“RoboSparky" A Penalty Kick Robot

Kevin Vora, Maitry Trivedi, Anil Venkata Kota and Piyush Rajesh Medikeri
Robotics and Autonomous Systems(AI), Arizona State University

I. Introduction

The aim of this project is to make a robot learn how to score a goal during a penalty shootout and also make another robot in the same scenario learn how to stop the goal during a penalty shootout, so our project is aimed at multi-agent learning. We have used reinforcement learning to make both the robot learn by spending more and more time collecting rewards and analyzing the best move for both the robots. We are training both the robots simultaneously where one agent learns and changes its action by adapting to the change of the other agent.

We have implemented a simulation environment containing a grass field, striker robot, goal, a soccer ball and a goalkeeper, where the agents can be observed as it evolves into a professional player via multiple episodes of practice sessions under different scenarios. We were successfully able to achieve our ambitious goal to incorporate a goalkeeper and observe how the agents evolve. If this approach could be used in real-life on real robots, to train the soccer player it would give a much better result than a player practicing penalty kicks with a real human because our robot would learn and adapt to that person by observing even a small characteristic that would give the robot an intuition of where the human is going to hit the ball or where the human going to go to stop the ball.

II. Methodology

A. Environment

We have used a virtual robotics experimentation platform (V-REP) for our simulations and results for this project. Our environment is a football field designed in V-REP with a football, a goal post and our striker robot “RoboSparky”. We have set specific dynamics in the V-REP environment for the football and the striker robot which allows it to hit the ball with a defined force towards the goal. We have also put some texture on the objects to make the environment look more realistic (Please refer Fig. 1).

B. Striker

We have implemented Q-learning for the striker agent without the goalkeeper in the environment, which is a model-free reinforcement learning algorithm, to learn a policy that tells an agent what action to take under varied circumstances. Here the agent is our robot and the action is the angle with which the agent should hit the ball. Different angles (A,B,C,D,E,F and G in Fig. 2) are achieved by changing the velocities in x and y directions with respect to the robot (Vx, -Vx, Vy and -Vy in Fig. 2). For the robot to hit the ball, we just assign velocity values in the x and y directions which makes it kick the ball

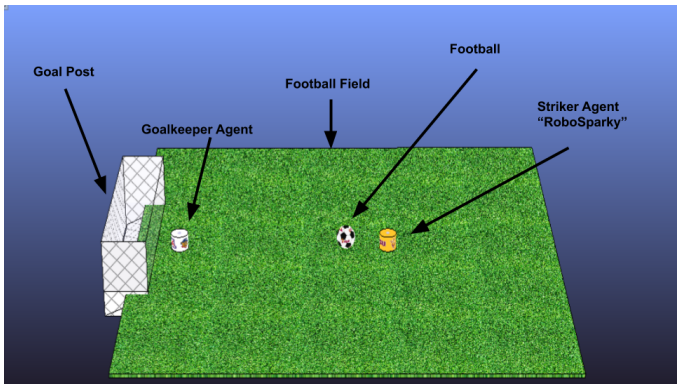

Fig. 1: Our V-REP simulation environment

in a specific direction. The robot eventually stops due to the ground friction. In terms of reinforcement learning, the actions would be the velocities in x and y directions (indirectly the angle at which we are hitting the ball), and the state would be the position of the ball with respect to the robot. The reward function for our algorithm is based on the end location of the ball after each episode. If the ball is inside the goal, then we give +1 reward (Positive reward in Fig. 2) to that action and if the position of the ball is not inside the goal, then we give -1 reward (Negative reward in Fig. 2) to that action.

After the successful implementation of the explained model, we increased the problem statement by adding the random position of ball. We selected 8 discrete values for the ball position from where the agent can hit the goal with particular velocity . The 9 values velocity in y direction Vy are defined from which some of the velocity . Therefore, the agent tries to learn total 72 combinations with the variation in ball and agent position. At the end of 20 exploration and 180 exploitation steps, the agent learns to kick a ball in the goal position.

C. Goalkeeper

The next part of our implementation is the goalkeeper where we are making the robot learn to stop the ball from entering the goal by obstructing the ball path. The method used for training the goalkeeper is quite similar to the training of the striker but with a little variation. Similar to the striker, here also we have used Q-learning to train the goalkeeper. Since it is just one state scenario where the next state would lead to the goal state, we have not taken state into account during Q-learning. The action is the velocity given to the goalkeeper to move in a direction, that is either left or right. Since the goalkeeper has to move only to the left or right (Vx and -Vx in Fig. 2), it needs velocity for only one axis (‘x-axis’) which would make the

robot move in that axis ('x-axis') and not the other ('y-axis'). Now as we know what the actions would be, we have to decide the action space. Since we have discretized the velocity values for the striker, we have to discretize it for the goalkeeper also (Ag and Bg in Fig. 2). Therefore, every time the striker hits the ball with some velocity the goalkeeper should move with a velocity corresponding to strikers shooting velocity that would stop the ball. This was done by trial and error method on the V-Rep simulator by manually checking which velocities would stop the ball, hit with some specific velocities by the striker. These specific striker velocities were stored in a database and the corresponding goalkeeper velocities were stored in another.

Now as we have the action space ready, we can start the learning process. In this whole goalkeeper implementation process we have always chosen the striker's velocity randomly from the database that we had stored for striker's velocities in. We start by dividing the training episodes into exploration episodes and exploitation episodes. In the exploration episodes, we randomly choose the velocities for the goalkeeper to move and learn and in the exploitation episodes, it chooses the action which has been awarded the maximum reward. We have kept a simple reward function for the goalkeeper. If it stops the ball, the action gets a reward of 1, if it does not stop and let the striker score a goal, then it gives that action as -1 and if the ball is outside the goalpost, then the action it took does not get any reward.

D. Striker and Goalkeeper Combination

To make this project more interesting we combined both the striker and the goalkeeper models with their learning algorithms in the same environment. But we needed a way to make these two models talk to each other. To make this interaction we changed the way in which the striker was hitting the ball in the goalkeeper model. In the goalkeeper model, we are always choosing the velocity of the striker randomly but here we gave the striker velocity the velocity it learned to score a goal with, that is the velocity with the highest reward. With this implementation, we were able to convert this project to a multi-agent learning model project, where both the agents, the striker, and the goalkeeper are learning simultaneously.

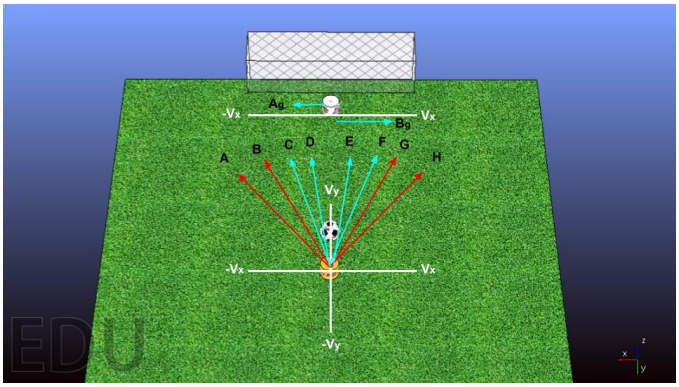

Fig. 2: Action space of the robots

Algorithm 1 Learning algorithm

procedure TRAINING($shooter, goalkeeper$)
  while $n \leq iterations$ **do**
    if $n \leq 20$ **then**
      (vx, gx) $\leftarrow chooseRandomActions$
    else
      vx $\leftarrow modelPredictionShooter$(steps)
      gx $\leftarrow modelPredictionGoalkeeper$(ballPos)
      Update Q-Table of shooter
      Update Q-Table of goalkeeper

E. Algorithms

We used q-learning to train the striker and goalkeeper in a discrete action space setting. When q-learning is performed we create a q-table or matrix that follows the shape of [state,action] and we initialize our values to zero. Then we update and store our q-values during the learning process. Both the agents have their own q-tables which get updated when they start learning.

The striker and the goalkeeper are both made to train simultaneously. The first 20 steps are taken as exploration steps and both of the agents choose random actions. They then start updating their Q-tables and learning during their exploitation steps. The striker kicks the ball first with a velocity (vx, vy) by choosing the action with maximum Q-value from the Q-table. Then the goalkeeper takes the position of the ball after it travels for a fixed interval of time as input and it moves with a velocity (gx, gy) which is the action with the maximum Q-value in its Q-table. Then both of the Q-tables are updated and this process continues till a fixed number of iterations or episodes. Both the agents were trained with a constant velocity in the y-direction i.e., they learn the velocities only in the x-direction vx and gx. We trained the agents for 75 episodes in our implementation.

F. Experimentation with continuous action space

We have also tried implementing Harmony search Algorithm which is a a population based meta-heuristic algorithm for optimization tasks. Developing reinforcement Learning Algorithm for continuous action space is very difficult task as it requires a lot of computation. Therefore, population based algorithm work better than Reinforcement Learning for these problems. In this scenario, we have defined a range for the velocity in y Direction of the agent which is [-27,27].The time taken to reach a goal is also accumulated in the cost function of the optimization task. The fitness of the solution for which agent hit a goal is 1+1/t, where t is the time the agent takes to reach a goal. The fitness of the solution for which agent miss a goal is 1+1/t. Therefore, at the end of 50 iterations, the agent learns to hit a goal in a minimal time.

.

III. Summary

We present a multi-agent learning environment, consisting of a penalty kick agent designed to shoot a goal and a

goalkeeper agent to block the goal. Q learning is used in this agent learning process. We have used the V-REP environment to showcase the results of our algorithm. Through the use of this algorithm, our agent is most likely to exhibit anticipatory behavior with the result of scoring a goal.We also explored harmony search algorithm to eliminate the problem of discrete action space. We also demonstrate a goalkeeper agent that blocks the ball hit by the player agent. In a multi agent environment continuous learning takes place where goalkeeper looks for moves to block and player strikes the ball in the direction which scores a goal. In future, agents can be replaced by humanoid robots and trained. Also these agents can learn new tactics if given better reward functions and action space.