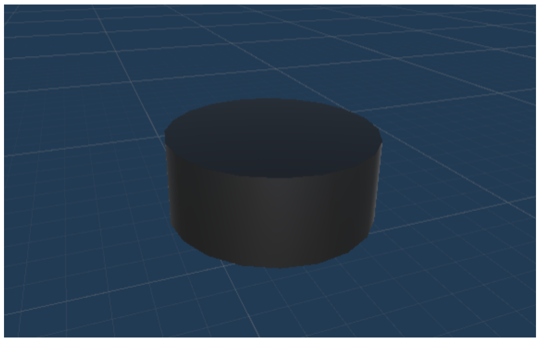

Fig. 2. The Puck

D. Table Characteristics

The air hockey table is composed of six walls, two goals, and a floor. The walls have low friction and high bounciness and the floor has no bounciness and low friction. The playing field has a length of 18 meters and a width of 8 meters. The goals each have a width of 4 meters.

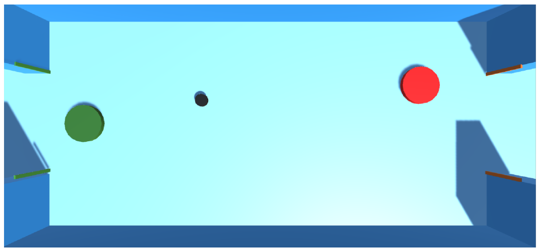

Fig. 3. The Playing Field

III. Algorithm

As mentioned earlier, reinforcement learning is based on a system of rewards. A simple reward system in an air hockey game is that the robot should be rewarded positively when positioned correctly and rewarded negatively when it is not. Based on this approach, the group came up with several reward systems and different tweaks for a total of 42 iterations of the algorithm, which implies that 42 artificial brains for the robot have been created. Detailed below are some of the most relevant iterations.

A. First Iterations

With the puck standing still, the robot was tasked to find it's position and hitting it. The training was considered complete after 50.000 steps had been completed. Rewards:

- 0.1 if the robot makes a move towards the puck;
- -0.1 if it moves away from the puck;
- -1.0 if the robot falls down from the platform;
- 1.0 if it hits the puck.

In this iteration, the robot was able to fall down from the playing field so as to learn to not go inside the goal. By the end of the training, the robot was able to move towards the puck most of the time but was unable to defend the puck when it was moving towards the goal, meaning that the robot kept getting scored on. This issue led the group to try a different training method, which was that instead of searching for a puck that was standing still, the robot would instead learn how to defend a shot from going inside its goal. In order to perform this, the group set up a training camp that had the puck moving from the opposing half court towards the robot's goal. The robot's main task was to prevent the shot from going in. The previous reward system was not reliable anymore, since the distance from the robot to the puck would always be reduced, whether the robot moved or not. The training was still considered complete after 50.000 steps. This way, the group used the following reward system:

- 0.1 if the z coordinate of the puck was between the z coordinate of the top of the robot and the z coordinate of the bottom of the robot;
- -0.1 if the condition stated above was not met;
- 1.0 if the robot successfully kept the puck from going inside the goal;
- -1.0 if the robot got scored on;

The results were more promising with this iteration. The robot was able to save multiple shots, but only the ones that were hit directly in the direction of the goal. This meant that the robot struggled to defend any shot that bounced off a wall before moving towards the goal. With these new results, the group eventually created the final learning algorithm.

B. Hyperparameter Tuning

Prior to writing the final learning algorithm, the group performed some hyperparameter optimization in order to have a faster and more efficient training process. There are many parameters in ml-agents training, but only a small portion of those had the need to be modified.

These parameters also depend on one another. The buffer size should be a multiple of the batch size, and a higher number of epochs requires a larger buffer size. With this information, the group decided to make some changes to the hyperparameters in the trainer configuration file.

- The buffer size was increased, in order to have a more stable training environment;
- The batch size was increased, in order to portray that the learning environment was a continuous action space;

TABLE I
HYPERPARAMETER DESCRIPTION

Hyperparameter	Description
buffer_size	*Number of agent experiences, i.e, observations, actions and rewards obtained that should be collected before updating the model. A larger buffer_size leads to more stable training updates*
batch_size	*Number of experiences used for one iteration of a gradient descent update. A continuous action space should have a higher batch size.*
hidden_units	*Number of units that are in each fully connected layer of the neural network. This number should be smaller for simple problems.*
num_epoch	*Number of passes through the experience buffer during gradient descent. A smaller number will result in more stable updates, but in slower learning*
max_steps	*Number of steps of the simulation, multiplied by frame-skip, that are run during the training process.*

- The hidden units were decreased, to lower the complexity of the learning model;
- The number of epochs was increased in order to develop faster learning;
- Increasing the maximum number of steps from 50.000 to 1.500.000 for a more efficient learning model.

This tuning is what lead to the final model training environment.

C. *Final Algorithm*

For the final iteration of the algorithm, the puck was being shot from the opposing field in a random direction, either bouncing twice on the side walls or moving directly towards the goal. The reward system is as follows:

- The linear equation between the puck and the goal is calculated. The robot receives a reward of 0.1 if he is positioned in that line, in front of the goal;
- -0.1 if the condition stated above is not met;
- 1.0 if the robot successfully prevents the puck from going inside the goal;
- -1.0 if the robot gets scored on.

The robot is not penalized for shots that hit the goal's wall. To deal with this, a timer was implemented so that if a shot does not go in the direction of the goal within 5 seconds, then the shot is reset. The results of this algorithm will be discussed in section V.

IV. EXPERIMENTS

In order to evaluate the performance of the learning model, an online tool named TensorBoard was utilized.

A. *TensorBoard*

TensorFlow is a visualization tool that allows the visualization of certain agent attributes throughout training. This tool uses an open source library named TensorBoard for performing computations using data flow graphs, which are a representation of deep learning models. Using this tool, we can analyze the evolution of the robot learning model.

V. RESULTS

A. *Training Phase*

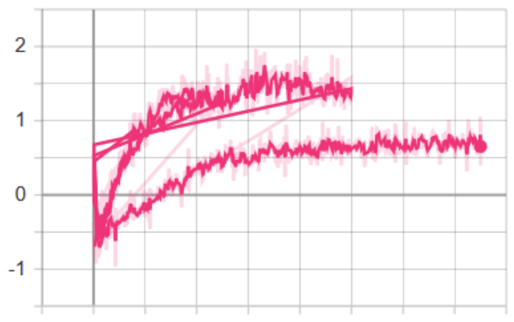

Fig. 4. Cumulative Reward

This graph displays the mean cumulative episode reward of the agent, which should increase in a successful training session. By analyzing this graph, we can see that the rewards for the robot started negatively, displaying that the robot was allowing many goals to be scored. Over time, the robot was allowing fewer goals to be scored, as the average of the rewards increases.

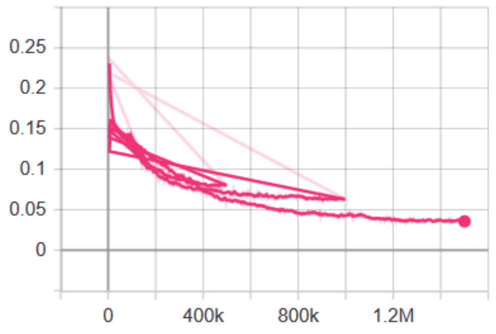

Fig. 5. Curiosity Inverse Loss

This graph represents the mean magnitude of the forward model loss function which corresponds to how well the model is able to predict the action taken between two observations. The lower the value, the more efficient the model is. Analyzing this graph, it can be observed that the model is improving over time and is able to predict better the action to perform.

Overall, the performance of the robot is shown to have improved over time while training.

B. *Playability*

After the training phase was completed, the robot was tested in several scenarios. The robot is extremely successful in defending shots that are sent in a straight line in the direction of the goal, since it is able to position itself in front of the puck, regardless of its initial position.

These results are promising, as they show that the robot is extremely efficient in saving simple shots. For shots that

TABLE II
STATISTICS FOR SHOTS SENT IN DIRECTION OF GOAL

Shots Taken	**Shots Saved**
50	*50*

bounce of the walls, the robot is able to save most of them, although it has difficulty in saving ones that bounce close to the goal.

TABLE III
STATISTICS FOR SHOTS THAT BOUNCE OF THE WALLS

Shots Taken	**Shots Saved**
50	*30*

The results showcased in this table are not as promising as the previous ones, as they prove that the robot is not very efficient when saving shots that bounce off the walls. However, while analyzing the shots that were taken, the group came to the conclusion that most of the shots that went in the goal were shots that bounced from the top wall, rather than from the bottom one, making it a point where the robot can still be improved.

When the robot is put against a human player, it doesn't stand much of a chance. Since the robot's main focus is defending, it is unable to make good shots against a player, therefore while it is most efficient as a goalkeeper, it struggles to score.

VI. CONCLUSIONS

In this project, an overall successfully working robot was created. There are obviously some improvements to be made, such as an enhancement to the robot's ability to guard its goal from shots coming from a greater Y coordinate than the robot's. The robot could also still be improved in terms of its attacking capabilities.

In a real experiment against a human player, the robot selected mostly the optimal behavior and executed reasonable adequate motions in each situation.

REFERENCES

[1] Unity-Technologies/ml-agents: Unity Machine Learning Agents Toolkit, [online], available at https://github.com/Unity-Technologies/ml-agents, consulted January 2020.

[2] Unity - Unity Machine Learning tools and resources, [online], available at https://unity3d.com/machine-learning, consulted January 2020.

[3] Reinforcement learning - GeeksforGeeks, [online], available at https://www.geeksforgeeks.org/what-is-reinforcement-learning/, consulted January 2020.

[4] What is an Air Hockey Table & How Does it Work?, [online], available at https://www.libertygames.co.uk/store/air_hockey_tables/buying-advice/what-is-air-hockey/, consulted January 2020.

[5] Reinforcement learning - GeeksforGeeks, [online], available at https://www.geeksforgeeks.org/what-is-reinforcement-learning/, consulted January 2020.

[6] TensorBoard - TensorFlow, [online], available at https://www.tensorflow.org/tensorboard, consulted January 2020.