ZJUNlict
Extended Team Description Paper for RoboCup 2016

Lisen Jin, Lingyun Chen, Xiaoxing Chen,
Yachun Li, Weijian Hu and Rong Xiong

National Laboratory of Industrial Control Technology
Zhejiang University
Zheda Road No.38, Hangzhou
Zhejiang Province, P.R.China
rxiong@iipc.zju.edu.cn
http://www.nlict.zju.edu.cn/ssl/WelcomePage.html

Abstract. ZJUNlict have participated in Robocup for about ten years since 2004. In this paper, we summarizes the details of ZJUNlict robot soccer system we have made in recent years. we will emphasize the main ideas of designing in the robots hardware and our software systems. Also we will share our tips on some special problems.

1 Introduction

Our team is an open project supported by the National Lab. of Industrial Control Technology in Zhejiang University, China. We have started since 2003 and participated in RoboCup 2004-2014. The competition and communication in RoboCup games benefit us a lot. In 2007-2008 RoboCup, we were one of the top four teams in the league. We also won the first place in Robocup China Open in 2006-2008 and 2011. We won the first price in 2013 and 2014, which is a great excitation to us. And we incorporate what we have done in recent years to this paper.

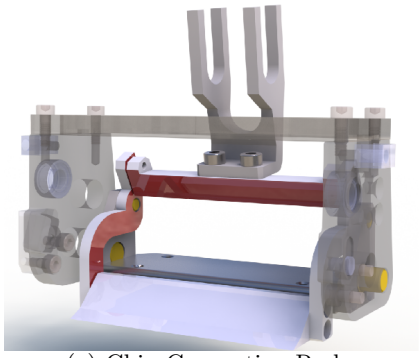
Our Team members come from several different colleges, so each member can contribute more to our project and do more efficient job.

2 Hardware

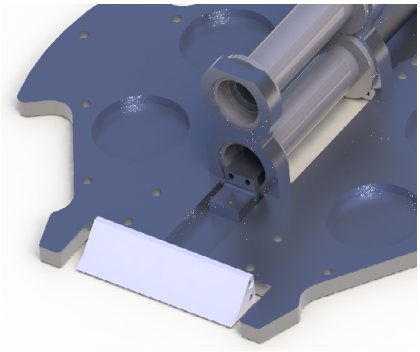
2.1 Mechanical Improvement

In last year's competition, we encountered with problems about the chip connecting rods' [Shown in Figure 1, red components] looseness, damage and deformation due to high frequent action. So we change the material with higher yield strength to replace old parts in order to prevent this.

In the meantime, we are developing new structure to directly impact on chip kicker instead using connecting rods structure. [Shown in Figure 2] We have conducted several experiments to optimize the new structure. We hope it will perform well both in chip distance ability and fatigue life.



(a) Chip Connecting Rods



(b) New Chip Structure

Fig. 1. Mchine imprоment

2.2 Fully Independent BLDC Drive Pack

Last year we successfully separated the motor control circuit, in order to improve the interchangeability between the modules. During the improvement we found that the control logic and algorithm which still were left in the center control processor is complex and inconvenient. So this year we decided to integrate the control logic and algorithm into the independent motor drive module and make it a more complete BLDC drive pack. The differences can be list as follow:

1. low-cost MCU was added to the module, only use to control a single motor.
2. line calibration interface was added to the MCU.
3. The MOSFET on PCB was changed to a smaller one, in order to reduce the size of the module.

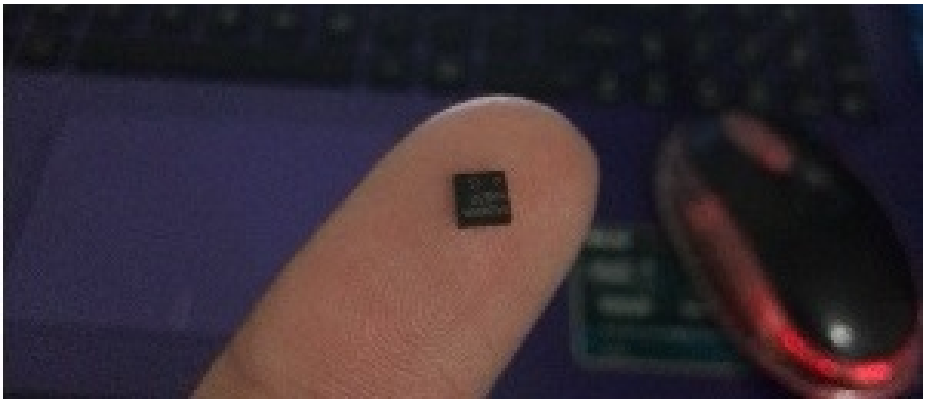


Fig. 2. The new MOSFET we selected

2.3 FPGA Improvement

In the past years, linear open-loop motion in y axis direction wasn't accurate. We want to locate the source of the problem by monitoring bottom control signal. But we almost run out of the logic elements and couldn't add a monitor module in FPGA. It also limits other electrical improvement. So we upgrade FPGA and rebuild Quartus II and NIOS II software programming environment. Also, we refactor all bottom control code to make it more expandable.

2.4 New Power Supply Subsystem

Previously, we use lithium battery to power the motor directly. This power supply approach has several disadvantages. First, as the time increase, the output voltage is not stable. Second, unstable current is bad for discharge efficiency and life of battery. For these reasons, we design a new power supply subsystem for motor. The rated power is 300W, rated current is 20A. In order to enhance the ability of moment over loading, we adopt 4-phase boost converter. When power supply subsystem output at 300W, the conversion efficiency is more than 90%, and the main peak frequency of ripple is 50MHz, based on test board. We hope we can integrate the subsystem this year.

2.5 Improvement of offline-test-mode

In order to improve the convenience of testing our robots, we modify former offline-test-mode. To be more specific, we need to test our robots to confirm they can communicate well with transmitter in certain frequency without operating the computer to send the packet. So we add offline communication tests as a subsidiary of offline-test-mode. In this mode, robots can automatically communicate with the computer and tell us the test result through the LED.

3 AI System

3.1 Strategy Hierarchical Architecture

The AI module for our off-board control system is shown in Fig. 3. It is the brain of planning strategy and coordination among robots in both attack and defense mode. The whole system is composed of world model, decision module and control module.

With the bayes-based filter evaluating the game status, the decision module selects appropriate play during the match in state of continuity well. Besides, the decision module is rebuilt in a hierarchical style. The plays focus on coordination between teammates, while the agents emphasize on planning skills for the assigned single task from the corresponding plays. The skills are vital for good ones, contributing to executing tasks with high efficiency. Both play-level and agent-level are configured with Config-files, and will be detailed in next section. The control module is composed of path planning and trajectory generation. It

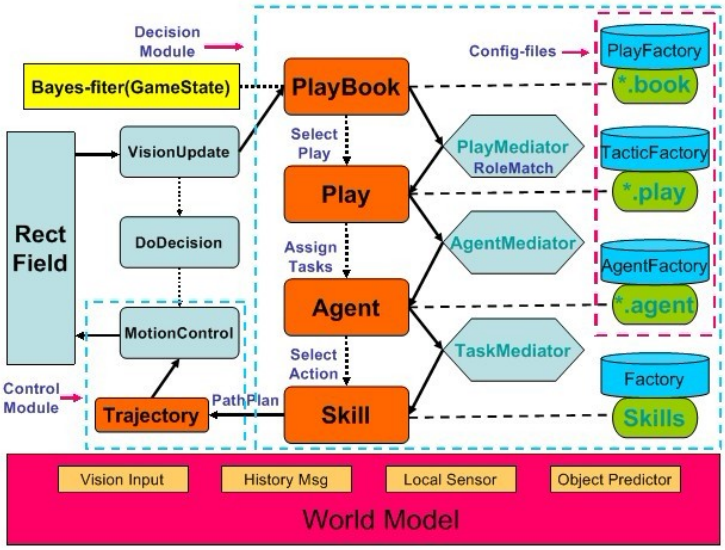


Fig. 3. Software Architecture

takes RRT algorithm to find a feasible path and Bangbang-based algorithm to solve two-boundary trajectory planning. The world model provides all the information in the match while the decision module will feedback to the predictor in world model. Thus, the close loop system adjusts all agents behavior according to the change of the environment in real time.

The Decision Module is designed in a hierarchical structure, which is consisted by several main modules as following:

1."The PlayBook Module", which selects the most appropriate play by using a Bayesian filter to evaluate the game status.

2."The Play Module", it focuses on coordination between teammates and is mainly organized by a Finite State Machine (FSM).

3."The Agent Module", it focuses on the planning skills of the single robot with the assigned tasks from the play. The Agent module selects its behaviors from the BT (Behavior Tree).

4."The Skill module", which is a direct interface of the Decision Module with the Control Module. The Skill module generates target point and selects trajectory generation method.

5."The Control Module", it is responsible for the path planning and trajectory generation. It traditionally uses the Rapidly-exploring Random Trees (RRT's) algorithm to find a feasible path and Bangbang-based algorithm to solve two-boundary trajectory planning.

6."The World Model", it provides all the information of the match. It collects all the original information from the cameras and sensor messages, then compute the massage using Kalman Filter method.

3.2 NormalPlay2016-New Generation Play

NormalPlay2016 is our first try on new generation of Plays. It deals with the normal state of our game. The normal state means the state not triggered by referee message, it is important for RoboCup-SSL game since the field is larger than before and the technique of teams are all improving, thus we developed NormalPlay2016. When developing a Play based on FSM, we used to consider all the possible states (or as much as we could) and conditions to enter or exit these states. But for our new generation play, we no longer develop in this way. By using learning module and state evaluation module based on Bayesian theory, NormalPlay2016 can itself generates all the possible routes and assign all the tasks automatically. To be more concise, what old plays doing was choosing, but new plays doing is thinking.

A brief working process of NormalPlay2016 is:

1. Evaluation of the state of the game and assign the ration of attack-defend. For example, when there is little threatening and the possibility of successful shooting of our team is big, the ration would be 5-0 or 4-1 (considering one robot always for goalie, thus in a-b, the sum of a and b is the total robots number of our team minus one).
2. Generating all the attack routes and defend routes.
3. Evaluation the possibility of successful attacking routes and successful defending routes based on Bayes theory, and choose the optimal route.
4. Evaluation the optimal route and decompose it to Skills of each robots, then assigning all these Skills to robots.

NormalPlay2016 is still in the test phase, we are working hard on develop this new play. And for more details, our lab will publish papers on this topic after we finish it on our website.

3.3 Defense Strategy in AI System

In this section, we mainly introduce our defense which is a vital part for competition and a superiority of our team. Our defense is called Close-Marking Defense. We get the information about all opponent robots and calculate feature value. According to the attribute value we will match the role for every opponent robot, such as leader, passer, etc. Then attack array is set up to describe the robot in order and design defense strategy at last. Feature Value We calculates all feature values for every opponent robot according to vision messages. Feature a value, which is used to estimate the threatening level, reflects on the state of opponent robots. We have more than a dozen of feature; some are simple value which can be calculated by vision message according to some obvious geometrical

relationship, just like the distance between robot and ball, the distance between opponent robot and our goal, shoot angle and so on. The other are some complex feature value which can be calculated by simple feature value, just like Touch Ball value (the ability to receive ball and shoot), Chase Ball value (the ability to chase all and shoot), Pass Ball value (the ability to chase and shoot). For example, Touch Ball value of an opponent robot depends on the shoot angle of the opponent robot, whether other robots block in the pass line, the distance between the robot and our goal. All These features will be used in the next step opponent robots role matching.

Opponent Robots Role Match Different roles have different priorities. A default role group includes:

- **Receiver** is an offensive role which receives the ball and finishes shooting.
- **Leader** is an offensive role which controls the ball.
- **Attacker** is an offensive role which is always ready for the attack.
- **Defender** is a defensive role which takes part in defense strategy.
- **Goalie** is the goalie.

The role matched degree is calculated according to the features of robot. For example, when we judge whether the robot is a receiver, we will use three feature values: Touch Value (the ability to receive ball and shoot), Chase Value (the ability to chase all and shoot), Receive Angle (the angle between ball receiving and ball shooting). Matched degree will be calculated for each robot in priority decreasing order.

Attack Array Attack array is a list of opponent robots. In attack array, we generate the attack array according to the match result. Now every opponent robot has their role and the matched degree for this role. We can compare the role priority and matched degree to set up attack array. But we don't need to mark for certain roles, such as goalie, opponent blocker in kick off area, which should be ignored.

Design Defend Script We design our defense script in the form of a Finite State Machine. The task is assigned in a state and can receive a parameter which is a number representing the order in attack array. So the robot will defend the corresponding opponent in attack array. There is an example to demonstrate our defense. Fig.2 shows the task assignment in the defense script. Defend Kick is a task defending the opponent robot which takes the free kick. We can see marking task receiving a parameter which is just an order in the attack array. A simulation is shown as Fig.4. Yellow team is the attack side. There are five attackers, four are of attacker role, and one is of leader role. Their defense superiority order is 4-3-1-5-2.

script.png

```
["DefendState"] = {
        Leader   = task.defendKick(),
        Special  = task.marking("First"),
        Middle   = task.marking("Second"),
        Defender = task.marking("Third"),
        Assister = task.marking("Fourth"),
        Goalie   = task.goalie(),
}
```

(a) a part of defense script

simulation.png

(b) defense simulation

Fig. 4. Mchine imprement

3.4 Parameters Feedback Adjust

In the previous sections, we have introduced that our system is based on Play-Skill frame. The Play decides what the team should do and the Skill decides what the specific player should do, that is, the aim of a Play is to make the system intelligent and the aim of Skill is to make the robots movement more precise.

Inside the cpp of a Skill there are many parameters. And these parameters may sometimes dramatically influence the performance of the whole system. During our previous years development of our system, we used to adjust all the parameters by ourselves before games. However the problem is the job is both tiring and time consuming. Furthermore, since the physical environment is different between different fields, even though our robots work out fine in our own field, in other fields it may not work the same way.

Facing all the problems above, we developed a parameters feedback adjust module. It is just like what we learnt from the principle of automatic control. The system will detect the deviation from expected value to real value, and use the deviation to adjust the parameters. We have already applied this module into several Skills for test, and it turned out to be useful. In the future, we will apply this module into our whole system and we do believe this module can improve the performance of our whole system.

4 Conclusion

Owing to all team members' hard work, we've made some improvements in our system, both in hardware and software. If the above information is useful to

some new participating teams, or can contribute to the small size league community, we will be very honor. We are also looking forward to share experiences with other great teams around the world.

References

1. Brett Browning, James Bruce, Michael Bowling and Manuela Veloso, STP: Skills, tactics and plays for multi-robot control in adversarial environments
2. Yonghai Wu, Penghui Yin and Rong Xiong, ZJUNlict Team Description Paper for RoboCup2011
3. Yonghai Wu, Xingzhong Qiu, Guo Yu, Jianjun Chen and Xuqing Rie: Extended TDP of ZjuNlict 2009 *Robocup 2009*
4. *Sebastian Thrun, Wolfram Burgard, Dieter Fox, Probabilistic Robotics, The MIT Press*