RoboCup 2023 SPL BadgerBots Team Description Paper

John Balis[1], Yoon-Chae Na[1], Adam Labiosa[1], Arun Ravi[1], and Josiah Hanna[1]

University of Wisconsin, Madison

1 Team Information

Our team name is BadgerBots and we are affiliated with the Computer Sciences Department at the University of Wisconsin – Madison, which is located in the state of Wisconsin in the United States of America. We are a group of students (PhD, MS, and undergraduate) advised by Prof. Josiah Hanna. Our team's contact email is jphanna@cs.wisc.edu. We do not have a team website at this time.

2 Code Usage

Our team codebase for the 2023 competition is based upon the BHuman 2022 code release . As described below, our primary contribution has been to replace the high level behaviors written by BHuman with new, learned behaviors based on deep reinforcement learning. As discussed below, we believe these changes to be transformative in nature. Thus, as of now, our robots use the perception, state-estimation, low-level motion control and communication modules provided by BHuman. We thank BHuman for making their full software stack available.

In addition to code from BHuman, we also added the following free and open source C++ libraries to our codebase:

1. **GCE-Math C++ library**
2. **StatsLib C++ library**
3. **cpp-json**

Additionally, we used the open source library **stable-baselines3** to train our policies, using proximal policy optimization[3][2].

Our team has not competed in prior RoboCup competitions, so there are no prior code uses to report.

3 Own Contribution

The central objective of our team is to enable complex robot behaviors that are completely learned from experience rather than manually programmed. Towards this aim, we have developed preliminary abilities to use reinforcement learning

(RL) algorithms to train deep neural network control behaviors that are the basis of our robot's high-level behaviors. Real-world robot soccer is a task that is substantially more difficult than many of the tasks today in which RL algorithms are developed and tested. Nevertheless, the ability to learn behaviors will likely prove to be critical to realizing the RoboCup vision that a team of robots will defeat the human world cup champions by the year 2050. By committing to RL for developing our team, we are taking an important step towards this vision.

Our team is pursuing research in several directions to enable the use of modern RL algorithms for RoboCup.

1. **RL in Abstract Simulators.**
   Training in simple, efficient simulators, then directly deploying on physical robots. This capability matters for RoboCup because with a well-developed pipeline for training policies, it potentially takes much less human programming to get a learned behavior than a manually programmed one.
2. **Automatic Simulator Grounding.**
   Simulators inherently fail to capture the complexity of real physical robots. To address this, we are using recordings from from complex simulations and the real robots to improve the realism of state transitions in the abstract simulator. This research matters for Robocup because if an abstract simualtor can be made to more closely match the transition dynamics of the real robots, it should create policies which work better with the physical robots.
3. **Multi-agent Reinforcement Learning.** Training multiple agents simultaneously so agents learn to coordinate with one anther. This direction is crucial for RoboCup as we find that robots trained individually otherwise require manual heuristics to prevent collisions and inefficiency.
4. **Offline RL and Data Augmentation.** Using recordings from the real robots for policy-improvement through offline reinforcement learning, and additionally augmenting these recordings with synthetic modified state transitions(for example, translating the robot and ball around the field). This matters for Robocup, because it's another potential approach to getting behaviors for the physical robots without having to manually program them.

In addition to enabling application of RL in RoboCup, the outcomes of these research directions can be published at leading robotics conferences as well as AI and machine learning conferences and journals.

To date, we have primarily relied upon RL in abstract simulation to train deep neural network polices that are then directly used on the physical NAOs. Below, we detail both how we use these trained policies on the physical robots and how we train them. Our eventual goal is to create a RoboCup SPL team where both high and low level control decisions are made by policies created through reinforcement learning.

3.1 Real time inference on NaoV6

After training a control policy in our abstract simulator (e.g., a single-agent keeper or attacker policy), we export our neural network policies from PyTorch into .h5 files which specify the architecture and weights of the policy's neural network. These files are saved to a subfolder of our fork of BHumanCodeRelease. We use a tool developed by BHuman called BHuman User shell(bush) to deploy our fork to the NaoV6 robots. The policy architecture and parameters are copied over in this process. The BHumanCodeRelease by default uses a system of abstractions where high level control primitives call low level control primitives according to some set of logical rules, heuristics, and location-based potential fields. We use the pre-made skills for low level control, but replace the high-level control logic for when the game state is playing with our own high-level control system, which extracts observations in the format expected by our neural policies using the perception code from BHumanCodeRelease. These observations are then given to the neural policy for inference and the output of the neural policy is used to parameterize low level skills. We aim to present a team where each high level control task: push ball to goal, defend goal, kickoff, etc. is performed only using a neural policy to choose parameters for low level walk and kick skills. In the future, we aim to consider replacing more and more components of the system with policies trained through RL, eventually replacing low level control with neural policies.

In the current version of our code, we limit our neural control to when the game is in the playing state. We have 3 distinct neural policies: a goalkeeper policy, a defender policy, and an attacker policy. We assign the robot numbered 1 the goalie policy, the robots numbered 2 and 3 the defender policy, and the robots numbered 4 and 5 the attacker policy. During kickoffs and penalty kicks, we leave the BHuman high level control code in place. Observations for our policies observation spaces are constructed using data from the BHuman Perception system, and the output of our neural policies is used to parameterize WalkAtRelativeSpeedSkill. We additionally have trained an attacker policy which is capable of kicking the ball into the goal. Currently it relies on the WalkAtRelativeSpeed skill and WalkToBallandKick skill, which it is able to select and parameterize via it's action space, but due to how it was trained in our abstract simulation, only kicks when it is right next to the ball, so it should be compatible with a lower level "Kick in place" skill(We want to rely on a minimal amount of BHuman high level control code).

3.2 Abstract Simulation for Control

We propose a strategy for producing high-level control policies via reinforcement learning in an abstract simulator. We created a reinforcement learning software suite in python which we call AbstractSim. Which provides an OpenAI gym interface for reinforcement learning[1]. This simulator models robots and balls as point masses in 2 dimensions, and uses very simple calculations for deciding on object collisions. These policies are trained to perceive the world as a vector to numbers encoding their own position and angle on the field and the ball's position. Each of our three distinct policies is trained in its own environment in AbstractSim. Figure 1 shows a screenshot of AbstractSim.

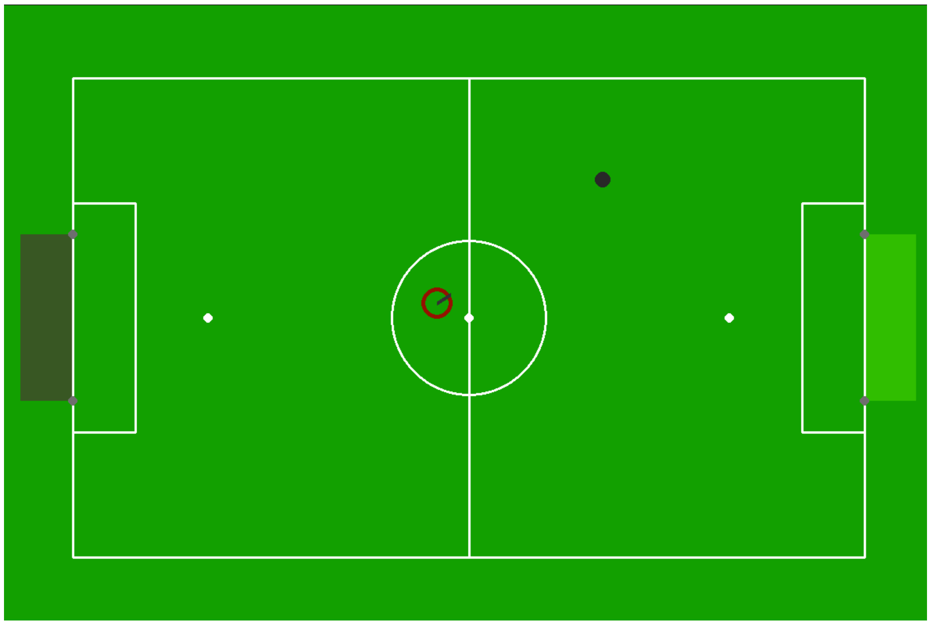

Fig. 1. A render of the push ball to goal environment in our abstract simulator.

4 Past History

We have no prior RoboCup competition experience as BadgerBots. Dr. Josiah Hanna previously participated on the UT AustinVilla team from 2015 – 2019.

5 Impact

Our aim is to elevate reinforcement learning as a strategy for developing robot control policies for the RoboCup Standard Platform League. Currently, many teams spend dozens of hours writing and tweaking behaviors for very specific situations (e.g., how should the robot move when near a goal post). This approach is *likely not scalable* as the league tries to play games with larger fields, more robots, and increasingly complex rules. Automating behavior development through reinforcement learning offers a path to scalable behavior development with less developer hours. We are starting with high level control policies, and eventually hope to replace even the low level controllers (i.e., walk and kick engines) in our fork of BHumanCodeRelease 2022 with reinforcement-learning derived neural control policies. Ultimately, we believe that learning will be crucial for realizing the RoboCup vision and part of our team's impact will be to spur others in this direction.

For the wider robotics community, RL is a promising approach for developing robots for tasks that are too complex for a programmer to specify optimal behavior. Demonstrating a competitive robot soccer team will require developing techniques that can scale the complexity of environments in which robots can be deployed. For the RL research community, RoboCup offers many challenges for today's RL algorithms and we hope that our efforts will inspire more RL researchers to participate in RoboCup.

6 Video Presentation

We provide a link to our demonstration video: https://www.youtube.com/watch?v=TT-3jCG4fqo

7 Acknowledgments

We would like to thank Shreyansh Sharma, Abishek Kumar, and Will Cong for their contributions to our RoboCup effort.

References

1. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: Openai gym (2016), https://github.com/openai/gym
2. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research **22**(268), 1–8 (2021), http://jmlr.org/papers/v22/20-1364.html
3. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (2017). https://doi.org/10.48550/ARXIV.1707.06347, https://arxiv.org/abs/1707.06347