

Foosball Table Goalkeeper Automation Using Reinforcement Learning

Tobias Rohrer, Ludwig Samuel, Adriatik Gashi, Gunter Grieser and
Elke Hergenröther

Darmstadt University of Applied Sciences, Schöfferstraße 3, 64295 Darmstadt, Germany

Abstract

Reinforcement learning (RL) is used to learn strategies to autonomously control the goalkeeper rod on a physical Foosball table in an efficient manner. The training is performed in simulation to be transferred to the physical Foosball table for execution. In order to make the training in simulation more robust, we applied domain randomization. By performing experiments we could prove that the agent learned reasonable movement patterns as well as efficient strategies which are applied by professional Foosball players to maximize the chance of defending a ball. The agent learned these strategies solely by the use of a sparse reward in a trial and error manner and without any hand-crafted or domain-specific knowledge.

Keywords

Reinforcement learning, Robotics, Automation, Foosball table

1. Introduction

Automating the game of football is a complex task as it includes autonomous player strategies like shooting, running, or passing as well as learning overall team strategies. Not only because of the public attention to the sport but also because of its challenging dynamics, football is seen as a challenge in AI and robotics for a long time. The well-known RoboCup challenge intends to support research in the field of robotics and AI by providing football as a standard problem since 1996 [1]. Recently researchers at Google released an open source 3D football simulation environment [2] to further push research in this field.


In the work at hand, we are aiming to automate the game of Foosball which is also known as table soccer. Foosball represents an abstraction of the real football game, eliminating some degree of complexity as the complexity of control of players is reduced. Yet the task of automating a Foosball table still includes learning autonomous as well as team strategies within a real time dynamic environment. As our goal is to automate Foosball on a physical system another level of complexity is introduced as real time sensor data and control are necessary. In this work, we are focusing on the automation of the goalkeeper rod. The concepts and methods presented can be applied to the other rods on the Foosball table in the future to archive our overall goal to automate a whole Foosball table.

LWDA'21: Lernen, Wissen, Daten, Analysen September 01–03, 2021, Munich, Germany

✉ tobias.rohrer@stud.h-da.de (T. Rohrer); ludwig.samuel@stud.h-da.de (L. Samuel); adriatik.gashi@stud.h-da.de (A. Gashi); gunter.grieser@h-da.de (G. Grieser); elke.hergenroether@h-da.de (E. Hergenröther)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

As the creation of a hand-crafted algorithm could be potentially time consuming and error prone, we are focusing on reinforcement learning (RL) to learn efficient strategies to automate the Foosball table. As learning strategies in RL is done in a trial and error manner, we created a digital twin of the Foosball table to provide a simulation to be used for training. After performing the training of the agent in simulation it was transferred to the physical Foosball table for execution. Although we trained the agent entirely in simulation, we could see during execution on the physical Foosball table that it picked up human movement patterns which are also applied by professional Foosball players.

2. Related work

In [3] they used RL to train the attacker rod on a Foosball table to shoot a ball. They also trained their agent in simulation and later transferred it to the physical Foosball table. It must be noted that they used the exact same physical Foosball table including the same control systems as we used in our work. However, in contrast to our work they used a continuous action space as well as Proximal Policy Optimization (PPO) [4] to train their RL agent. By using a discrete action space in our work we simplified the control mechanics of the Foosball table and thus hypothesize better applicability of our approach to be used to automate the whole Foosball table in future. In addition, our method shows another approach of automating a Foosball table rod using RL. In another, earlier autonomous Foosball table project a predefined decision tree was used to map the current state of the game to actions [5]. In [6] they are learning an autonomous Foosball table to perform the actions of locking and kicking the ball by imitation. Similar to the game of Foosball, the automation of an Air hockey table also includes autonomous real time decision making in a dynamic environment. In [7] they used deep Q-learning to train an agent to control a robot to strike a puck on an Air hockey table. As another example, in [8] they applied reinforcement learning to learn policies able to return a table tennis ball by controlling robot joints in a simulated environment.

3. Theoretical background

In reinforcement learning an *agent* learns to solve a task by learning from trial and error when interacting with an *environment*. In the following, an overview of the RL algorithms and concepts used in this work are given. For more a more detailed description please refer to [9].

3.1. Markov decision processes

The Markov decision process (MDP) [10] provides the mathematical formalism which is used in reinforcement learning. An MDP is defined by a state space S , an action space A , transition probabilities $p_i(s_{t+1}|a, s_t)$ which define the probability to transition from state s_t to state s_{t+1} if the agent chooses action a , as well as a reward function $r(s_t, a, s_{t+1})$ which defines the reward the agent gets for transition from state s_t to s_{t+1} using action a . The *return* $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$ defines the sum of future rewards from the current time step t discounted by factor γ which is used to define how much impact future rewards have compared to immediate rewards. The agents goal

is to maximize the *return* by choosing optimal actions depended on the current state s_t . This is done by a policy π which defines the mapping from a state to an action. [10, 9, 11]

3.2. Q-learning

In Q-learning we try to find a Q-function $Q(s_t, a) \rightarrow R_t$ which estimates the *return* the agent can expect for being in state s_t and taking action a . Those expected returns based on the state and chosen action are also named Q-values (Quality values). Once an optimal Q-function is defined, a greedy policy π which always chooses the action with the highest Q-value $\pi_{opt}(s) = \arg \max_a Q(s_t, a)$ is the optimal policy which maximizes the expected future return.

As in many RL problems the MDP is not fully known beforehand, we must introduce a procedure that helps us explore the MDP during training. Otherwise, the usage of a purely greedy policy would probably end in a local minimum. Therefore an ϵ -greedy-policy can be used during training. By using an ϵ -greedy-policy we are choosing to take random acting at every time step t with probability ϵ and action according to π_{opt} with probability $1 - \epsilon$ [12, 10].

3.3. Deep Q-learning

In Q-learning, the Bellman equation [10] is used to calculate the Q-values for all state action pairs in an iterative process. However, this method is computationally not suitable for real world problems with large state and action spaces. In Deep Q-learning, a neural network is used as Q-function which estimates Q-values [12].

4. Task definition

In the presented work we are trying to teach an RL agent to control the goalkeeper rod on a Foosball table. By applying reinforcement learning we aim to learn a strategy that maximizes the chance of the goalkeeper to save a ball which is shot at the goal. In the following, we describe the hardware setup of the Foosball table as a physical target environment. Furthermore, we describe how we framed the task as an MDP.

4.1. System overview

As shown in figure 1, one side of the Foosball table is controlled by a set of motors that mimics an autonomous controlled team. Programmable logic controllers (PLC) are used as interface between these control motors and the reinforcement learning agent. The communication between the PLC and the RL agent is done via the Open Platform Communications Unified Architecture (OPC-UA) which is used to read the current angular and lateral positions of the rods. To get visual information about the current state of the Foosball game, a *Logitech Brio* webcam with 30 frames per second (fps) is mounted on top of the table. The OPC-UA interface can also be used to execute the actions chosen by the reinforcement learning agent to control the rods by setting target angular and lateral positions. Even though we are only focusing on the control of the goalkeeper rod in this work, the architecture is ready to handle an agent which controls the whole team. As we are using the same Foosball table and control mechanics

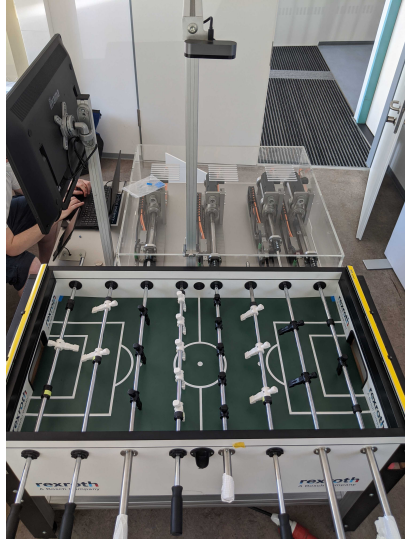


Figure 1: The Foosball table with controlled rods for one team.

as in [3] we refer to their work for a more detailed system description from an engineering perspective.

In order to process the images captured by the camera, we are using a simple yet effective algorithm to extract the coordinates of the ball. We are using a ball that is colored red and are simply searching the image for an approximate ball shaped item with a similar color. In the future, it is planned to also extract information about the position of the human controlled rods from the images captured by the camera.

4.2. Foosball as MDP

The following subsection describes how we framed our task to learn an agent which controls the goalkeeper rod on a Foosball table as an MDP.

4.2.1. States

For every frame which is captured by the camera (30 fps), the information about the ball position gets extracted and combined with the current lateral position of the goalkeeper rod to form a *frame descriptor*. A *frame descriptor* is 3-dimensional and contains information about the current x and y coordinates of the ball as well as the lateral position of the goalkeeper rod. We are combining information of 4 successive 3-dimensional *frame descriptors* into one 12-dimensional state s_t . This is done to comply with the Markov property as it is possible to calculate the direction and velocity of the ball and goalkeeper rod from a single state s_t without relying on the past states s_1, \dots, s_{t-1} . It is also important to note that we designed our states to be disjoint which showed to be more effective. Meaning one *frame descriptor* can only be included in one state. This leads to the fact that a new state s_t is generated every 4 frames. Or in other words: 4

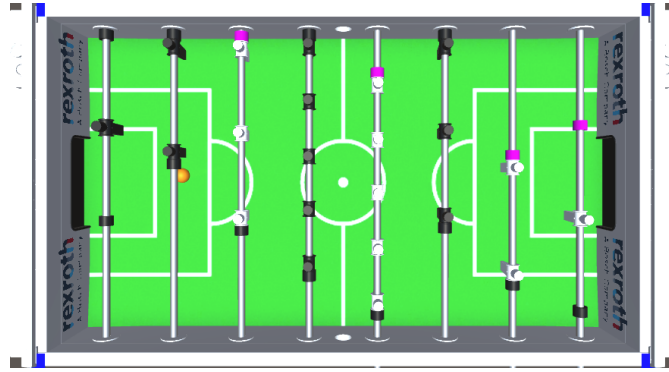


Figure 2: Simulation of the Foosball table

frames make one time step t . As we are working with 30 fps, we reach a new state s_t every 133 ms.

4.2.2. Actions

A discrete action space with 3 possible actions is used: $a \in \{left, right, noop\}$. The agent can either choose to move the goalkeeper rod to the left, to the right or not move it. When moving the goalkeeper, a new absolute target position is calculated based on its current position and a fixed delta. We set the delta to 5% of the complete range of motion of the goalkeeper rod. It is noteworthy that the movement to a target position will be overwritten and interrupted instantly by successive actions if not completed. This helps us to prevent an evolving lag in the execution of the chosen actions.

4.2.3. Rewards

We implemented a sparse reward structure where the agent gets an immediate reward of +100 if the goalkeeper holds a ball and -100 if a goal was scored. An episode ends only if the goalkeeper was not able to defend the shot and thus a goal was scored.

5. Training

A simulation was used for training the RL agent as the physical system only provides data with the help of human interaction. After training in simulation the agent was transferred to the physical Foosball table for execution.

5.1. Simulation

The Foosball table was simulated by creating a digital twin of the Foosball table using Unity¹. Sizes and distances as well as other physical properties like the mass of the real table and its

¹<https://unity.com/>

Table 1
Hyperparameters of Deep Q-learning

| Parameter | Value |
|--------------------------|------------------|
| Number of layers | 4 |
| Optimizer | Adam |
| Learning rate | 0.01 |
| Replay buffer size | 10^6 |
| Mini batch size | 32 |
| Discount factor γ | 0.95 |
| ϵ | $\in [0.8, 0.2]$ |

components were captured and applied to the simulation. Analog to the OPC-UA interface of the real Foosball table, the simulation provides a communication interface using network sockets. This interface is used to get the information needed to calculate the current state s_t of the game as well as to execute actions. Like on the physical Foosball table described in section 4.1 we set the frame rate of the simulation to 30fps, where the information of 4 successive frames form one state s_t . The information about the position of the ball is directly provided by the simulation and is not calculated based on the rendered image of the simulation. This introduces a gap between simulation and reality which has to be removed in future work.

In order to train the goalkeeper to save the ball, we simulated artificial shots targeted at the goal. In order to do so, the ball is placed randomly somewhere on the axis of the opponent’s striker rod. The shot is then simulated by applying a random amount of force F in Newton to the ball, where $F \sim U[0.05, 0.06]$ and the mass of the ball is simulated with 0.022 kg. In 80% of the simulated shots, we are determining the position of the goalkeeper to select a target shot position that lies randomly in the larger of the two gaps between the goalkeeper and the goalpost. This behavior should imitate a human player who actively tries to aim for the larger of the two areas between the goalkeeper and the goalpost. In the following, we are referring to this shot pattern as *smart shots*. In 20% of the shots, we are targeting a complete random position in the goal, which is independent of the position of the goalkeeper.

Like in the physical system described in 4.1 the simulation already supports all rods to be controlled independently. This means the simulation as well as the physical system can be used for training a RL agent to automate not only the goalkeeper rod but the whole game of Foosball, which is our ultimate future goal.

To prevent the agent to overfit certain dynamics or measures of the simulation, we applied domain randomization. In each episode, during training we are simply multiplying the physical properties of ball mass, ball drag, ball scale as well as the goalkeeper figure scale by an individual random factor drawn from a normal distribution with a mean of 1 and a standard deviation of 0.05.

5.2. Applying Deep Q-learning

We used Deep Q-learning to learn an efficient strategy to control the goalkeeper rod of the Foosball table. We used a Double Deep Q-network (Double DQN) [13] in order to estimate the

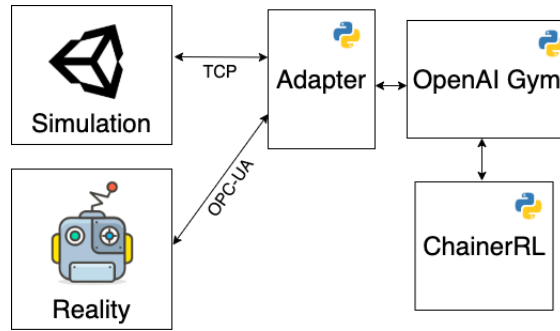


Figure 3: Software architecture

Q-values required to choose the optimal actions. For the purpose of balancing exploitation and exploration, we used a linear decay strategy to gradually reduce our ϵ , which defines the chance of exploration by performing a random action during training. The Hyperparameters of the network as well as other Deep Q-learning specific Hyperparameters can be seen in table 1. We trained the agent for 200.000 steps which approximately equals 20.000 episodes during the training. This took around 7 hours and 20 minutes to complete given the frame rate of 30fps.

6. Implementation details

Figure 3 shows the software architecture used to implement the presented concepts. We are using a custom adapter module to handle the communication with the simulation via network sockets as well as with the Foosball table using OPC-UA. One can choose before the start of training or execution of the agent whether it should run in simulation or Foosball table mode. Furthermore, the adapter module handles scaling of the coordinates to unify the states and actions of the physical Foosball table and the simulation. By using OpenAI Gym [14] we were able to implement our MDP defined in 4.2 as a standardized interface, called an *environment*. The usage of OpenAI Gym provides the possibility of using a variety of reinforcement learning libraries that implement state of the art RL algorithms. We are using the ChainerRL [15] deep reinforcement learning library which contains a ready to use implementation of the Deep Q-learning algorithm. The architecture as described in figure 3 is implemented in Python. *

7. Results

Because reproducible quantitative results of the agent's performance on the physical Foosball table are hard to determine we split between quantitative and qualitative results in this section. The evaluation in simulation is described in a quantitative manner whereas the results on the physical Foosball table are reported qualitatively.

Table 2

Quantitative test results in simulation

| | Tested with 100% random shots | Tested with 100% smart shots |
|-------------------------------------|-------------------------------|------------------------------|
| Trained with 80% <i>smart shots</i> | 76.3 % | 74.2% |
| Trained with 100% random shots | 70.7% | 62.9% |

7.1. Quantitative (simulation)

The evaluation of the performance of the goalkeeper was mainly conducted in simulation as it provides identical testing conditions which are necessary to compare the performance of different strategies, algorithms, or Hyperparameters. Table 2 lists the results after simulating 2000 shots. The numbers denote the percentage of shots that the goalkeeper was able to defend. It can be seen that the introduction of the *smart shots* described in section 5.1 has a positive impact on the goalkeeper performance. It is noteworthy that an agent trained using 80% *smart shots* is better in saving random shots than an agent trained using 100% random shots. This means that the imitation of human-like behavior by actively aiming for blank spots in the goal by *smart shots* helped the agent adapting strategies to save the ball.

7.1.1. Impact of reward structure

We have experimented with the impact of different concepts of how rewards are given to the agent. The goalkeeper performance which could be archived by implementing the reward structure as described in section 4.2.3 can be found in table 2. In an experiment, we changed the reward structure in a way that one episode also ends whenever the goalkeeper saves the ball and not only when a goal was scored. The agent was thus only able to either get a reward of +100 for saving the ball or a reward of -100 for not saving the ball. Using this strategy we could see a huge drop in the performance of the goalkeeper which was trained using 80% smart shots as it was only able to save 54% of random shots. This underlines the importance of choosing an appropriate reward structure for the agent to be able to learn efficient strategies.

7.2. Qualitative (physical Foosball table)

We executed the agent which was trained in simulation on the real Foosball table to evaluate its performance. We could observe that the agent picked up some efficient movement patterns in order to hold a ball. First, we tested if the agent is able to react based on the current position of the ball. By placing the ball at random positions on the Foosball table we could see that the agent adjusts the position of the goalkeeper based on the position of the ball in a way that maximizes the chance of holding it (figure 5). Professional Foosball players tend to move the goalkeeper rod quickly back and forth by a small ratio to maximize their chance of defending an opponent's shot. We could see that the agent also learned this strategy as it tends to also move quickly back and forth by a bit centered around the position of the ball. This behavior is outlined in figure 4. It must be noted that the agent was able to learn this strategy solely by the use of a sparse reward in a trial and error manner. However, we can see a lag in the agent's

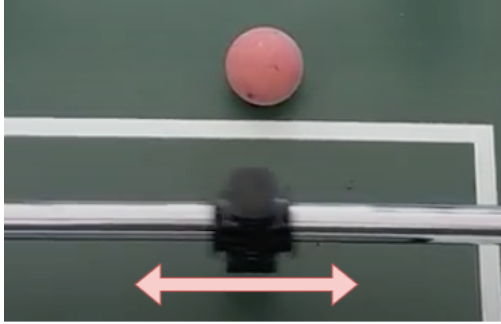


Figure 4: Goalkeeper moves quickly back and forth to maximize its chance to hold the ball.

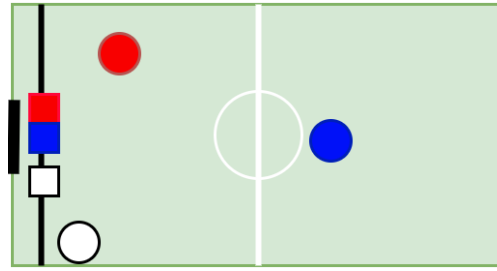


Figure 5: Agent adjusts goalkeeper position (rectangle) relative to the position of the ball (circle) to maximize chance of defending.

decision when quickly changing the position of the ball by passing or shooting with a high amount of force. This lag could not be shrunk significantly by applying the measures of domain randomization as described in section 5.1. We hypothesize that the reason for the lag lies in a different kind of processing the information of the ball position as described in section 5.1 which introduces a gap between simulation and reality which we plan to close in future work.

8. Conclusion and future work

As mentioned earlier, our ultimate future goal is to automate the whole game of Foosball by learning a RL agent which controls all rods of one side of the table. Therefore, in future work we will test and work on the applicability of the presented approach to control multiple rods. The simulation as well as the physical system already provide the basis to archive this future goal. By now, we could train a reinforcement learning agent to control the goalkeeper rod on a physical Foosball table by the application of Deep Q-learning. By performing experiments we could see that it picked up human movement patterns as well as strategies used by professional Foosball players. It must be noted that those movement patterns were learned solely using a sparse reward signal in a trial and error manner. The work at hand is another example of the wide applicability of reinforcement learning in robotics.

References

- [1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, H. Matsubara, Robocup: A challenge problem for ai, *AI Magazine* 18 (1997) 73–85.
- [2] K. Kurach, A. Raichuk, P. Stanczyk, M. Zajac, O. Bachem, L. Espeholt, C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, S. Gelly, Google research football: A novel reinforcement learning environment, *CoRR* abs/1907.11180 (2019). URL: <http://arxiv.org/abs/1907.11180>. arXiv:1907.11180.
- [3] S. De Blasi, S. Klöser, A. Müller, R. Reuben, F. Sturm, T. Zerrer, Kicker: An industrial drive and control foosball system automated with deep reinforcement learning, 2020. doi:10.13140/RG.2.2.19201.28003.

- [4] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, CoRR abs/1707.06347 (2017). URL: <http://arxiv.org/abs/1707.06347>. arXiv:1707.06347.
- [5] T. Weigel, Kiro - a table soccer robot ready for the market, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation, 2005, pp. 4266–4271. doi:10.1109/ROBOT.2005.1570776.
- [6] D. Zhang, B. Nebel, Learning a table soccer robot a new action sequence by observing and imitating, 2007, pp. 61–.
- [7] A. Taitler, N. Shimkin, Learning control for air hockey striking using deep reinforcement learning, CoRR abs/1702.08074 (2017). URL: <http://arxiv.org/abs/1702.08074>. arXiv:1702.08074.
- [8] W. Gao, L. Graesser, K. Choromanski, X. Song, N. Lazic, P. Sanketi, V. Sindhwani, N. Jaitly, Robotic table tennis with model-free reinforcement learning, arXiv preprint arXiv:2003.14398 (2020).
- [9] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, A Bradford Book, Cambridge, MA, USA, 2018.
- [10] R. Bellman, A markovian decision process, Journal of mathematics and mechanics 6 (1957) 679–684.
- [11] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, W. Zaremba, Learning dexterous in-hand manipulation, The International Journal of Robotics Research 39 (2020) 3–20. URL: <https://doi.org/10.1177/0278364919887447>. doi:10.1177/0278364919887447. arXiv:<https://doi.org/10.1177/0278364919887447>.
- [12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533. URL: <https://doi.org/10.1038/nature14236>. doi:10.1038/nature14236.
- [13] H. van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, CoRR abs/1509.06461 (2015). URL: <http://arxiv.org/abs/1509.06461>. arXiv:1509.06461.
- [14] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, 2016. arXiv:arXiv:1606.01540.
- [15] Y. Fujita, P. Nagarajan, T. Kataoka, T. Ishikawa, Chainerrl: A deep reinforcement learning library, Journal of Machine Learning Research 22 (2021) 1–14. URL: <http://jmlr.org/papers/v22/20-376.html>.