Report on RoboCup Federation Project "the Research Challenge"

Aijun Bai, Feng Wu, Xiaoping Chen

Department of Computer Science, University of Science and Technology of China
xpchen@ustc.edu.cn

Abstract. This summary reports main efforts and results in the RoboCup Federation project "the Research Challenge". The original motivation of the project was to establish an abstract and simplified 2D competition as a research challenge of the 2D league. We have tried to fulfill the goal through a middleware that implements a set of high-level actions, instead of the soccer server primitive actions, for the agents. This allows one to concentrate on "high-level" research issues such as decision-making in a multi-agent setting as well as agent cooperation. It provides a new bridge for linking RoboCup competitions with the related basic research in AI and Robotics, such as planning and machine learning.

Keywords: robocup, soccer simulation 2d

1 Introduction

As one of oldest leagues in RoboCup, Simulation 2D has achieved great successes and inspired many researchers all over the world to engage themselves in this game each year [1]. Hundreds of research articles based on Simulation 2D have been published[1]. Comparing to other leagues in RoboCup, the key advantage of Simulation 2D is the abstraction made, which relieves the researchers from having to handle low-level robot problems such as object recognition, communications, and hardware issues [2]. The abstraction enables researchers to focus on higher level concepts such as cooperation and learning. One important goal of Simulation 2D is to serve as a testbed for AI and boost the research in related fields. The key challenges lie in the fact that it is a fully distributed, multi-agent stochastic domain with both teammates and opponents and the continual state, action and observation space [3].

It has been more than a decade since the first competition in 1997. However, the gap between basic AI research and Simulation 2D is still relatively large. For example, it is very hard to test and study some state-of-the-art planning and learning approaches (e.g. MDP/POMDP/DEC-POMDP planning, Reinforcement Learning, etc) in the 2D domain due to the extremely large problem size. In this report, we tried to develop an abstract and simplified 2D competition as a research challenge. Given the huge state space of Simulation 2D, it is

[1] `http://www.cs.utexas.edu/~pstone/tmp/sim-league-research.pdf`

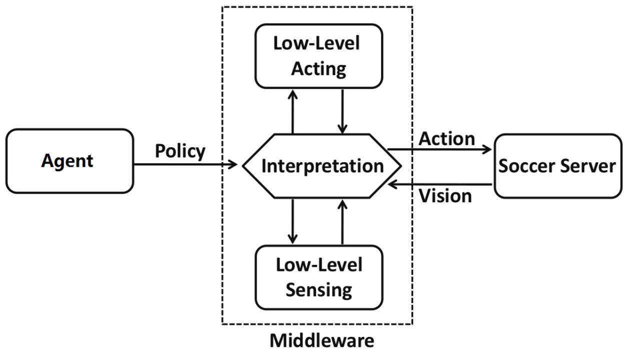

Fig. 1. The Architecture of the Middleware

more scalable to do planning and learning with only state samples and construct policies using abstracted actions and observations. A middleware is introduced to communicate between the agents and the soccer server. The middleware provides some simplified and abstracted interfaces for the agents. Furthermore, the middleware can benefit from the existing team codes for implementing the low-level acting and sensing functions. With the middleware, a robot soccer team can compete with other teams on the standard platform–the soccer server–used in RoboCup. The goal of this effort is to promote basic research with the proposed new competition. It is also hopeful to draw attentions to this new competition from researchers outside of RoboCup.

The remainder of this report is organised as follows. Section 2 presents the basic architecture of the middleware. Section 3 briefly introduces how the middleware modeling Simulation 2D. Section 4 describes some implementing details, and Section 5 concludes.

2 The Middleware

To achieve abstraction and simplification in Simulation 2D, a middleware is needed as the bridge between the soccer server and the agents. The basic functions of the middleware are twofold: (1) It receives the information from the soccer server, processes it and outputs a set of observations to the agents; (2) It takes high-level actions performed by the agents, interprets them, and sends a series of primitive commands to the soccer server. Currently, agents for this research challenge include only soccer players. Online coaches are not supported by the middleware. The middleware aims at modeling the Simulation 2D as a typical multi-agent system to promote basic research. The benefit of the middleware is to bridge the gap between the low-level function of the soccer server and the high-level decisions of the agent.

Algorithm 1 Example of Control Rules

```
if has-ball then
  if can-shoot then
    shoot-goal(best-shoot-direction)
  else if should-pass then
    pass-ball(best-pass-teammate)
  else
    dribble-ball(best-dribble-direction)
  end if
else if is-fastest-to-ball then
  intercept-ball
else
  go-formation
end if
```

With the help of high-level actions, the researchers can spend more time on high-level planning and learning, dealing with cooperation and competition among teammates and opponents. The complexities of this competition is significantly lower than the traditional 2D league, but it is much higher than the current benchmarks in common basic research domains. It is hopeful to offer sufficient complexities to be interesting from a research point of view. A series of planning and learning approaches are able to be directly tried and tested in this research challenge.

3 Modeling Simulation 2D

In order to incorporate domain knowledge and simplify the planning task, the agents choose actions not from the simulator's primitive actions such as `dash` and `kick`, but from higher level actions constructed from skills used in our WrightEagle team. Depending on whether the agent possessing the ball or not, the high-level actions include:

– `pass-ball(`k`)`: The agent directly kicks the ball to the teammate k.
– `dribble-ball(`d`)`: The agent dribble the bal in direction d.
– `shoot-goal(`d`)`: The agent kicks the ball towards the goal in direction d.
– `go-to-position(`p`)`: The agent moves to a specified position p by dashing and turning.
– `intercept-ball`: The agent moves directly towards the interception point to get the ball. The interception point is calculated based on the dynamic information of the ball.
– `do-block`: The agent blocks the opponent ball controller, hassle it and try to steal the ball.
– `do-mark(`k`)`: The agent marks opponent k to destroy potential passing opportunities.
– `go-formation`: The agent dashes towards a position in order to maintain a predefined formation of the team.

Table 1. Messages from Agents to Middleware

Message	**Description**
(init *TeamName* [goalie])	Initialization message
(goto x y)	Go to position (x, y)
(pass k)	Pass ball to teammate k
(dribble d)	Dribble ball in direction d
(shoot d)	Shoot to goal in direction d
(intercept i)	Intercept the ball at cycle i
(block)	Block opponent ball controller
(mark k)	Mark opponent k
(formation)	Go to formation position

It is straightforward to include other high-level actions in this model, e.g. other types of passing or dribbling behaviors. Using these high-level actions, researchers can manually write simple control rules (a.k.a the policies) for agents. A naive example is shown in Algorithm 1.

In the above example, the abstract observations are the combinations of the following information:

– `has-ball`: The agent is possessing the ball.
– `is-fastest-to-ball`: The agent is fastest to get the ball.
– `can-shoot`: The agent is able to score by shooting the ball to the goal.
– `should-pass`: The agent should pass the ball to one of its teammates instead of dribbling by itself.

Note that the observation variables may have interdependence, e.g. `can-shoot` and `should-pass` can be true if and only if the variable `has-ball` is true. Therefore the structure of the observation space is exploited to further simplify the model. In fact, rather than boolean variables, `can-shoot` and `should-pass` are designed to be real values in practice to represent the chance to score or the reward of passing instead of dribbling. Along with these abstract observations, some low-level observations are also provided by the middleware, including (but not limited to):

– `ball-position`: The global position of the ball.
– `ball-velocity`: The global velocity of the ball.
– `teammate-position(`k`)`: The global position of teammate k.
– `opponent-position(`k`)`: The global position of opponent k.
– `teammate-intercept-cycle(`k`)`: The interception cycle of teammate k.
– `opponent-intercept-cycle(`k`)`: The interception cycle of opponent k.

Low-level observations may be useful for constructing more precise controller. The full set of observation variables are not going to be described in this report for simplicity.

Table 2. Messages from Middleware to Agents

Message	**Description**
(init *Side Unum*)	Response to initialization message
(position ball x y)	Ball's current position is (x, y)
(velocity ball x y)	Ball's current velocity is (x, y)
(position teammate k x y)	Teammate k's current position is (x, y)
(position opponent k x y)	Opponent k's current position is (x, y)
(intercept cycle teammate k i)	Teammate k's interception cycle is i
(intercept cycle opponent k i)	Opponent k's interception cycle is i
(has_ball)	Whether the agent is possessing the ball
(can_shoot c)	The chance to shoot
(should_pass c)	The chance to pass
(is_fastest_to_ball)	Be fastest to get the ball

4 Implementing the Middleware

To implement the middleware, we borrow ideas from RL-Glue [4], a language-independent software for reinforcement learning experiments. The architecture of the middleware is shown in Figure 1. We develop protocols for communication between the agents and the middleware. The protocols describe the input and outcome of the agents, i.e. the high-level actions and abstract observations. In the middleware, the messages received from the soccer server are processed by the low-level sensing component and the high-level actions of the agent are interpreted by the low-level acting component. The low-level components are implemented by other existing team sources such as WrightEagleBase. For simplicity, the middleware pretends to be the whole team and integrates all the messages originally sent to each agent (including the two online coaches) in standard Simulation 2D. Hence the middleware can model a relatively precise world state. These "true" state information are then sent to the agents instead of local and noisy ones. Parts of protocols are briefly described in Table 1 and Table 2.

Similar to RL-Glue, the middleware can be used either in internal or external mode. In internal mode, all the components are linked into a single program, and their communication is through function calls as long as all of them are written exclusively in C/C++. In external mode, the agent, the low-level acting and sensing components are into separate programs. Each program connects to the interpreter program, and all communication is over TCP/IP socket connections. External mode allows these programs to be written in any programming language that supports socket communication such as C/C++, Java, Python, etc. Each mode has its strengths and weaknesses. Internal mode has less overhead while external mode is more flexible and portable.

Figure 2 shows the network topology of a match when starting in external mode with full teams. Each arrow in the network topology indicates an individual socket instance. The middleware totally creates 24 UDP/IP connections to the soccer server, pretending as 11 soccer players and one online coach for each of the

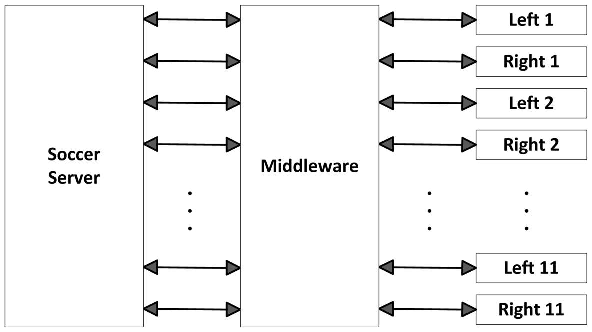

Fig. 2. The Network Topology of a Match

two teams. The protocols used between the middleware and the soccer server stay the same as the original Simulation 2D match. Simultaneously, the middleware maintains the other 22 connections for the research challenge agents, denoting as Left 1, Right 1, etc. Note that, the middleware can run with no trouble if only parts of the agents are connected. In this case, it creates connections to the soccer server respectively. Hence, from the server side, there will be no difference between this research challenge match and the standard Simulation 2D match. It is hopeful that the team developed over the middleware can compete with other existing teams developed by Simulation 2D community.

5 Conclusion

In this report, we present a novel solution to bridge the gap between basic AI research and RoboCup Soccer Simulation 2D by launching an abstract and simplified 2D competition as a research challenge. The middleware enables agents to interact with the soccer server through a set of high-level actions, instead of original primitive ones. The protocols used by the middleware formulate the messages of the soccer server as abstracted actions and observations.

The first competition of this research challenge is going to be launched in RoboCup China Open 2012. Before that, we will encourage some national teams to take part in this competition. It is hopeful to test the platform and collect some data during the competition, which will be very useful for further testing and development. We hope that this research challenge will serve as a new testbed for AI community and boost the research in related fields.

6 Acknowledgments

We would like to thank the Steering Committee members of this project (Daniel Polani, Peter Stone, Noda Itsuki, Sahar Asadi and Luis Paulo Reis). Special thanks to Daniel Polani, Peter Stone and Noda Itsuki for helpful discussions on this project.

References

1. Gabel, T., Riedmiller, M.: On progress in robocup: the simulation league showcase. RoboCup 2010: Robot Soccer World Cup XIV (2011) 36–47
2. Chen, M., Foroughi, E., Heintz, F., Huang, Z., Kapetanakis, S., Kostiadis, K., Kummeneje, J., Noda, I., Obst, O., Riley, P., et al.: Robocup soccer server for soccer server version 7.07 and later. (2003)
3. Stone, P.: Layered learning in multiagent systems: A winning approach to robotic soccer. The MIT press (2000)
4. Tanner, B., White, A.: Rl-glue: Language-independent software for reinforcement-learning experiments. The Journal of Machine Learning Research **10** (2009) 2133–2136