MT2022: Team Description Paper

Linsheng Guan, Qiuna Chen, Jingjing Wang, Haotian Xiang, Shuaishuai Meng, Cunyuan Wang, Haoyu Xu, Haicheng Fang, Shengbing Chen

College of Artificial Intelligence and Big Data,
Hefei University,
Anhui Province, P.R.China, 230601
hfuumt@163.com

Abstract. MT2022 is a team of soccer simulation 2D league. Members of the team are from Hefei University, and they have a strong passion for robots. Since 2012, the MT2022 team has participated in RoboCup ChinaOpen tournament and RoboCup every year and has achieved many good results. This paper briefly describes the backgroud of MT2022 and the main works of our team since the RoboCup 2021. Through these works we have greatly improved the competive ability of our team.

Keywords: Log Data Mining Tool, Half Field Offense, Keepaway Mode, Shoot Training, Dribble Training, Freekick Etc Strategy, Intercept Strategy.

1 Introduction

MT2022 team comes from Hefei University in China. We actively participated in various of RoboCup competitions, and achieved good results. We have won the top eight 5 times in soccer simulation 2D league of RoboCup since 2015. Through the game, we have made lots of friends, and the strategy of MT team was recognized by many teams. Main achievements in recent years include: the third place of RoboCup 2018, the fifth place of RoboCup 2019, and the champion of RoboCup Portugal Open 2016. Through extensive training, testing, and logging data mining, we found some shortcomings of the team and optimized the team code in many ways. Now, we have improved the strategy and codes. we hope this could improve the level of our team, and get the effect in RoboCup 2022.

In the RoboCup 2022 World Cup competition, we hope to show the latest research results and team technology and achieve better results.Actively discuss, learn and communicate with other teams during the game.

2 The Underlying Of The MT2022

The MT's team base is agent2d[7]. We use librcsc as the underlying library. The way action chains are used at the bottom. We modify and optimize on the basis of MT2021[2].

3 Tools

To do a good job, he must sharpen his tools.

3.1 HFO

HFO[5,6] stands for Half Field Offense in the RoboCup Soccer Simulation. Half Field Offense is a subtask in RoboCup simulated soccer, simulating the situation that a team's attack must cross the opponent's defense to shoot. This repository provides the ability to quickly and easily connect your learning agent to an HFO domain. HFO provides interfaces for C++ and Python. This year we have made changes to the HFO source code to link MT and other team binaries. We use HFO for training and testing to verify the feasibility of the shoot algorithm we designed.

Fig.1. 2v1 in HFO

```
EndOfTrial: 92 / 200 20601 GOAL-11
TotalFrames = 20602, AvgFramesPerTrial = 103.0, AvgFramesPerGoal = 98.4
Trials             : 200
Goals              : 92
Defense Captured   : 36
Balls Out of Bounds: 72
Out of Time        : 0
[start.py] Cleaning up server and other processes
```

Fig.2. Training results in 2V1 mode in HFO

3.2 Keepaway

Keepaway is a subtask of robot soccer in which one team, the keepers, tries to maintain possession of a ball within a limited region, while another team, the takers, tries to gain possession. Parameters of the task include the size of the region, the number of keepers,

and the number of takers. To implement keepaway mode, set server::keepaway=true in the .rcssserver/server.conf file. Parameters server::keepaway_length, server::keepaway_width to set the size of the keepaway area. Keepaway through 2V2, 2V3 and other small scale simulation, so it is a little smaller than the normal football match.Because the situation is limited, different learning methods can be applied in real time. In Keepaway mode, there are keeper and Taker roles.Taker repeatedly grabs the ball until it goes out of the rectangular area as a play ends. We think running keepaway is more time efficient and purposeful than running at all. After understanding the characteristics of keepaway mode, we mainly use Keepaway to train and test our intercepts. According to the feedback from a lot of tests and training, the algorithm of block timing and target point of intercepting is optimized.

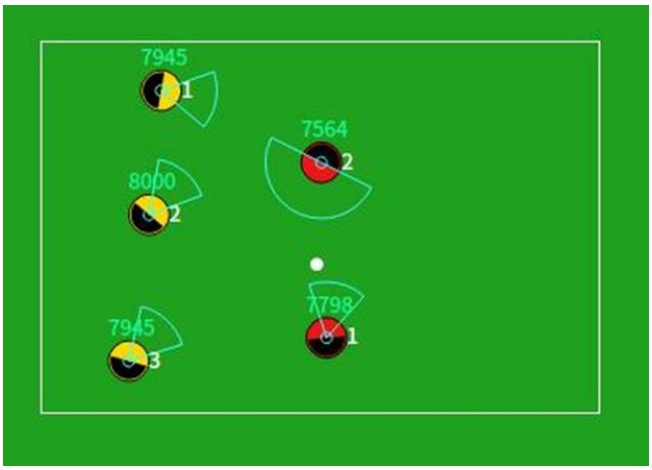

Fig.3. 3V2 in Keepaway mode

3.3 Log Data Mining Tool

The rcg and rcl files generated by the team during the game record various information such as the position and speed of the ball and the position of the players in each cycle during the game. In order to better analyze ourselves and other teams, we wrote a log data mining tool in Python to mine the information we needed from the log files and save it into .csv files. We conduct data mining and data analysis on the extracted information, and use the obtained analysis results for parameter tuning and algorithm optimization. We've been working hard to optimize our log data mining tools.

3.4 The Data Analysis

We use ofstream in MT code, output the field information and save it to .csv file, and do data mining on it. Adjust algorithms and strategies through feedback results.

3.5 Build A Database Of Different Teams

Know your enemy and win every battle. With the help of log data mining tools, we collected the offensive and defensive advantages and loopholes of each team (including MT), and input the relevant data into the database. Increased understanding of other teams' tactics, shortcomings, etc. We take the best and discard the worst from the database, learn from other teams' good offensive and defensive strategies, and develop specific strategies for teams with obvious flaws. It worked well. We will continue to improve and enrich the database.

4 Action Chain Movement Adjustment And Optimization

4.1 Shoot Training

Shoot is the ultimate goal of attack, but also the key to the outcome of the game.

We have further developed the log data mining tool on the original basis. We've extracted, such as the period of shoot and scoring, the x coordinate of the ball, the y coordinate of the ball, the fractional velocity of the ball in the x direction, the fractional velocity of the ball in the y direction, and we've extracted it and saved it in a .csv file. Then, data cleaning is carried out, mainly excluding the goals scored by our side in the way of tackle, the goals mistakenly put into the tackle by the enemy players, and the goals kicked (screening conditions are the vector positive and negative value of the speed of the ball x y direction, the size of the speed of the ball, etc.).

We use this tool to verify the effectiveness of the shoot algorithm we designed. We take MT2021 as the initial code, and take shoot view, shoot judgment condition decision, shoot angle and shoot speed as the main optimization direction. Control variable method is used to set up six teams, Team1(MT2021), Team2(MT2021 based on the adjustment of shoot view), Team3(MT2021 based on the adjustment of shoot judgment condition decision), Team4(MT2021 based on the adjustment of shoot angle), Team5(MT2021 based on the adjustment of shoot speed), Team6(Based on MT2021, adjust the shoot view of vision, adjust the shoot judgment condition decision, shoot angle, shoot speed). We used HFO's 2V1 mode for training and each Team played 1000 matches against HELIOS[9] to get the match log.

View adjustment training Through the data mining of the game log files, we found that the targeted adjustment of view (including turning neck, turning body rotation, adjusting the width of view, adjusting the quality of view). It is helpful to improve the accuracy of obtaining real-time information on the field, and improve the reliability of predicting the information of players on the field after several cycles, which is conducive to the decision-making of shoot.

Shoot judgment conditional decision making In agent2d, shoot generation: ShootGenerator is the action generator for shoot. It generates shoot movements based on the situation on the field, and gives a score for each successful shoot. Action information

about shoot is stored in std::vector<course>. Shoot execution: in SamplePlayer::doShoot() call Bhv_StrictCheckShoot().execute(this) to execute std::vector<course> ShootGenerator::ScoreCmp() ShootGenerator::ScoreCmp().

We adjusted the decision of shoot judgment condition and achieved certain effect. We do data mining on the game log files. After reading the shoot_generator code, we found that n_step, the period in which the enemy can intercept the ball, has a greater effect on the shot conditioning than any other factor. The formula for n_step is int n_step = (n_turn==0 ? n_turn + n_dash : n_turn + n_dash + 1). N_turn is the number of cycles it takes for the enemy to adjust their body turn, and n_dash is the number of cycles it takes for the enemy to get the ball and sprint.We did a lot of training, modified and optimized the n_turn and n_dash algorithms, and relaxed the shoot conditions. In the same case, the current algorithm is more likely to trigger the shoot action, and sometimes the effect of a long shot, the effect is better.

Shoot angle training The best thing about shoot into a blind corner (far or near, near the goal post) is that it is difficult for opposing goalkeepers and other defenders to tackle the ball because of the Angle. Moreover, when the defender is in the near corner, most teams will choose to shoot to the far corner when attacking. It takes a long time for the defender to intercept the ball and run a longer distance (compared with other shots), which increases the difficulty of defending and improves the success rate of shoot. The disadvantage of shoot into a dead end is that sometimes the ball can then hit the post or out of the woodwork or the bottom line.

How to achieve the first shot dead Angle when shoot? We are responsible for the shoot shoot in the generator shoot_generator grade to grade function ShootGenerator shot::evaluateCourses(const WorldModel &wm) make the changes. Increased the score weight on shots where the target point is dead. The algorithm of scoring function was optimized reasonably, and a lot of training was carried out to achieve more dead corner shots.

We used log data mining tools for HFO's 2V1 mode training (each Team played 1000 matches against HELIOS), and the game logs were obtained for data mining and visual analysis. As shown in Fig.4, the evaluation factors of the pie charts from the first to the fourth lines are shoot distance, ball speed when shoot, ball speed when scoring, and absY(absolute value of y coordinate) when the ball passes the goal line. The pie charts from the first column to the sixth column are team1 to team6 respectively. The different colors in the rightmost example chart represent the specific values of the evaluation factors in each pie chart. According to the statistics of successful shoot times, the success rates of shoot from Team1 to Team6 are 81.4%, 80.9%, 78.1%, 78.3%, 86.3% and 83.3% respectively. As can be seen from the shot success rate and Figure 4, compared with the original code Team1, Team5 that adjusted the rate of fire (try to shoot at a higher speed) had the best effect and the most improved success rate.

However, the success rate of Team3 and Team4 in shoot judgment conditional decision and shoot angle training decreased, but the effect of long-range shoot and dead angle shoot was realized. This is what we need against more defensive teams, but we need to improve our shoot success rate. Team6, which combines all the tweaks together, is also somewhat better than the original code.

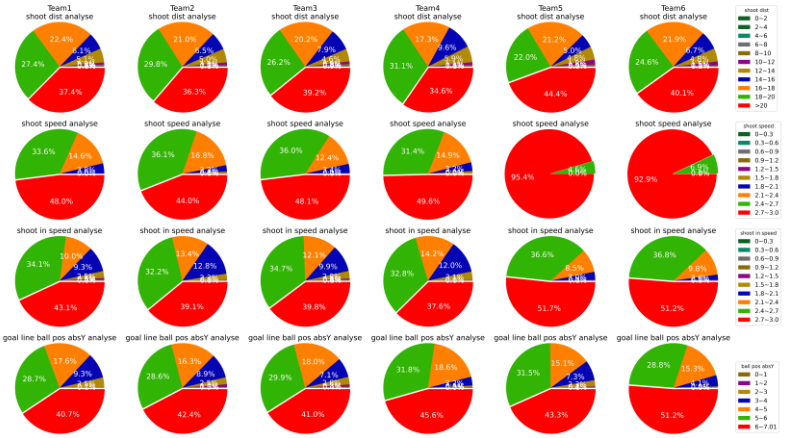

Fig.4. Data mining of shoot information

4.2 Dribble Training

In the game, when we have the ball, it allows us to advance into the penalty area at a faster pace, which is good for the fast counter-attack style. We didn't use our quick counter-attack very well and it was sloppy. The main reason is that, when attacking with the ball, players seldom take the ball quickly to the opponent's penalty area, but prefer to pass the ball to teammates, and many of them are back to teammates. This will cause the enemy will have more time to defend, our attack lack of pressure. In response, we optimized the environmental assessment of the relevant situation, and our ball carrier gave priority to pass the ball to the teammate whose coordinate was larger than that of the ball during the quick counter-attack. The player with the largest x coordinate breaks through to the offside line with fast dribble (if it is not safe to pass the ball to a teammate whose x coordinate is larger than the ball's x coordinate, the ball holder tries to break through with the ball with fast dribble) and tries to pass the ball. After passing, if in the side is the bottom, in the middle of the single blade. We have designed two new fast dribble strategies, one is fast dribble when the ball is away from the body when it is safe, and one is fast dribble when the ball is not in the body when it is not safe. We carried out a large amount of training, and according to the success of carrying the ball, adjust the algorithm, improve the success rate and effectiveness of carrying the ball.

5 Adjustment And Optimization Of Running Strategy

5.1 Freekick Etc Strategy

We start running and passing drills In the game, due to the free_kick, kick_in, etc., waiting for the freekick etc time is longer.Some teams, such as Germany's FRA-United, will use a marking strategy and stick close to our players. Makes freekick etcs more difficult, making it harder to pass the ball to teammates. In the RoboCup 2021, we were repeatedly robbed by the marking teams such as FRA-UNIted[8] at the freekick etc, so we were very passive on the pitch. We adjusted our freekick etc strategy a little bit before we were about to freekick etc. When the timing is right, our receiving team, except for the kicker, runs to a certain coordinate around them (random and the target point is 4-8 meters away from them).

The Opponent's Positioning Strategy For The Freekick Etc We designed a marking strategy for freekick etc. Stand close to the opponent player who is close to you at freekick etc and keep an eye on the opponent player (except the freekick etc player).

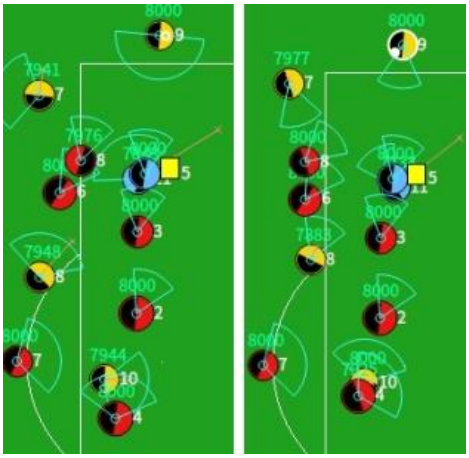

Fig.5. before our freekick

Fig.6. before the opponent's kick_in

5.2 Intercept Strategy

Interception can be understood as: when the interception condition is met, the player performs a series of actions such as do dash and do turn to get the ball. There are three key points in the interception: one is the judgment of the timing of the interception, called block. The second is the calculation of the interception point. The third is how to reach the interception point. We mainly modified and optimized the calculation of the interception point and how to reach the interception point. On the basis of the original algorithm, we have adjusted and optimized the algorithm, including distance,

strength value, turn, sprint and so on.Run to the spot where the ball is taken for the shortest period of time (consider the distance you need to travel, etc.). Adjusted the stamina usage algorithm related to stamina to sprint with as much stamina as possible when given the opportunity to get the ball.Adjust body steering more properly (the latest rcssserver can turn body 360 degrees). Increased initiative to get the ball and get it faster.

6 Summary And Prospect

Since the establishment of MT team in 2012, we have been working hard to optimize the algorithm. Enrich the style of play and adjust the strategy of attack and defense to make the team move more intelligent and cooperate more efficiently. We plan to use more machine learning, reinforcement learning, and deep learning methods to develop MT in the future. We are looking forward to working with excellent teams from all over the world to make progress together and wish RoboCup better and better.

References

1. Zheng Yang, Ziqiang Liu, Xiaorui Wang, Ning Dong, Xiangben Hu, JingLi, Shengbing Chen, Gang Lv, MT2018 Team Description Paper, The 22th annual RoboCup International Symposium, Canada, 2018.
2. Linsheng Guan, Qiuna Chen, Zengcheng Shi, Wenlong Lv, Shengbing Chen, MT2021 Team Description Paper, The 24th annual RoboCup International Symposium, France, 2021.
3. BingChen, et, al. Analysis and optimization of agent 2D underlying action chain mechanism in Robocup2D project [J]. Journal of System Simulation, 2017, 29(11): 2782-2787.
4. BingChen, et, al. Mining and Verification of Attack Behavior in Robocup2D Simulation Countermeasure [J]. Journal of System Simulation, 2018, 30(12).
5. Matthew Hausknecht, Prannoy Mupparaju, Sandeep Subramanian, Shivaram Kalyanakrishnan, and Peter Stone, Half Field Offense: An Environment for Multiagent Learning and Ad Hoc Teamwork, Adaptive Learning Agents 2016.
6. Shivaram Kalyanakrishnan, Yaxin Liu, and Peter Stone, Half Field Offense in RoboCup Soccer: A Multiagent Reinforcement Learning Case Study, RoboCup International Symposium 2006.
7. Hidehisa Akiyama, Tomoharu Nakashima, HELIOS Base:An Open Source Package for the RoboCup Soccer 2D Simulation, The 17th annual RoboCup International Symposium, July 1, 2013.
8. Thomas Gabel, Philipp Klöppner, Yalcin Eren, Fabian Sommer, Steffen Breuer, Eicke Godehardt, FRA-UNIted — Team Description 2021, The 24th annual RoboCup International Symposium, France, 2021.
9. Hidehisa Akiyama, Tomoharu Nakashima, and Katsuhiro Yamashita, HELIOS2013 Team Description Paper, The 17th annual RoboCup International Symposium, Eindhoven, 2013.