SRC 2018 Team Description Paper

Ren Wei*, Wenhui Ma, Zongjie Yu, Wei Huang, Shenghao Shan

Fubot Shanghai Robotics Technology Co. LTD,
Shanghai, People's Republic of China
ninjawei@fubot.cn

Abstract. In this paper, we present our robot's hardware overview, software framework and free-kick strategy. The free-kick tactic plays a key role during the competition. This strategy is based on reinforcement learning and we design a hierarchical structure with MAXQ decomposition, aiming to train the central server to select a best strategy from the predefined routines. Moreover, we adjust the strategy intentionally before the final. Our team won the first place in the SSL and we hope our effort can contribute to the RoboCup artificial intelligent progress.

Keywords: RoboCup, SSL, MAXQ, free-kick, reinforcement learning.

1 Introduction

As a famous international robot event, RoboCup appeals to numerous robot enthusiasts and researchers around the world.

We introduce the hardware overview and software framework in this paper. The software framework has a plugin system which brings extensibility. For the high level strategy, our energy is focused on the free-kick because we want to find a more intelligent and controllable one. Controllable means that we hope our team can switch strategy in case that the opponent change their strategy in next game. The intelligent and the controllable are not contradictory. Many research also indicate the importance of free-kick [1][3].

In recent years, many applications about reinforcement learning have sprung up, for instance, the AI used in the StarCraft and DotA. These applications require the cooperation between agents and the RoboCup is a perfect testbed for the research of reinforcement learning for its simplified multi-agents environment and explicit goal. In this context comes our free-kick strategy.

The remainder of this paper is organized as follows. Section 2 describes the overview of the robot's hardware. Section 3 presents the details of robotics framework we used. Section 4 introduces the markov decision process (MDP) and the MAXQ method in the 4.1, then illustrates the application in our free-kick strategy. The Section 5 shows the result. Finally, Section 6 concludes the paper and points out some future work.

2 Hardware

In this part, we describe the overview of the robot mechanic design. The controller board is shown in Figure 1 and the mechanical structure is in Figure 2.

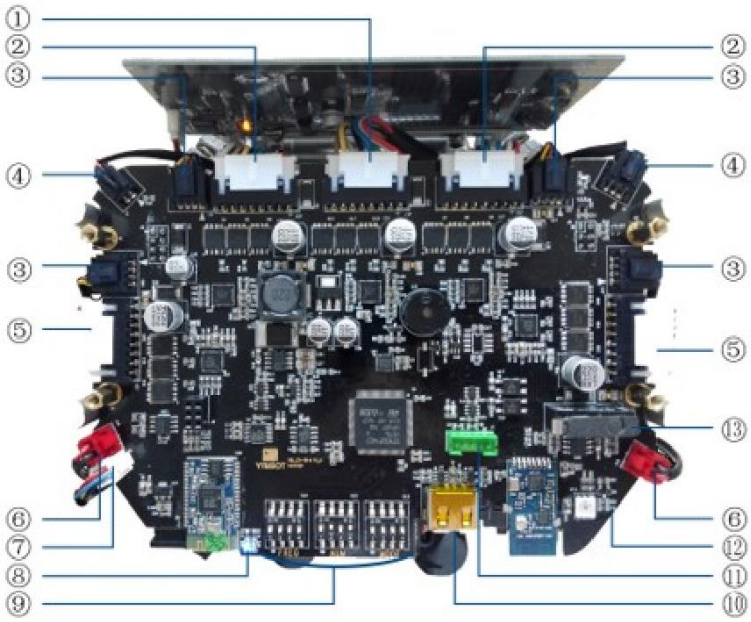

Fig. 1. Controller board overview

Our CPU is STM32F407VET6. The main components are:

(1) Colored LED interface
(2) Motor Controller interface
(3) Encoder interface
(4) Infrared interface
(5) Motor interface
(6) Speaker interface
(7) LED screen interface
(8) Mode setting switcher
(9) Bluetooth indicator
(10)Debug interface
(11)Joystick indicator
(12)Booster switcher

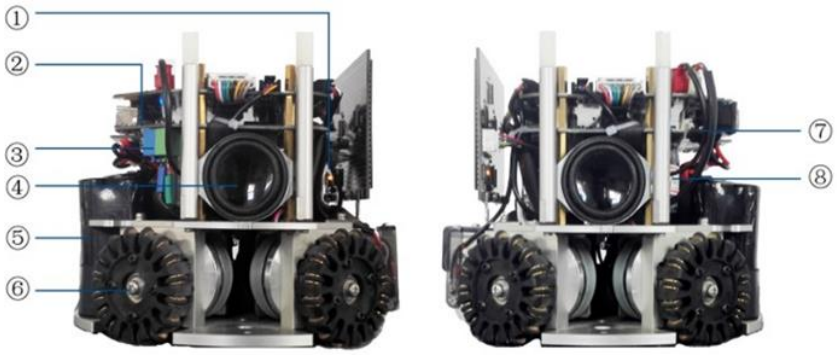

Fig. 2. Mechanical structure

(1) LED screen
(2) Charge status indicator
(3) Kicker mechanism
(4) Bluetooth Speaker
(5) Battery
(6) Universal wheel
(7) Power button
(8) Energy-storage capacitor

3 Software Framework

RoboKit is a robotics framework developed by us, as shown in Figure 3. It contains plugin system, communication mechanism, event system, service system, parameter server, network protocol, logging system and Lua Script bindings etc. We develop it with C++ and Lua, so it is a cross platform framework (working on windows, Linux, MacOS etc.). For SSL, we developed some plugins based on this framework, such as vision-plugin, skill-plugin, strategy-plugin etc. Vision-plugin contains multi-camera fusion, speed filter and trajectory prediction. Skill-plugin contains all of the basic action such as kick, intercept, chase, chip etc. And strategy-plugin contains defense and attack system.

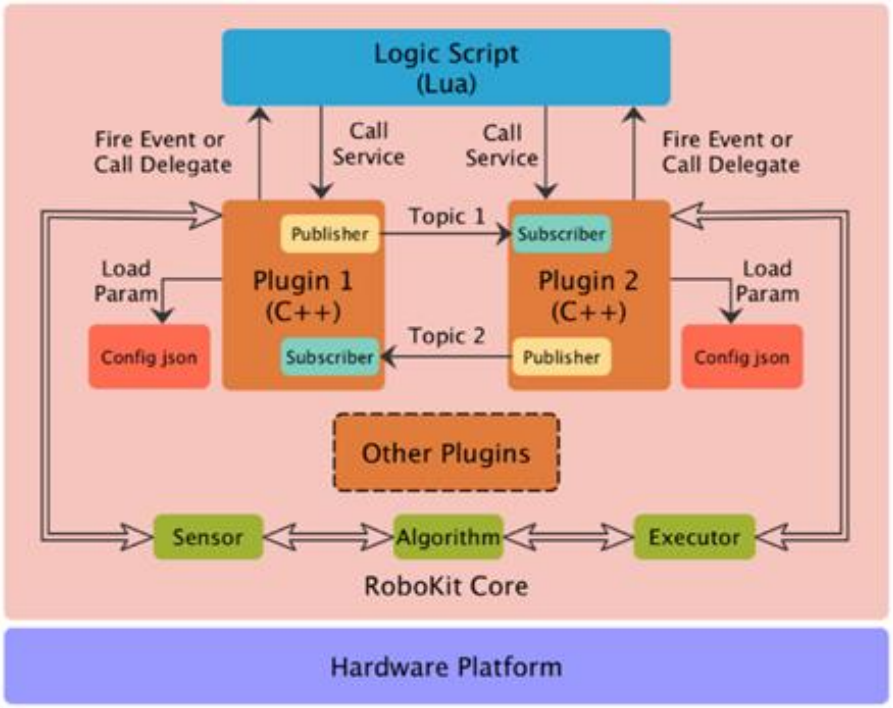

Fig. 3. RoboKit structure

4 Reinforcement Learning

Reinforcement learning has become an important method in RoboCup. Stone, Veloso[4][5][13], Aijun[2], Riedmiller[14] et al. have done a lot of work on the online learning , SMDP Sarsa(λ) and MAXQ-OP for robots planning.

Free-kick plays a significant role in the offense, while the opponents' formation of defense are relatively not so changeable. Our free-kick strategy is inspired from that a free-kick can also be treated as a MDP and the robot can learn to select the best free-kick tactics from a certain number of pre-defined scripts. For the learning process, we also implement the MAXQ method to handle the large state space.

In this chapter we will first briefly introduce the MDP and MAXQ, further details can be found here[10]. Then, we will show how to implement this method in our free-kick strategy, involving the MDP modeling and the sub-task structure construction.

4.1 MAXQ decomposition

The MAXQ technique decomposes a markov decision process M into several sub-processes hierarchically, denoted by $\{M_i, i = 0, 1, \dots, n\}$. Each sub-process M_i is also a MDP and defined as $\langle S_i, T_i, A_i, R_i \rangle$, where S_i and T_i are the active state and termination set of M_i respectively. When the active state transit to a state among T_i, the M_i is solved. A_i is a set of actions which can be performed by M or the subtask M_i. $R_i(s'|s, a)$

is the pseudo-reward function for transitions from active states to termination sets, indicating the upper sub-task's preference for action a during the transition from the state s' to the state s. If the termination state is not the expected one, a negative reward would be given to avoid M_i generating this termination state[10]

$$Q_i^*(s,a) = V^*(a,s) + C_i^*(s,a) \tag{1}$$

Where $Q_i^*(s,a)$ is the expected value by firstly performing action M_i at state s, and then following policy $\boldsymbol{\pi}$ until the M_i terminates. $V^{\pi}(a,s)$ is a projected value function of hierarchical policy $\boldsymbol{\pi}$ for sub-task in state s, defined as the expected value after executing policy $\boldsymbol{\pi}$ at state s, until M_i terminates.

$$V^*(i,s) = \begin{cases} R(s,i) & if\ M_i\ is\ primitive \\ max_{a \in A_i} Q_i^*(s,a) & otherwise \end{cases} \tag{2}$$

$C_i^*(s,a)$ is the completion function for policy $\boldsymbol{\pi}$ that estimates the cumulative reward after executing the action M_a, defined as:

$$C_i^*(s,a) = \sum_{s',N} \gamma^N P(s',N|s,a) V^*(i,s') \tag{3}$$

The online planning solution is explained in[2], and here we list the main algorithms.

```
Algorithm 1. OnlinePlanning ()
Input: an MDP model with its MAXQ hierarchical structure
Output: the accumulated reward r after reaching a goal
r ← 0
s ← GetInitState()
r ← r + InitialAction(a_0, s)
while s ∉ G_0
do
   ⟨v, a_p⟩ ← EvaluateState(0, s, [0,0,⋯,0])
   r ← r + ExecuteAction(a_p, s)
   GetNextState()
return r
```

Here we set an initial action update before the system start updating. The initial action enable us to modify the strategy according to the opponent's defense formation.

```
Algorithm 2. EvaluateState(i,s,d)
Input: subtask M_i, state s and depth array d
Output: ⟨V*(i,s), a_p*⟩
if M_i is primitive then return ⟨R(s,M_i), M_i⟩
else if s ∉ S_i and s ∉ G_i then return ⟨−∞, nil⟩
else if s ∈ G_i then return ⟨0, nil⟩
else if d[i] ≥ D[i] then return ⟨HeuristicValue(i,s), nil⟩
else
    ⟨v*, a_p*⟩ ← ⟨−∞, nil⟩
```

```
    for M_k ∈ Subtasks(M_i) do
     if M_k is primitive or s ∉ G_k then
        ⟨v, a_p⟩ ← EvaluateState(k, s, d)
        v ← v + EvaluateCompletion(i, s, k, d)
        if v > v*
          ⟨v*, a_p*⟩ ← ⟨v, a_p⟩
    end
return ⟨v*, a_p*⟩
```

Algorithm 2 summarizes the major procedures of evaluating a subtask. The procedure uses an AND-OR tree and a depth-first search method. The recursion will end when:

(1) the subtask is a primitive action;
(2) the state is a goal state or a state outside the scope of this subtask;
(3) a certain depth is reached.

```
Algorithm 3. EvaluateCompletion (i,s,a,d)
Input: subtask M_i, state s, action M_a and depth array d
Output: estimated C*(i,s,d)
G̃_a ← ImportanceSampling(G_a, D_a)
v ← 0
for s' ∈ G̃_a do
  d' ← d;
  d'[i] ← d'[i] + 1
  v ← v + 1/G̃_a EvaluateState(i, s', d')
end
return v
```

Algorithm 3 shows a recursive procedure to estimate the completion function, where $\widetilde{G}_a$ is a set of sampled states drawn from prior distribution D_a using importance sampling techniques.

4.2 Application in free-kick

Now we utilize the techniques we mentioned in our free-kick strategy. First we should model the free-kick as a MDP, specifying the state, action, transition and reward functions.

State space: As usual, the teammates and opponents are treated as the observations of environment. The state vector's length is fixed, containing 5 teammates, 6 opponents, a ball and the agent self.

Action space: For the free-kick, the actions includes kick, turn and dash. They are in the continuous action space.

Transition: We predefined 60 scripts which tell agent the behavior of team-mates. These scripts are chosen randomly. For the opponents, we simply assume them moving

or kicking (if kickable) randomly. The basic atomic actions is modeled from the robot kinematic analysis.
Reward function: The reward function considers not only the ball scored, which may cause the forward search process terminates without rewards for a long period. Considering a free-kick, a satisfying serve should never be intercepted by the opponents, so if the ball pass through the opponents, we give a positive reward. Similarly, we design several rewards function for different sub-tasks.

Next, we implement MAXQ to decompose the state space. Our free-kick MAXQ hierarchy is constructed as follows:

Primitive actions: We define three low-level primitive actions for the free-kick process: the kick, turn and dash. Each primitive action has a reward of -1 so that the policy reach the goal fast.
Subtasks: The kickTo aims to kick the ball to a direction with a proper velocity, while the moveTo is designed to move the robot to some locations. To a higher level, there are Lob, Pass, Dribble, Shoot, Position and Formation behaviors where:

(1) Lob is to kick the ball in the air to lands behind the opponents;
(2) Pass is to give the ball to a teammate.
(3) Dribble is to carry the ball for some distance.
(4) Shoot is to kick the ball to score.
(5) Position is to maintain the formation in the free-kick.

Free-kick**:** The root of the process will evaluate which sub-task should the place kicker should take.

Our hierarchy structure is shown in Figure 4. Note that some sub-tasks need parameters and they are represented by a parenthesis.

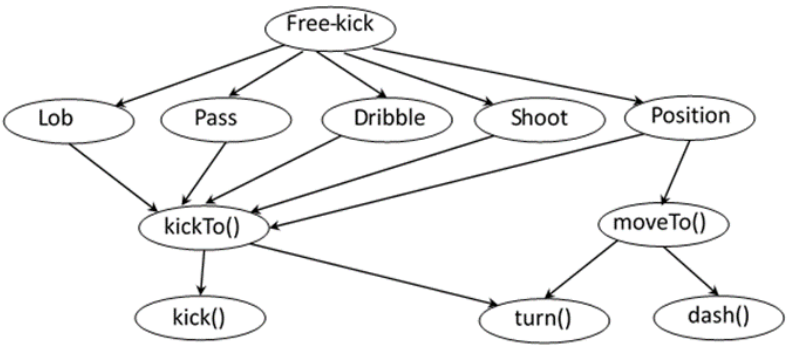

Fig. 4. Hierarchical structure of free-kick

5 Conclusion

This paper presents our robot's hardware and software framework. We implement the reinforcement learning in our free-kick tactic. Based on the related work, we divide the free-kick into some sub-tasks and write some hand-made routines for the learning process. Our contribution lies in the realization of reinforcement learning in the SSL, which is a first step from simulation to reality.

References

1. Mendoza, J. P., Simmons, R. G., & Veloso, M. M. (2016, June). Online Learning of Robot Soccer Free Kick Plans Using a Bandit Approach. In ICAPS (pp. 504-508).
2. Bai A., Wu F., Chen X.: Towards a Principled Solution to Simulated Robot Soccer. In: Chen X., Stone P., Sucar L.E., van der Zant T. (eds) RoboCup 2012. Lecture Notes in Computer Science, vol. 7500, pp. 141-153. Springer, Heidelberg (2013).
3. Cooksey, P., Klee, S., & Veloso, M. (2016). CMDragons 2015: Coordinated Offense and Defense of the SSL Champions. RoboCup 2015: Robot World Cup XIX, 9513, 106.
4. Cooksey, P., Klee, S., & Veloso, M. (2016). CMDragons 2015: Coordinated Offense and Defense of the SSL Champions. RoboCup 2015: Robot World Cup XIX, 9513, 106.
5. Stone, P., Sutton, R., Kuhlmann, G.: Reinforcement learning for robocup soccer keepaway. Adaptive Behavior 13(3), 165–188 (2005).
6. Kalyanakrishnan, S., Liu, Y., Stone, P.: Half field offense in roboCup soccer: A multiagent reinforcement learning case study. In: Lakemeyer, G., Sklar, E., Sorrenti, D.G., Takahashi, T. (eds.) RoboCup 2006. LNCS (LNAI), vol. 4434, pp. 72–85. Springer, Heidelberg (2007).
7. Thrun, S., Burgard, W., Fox, D.: Probabilistic robotics, vol. 1. MIT press, Cambridge (2005).
8. Bruce, J., & Veloso, M. (2002). Real-time randomized path planning for robot navigation. In Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on (Vol. 3, pp. 2383-2388). IEEE.
9. Yangsheng Ye, Yue Zhao, Qun Wang, Xiaohe Dai, Yuan Feng and Xiaoxing Chen: SRC Team Description Paper for RoboCup 2017 (2017).
10. Dietterich, T.G.: Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Machine Learning Research 13(1), 63 (May 1999).
11. Ren, C., Ma, S.: Dynamic modeling and analysis of an omnidirectional mobile robot. In: Intelligent Robots and Systems (IROS), pp. 4860–4865 (2013).
12. Kober, J., Mülling, K., Krömer, O., Lampert, C.H., Schölkopf, B., Peters, J.: Movement templates for learning of hitting and batting. In: IEEE International Conference on Robotics and Automation 2010, pp. 1–6 (2010).
13. Stone, P.: Layered learning in multi-agent systems: A winning approach to robotic soccer. The MIT press (2000).
14. Gabel, T., Riedmiller, M.: On progress in roboCup: The simulation league showcase. In: Ruiz-del-Solar, J. (ed.) RoboCup 2010. LNCS (LNAI), vol. 6556, pp. 36–47. Springer, Heidelberg (2011).