ZJUNlict
Extended TDP for RoboCup 2013

Yonghai Wu, Yue Zhao, Yifan Shen, Chuan Li,
Li Fang, Hangjun Tong and Rong Xiong

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

Abstract. ZJUNlict have participated in Robocup for about nine 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 new 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 2004-2012. 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. Last year, we won the second price in Mexico City, which is a great excitation to us. And we incorporate what we have done in recent years to this paper.

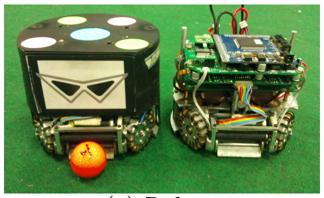
Our Team members come from serveral different colleges, so each member can contribute more to our project and do more efficient job.

Team Leader Yonghai Wu (AI)
Team Member
- AI: Yue Zhao, Qun Wang, Hangjun Tong, Xiaohe Dai
- Electronic: Yifan Shen, Chuan Li, Li Fnag

2 Hardware Architecture

2.1 Robot

Components of the Robot Our robots are equipped with 4 omni-directional wheels. Each is driven by a 30 watt brushless Maxon motors which help our robot run with about $3.0m/s$ and $6.0m/s^2$. The reduction ratio of the gearbox


(a) Robot

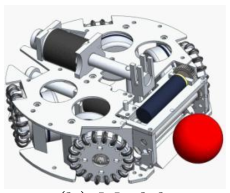

(b) Model

Fig. 1. Our Robots

with internal spur gear is 4:1. Besides there are three major machinery devices: a dribbling device, a shooting device and a chipping device. We redisigned the omni-wheels to reduce the friction between the small passive wheels and the driving wheel so that our robot's movement is smoother. We spend a lot of time testing the dribber materail in order to choose a satisfactory one. The robot is shown in Figure 1(b).

Our circuit architecture is FPGA based all-in-one solution as the central processor module. Our motor driving part is a stable module based on MC33035, which has been developed to complete the four years since 2007 in Atlanta.

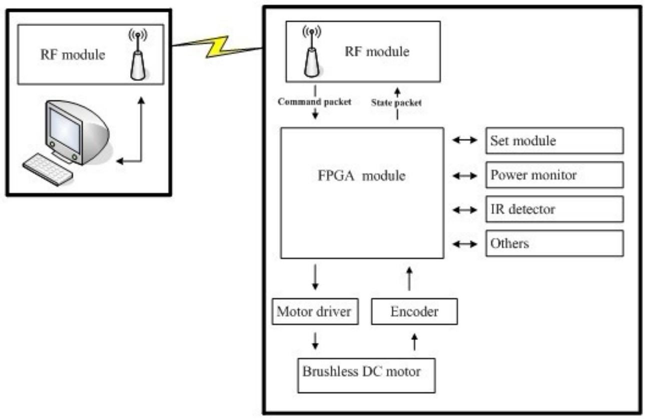
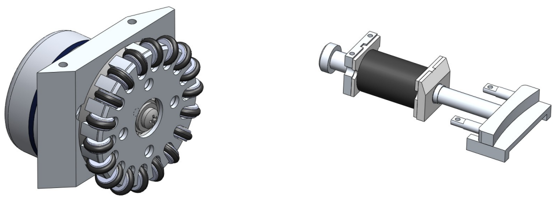


Fig. 2. Schematic Diagram

There is an encoder module to form a local feedback control loop. A commercial wireless module based on nRF2401 is used on our robot. Meanwhile, we develop a new communication system between the PC and robots. We choose two smaller capacitors with higher voltage, to achieve a better result. They will help us save more power and permit several shots in short intervals. In addition, we have set module, power monitor module, IR detector module, and so on, in order to complete the functions of our robots. We use Protel Altium Designer 6.0 to layout the PCB board.

Mechanical Design Omni-directional Wheel Design Each wheel is composed of a big aluminous wheel with 18 grooves distributed equably in Circle, there is a small wheel founded with PM in every groove. Similarly, there is a groove in the small wheel, covered by a o-ring. The omni-directional wheel is shown in Figure 3(a) In RoboCup 2008, the wheel exhibited some problems, such as large gap and friction. Now the wheel is redesigned in some details, and receives a perfect performance. Except the wheels, we do not change a lot in the other parts. To improve the manufacture precision, we make all the parts with CNC machine tools.


(a) Omni-directional wheel (b) Flat-kick device

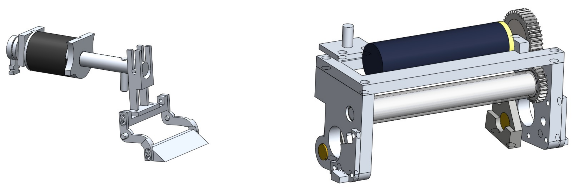

(c) Chip-kick device (d) Dribbling device

Fig. 3. Mechanical Parts

Shooting Device Design The robot's shooting device is the primary method of both scoring and passing. It is made up of a an electromagnet and a simple mechanical structure. The electromagnet is made by ourselves and is calculated accurately in advance. It is drive by two big capacitors which is fixed on the floor board above. Since RoboCup 2007, we are pleased with our shooting devices and don't change a lot. It is shown in Figure 3(b). The shooting device can give the ball a maximum velocity of 10m/s. In the match, it is controlled by the circuit, the time and the force of kicking the ball is also in charge. This part of the robot is usually cooperate with others, such as the chipper,the dribbler.

Chipping Device Design The chipping device allows the robots to pass the opponent by kicking the ball into the air. As same as the shooting devices, it is also drive by two capacitors, the method to control it is also the same. It is shown in Figure 3(c). When the chipping device works, the shovel close to the ground can chip the ball to a maximum height of 0.8m and a maximum length of 3.2m. The angle of the shovel and the height between the chipping pole and the ground influence the performance most. When the ball falls to the ground, it is a litter difficult for the partner to get the ball steadily and quickly. Because of the elasticity of the ball and ground both play important parts.

Dribbling Device Design The dribbling device is the assembly that controls the ball. It is designed to stop a ball, control it and prevent losing it. The dribbling device is drive by a motor, accelerated by a pair of gears. A stick swathed with a special pipe circum gyrates when the ball comes close to the robot or it must compete for the ball with the opponent. From Robocup 2006 at Bremen, we found that the ball controlling ability is not very satisfactory, these years, we have tried many ways to improve it. We get a lot of experiences, receive a much better result now. It is shown in Figure 3(b). The higher rotate speed the motor circum gyrate, the bigger force the ball is given, but on the other hand, the ball will be easier to lose control. To get a better effect, we redesigned the limit device, besides we spend a lot of time choosing the material.

Electronic Design

Driving Peripheral Design There is a local feedback control loop on the robot. Motor driving module based on a self constructed verilog module which consists of a rotor position decoder for proper commutation sequencing, temperature compensated reference capable of supplying sensor power, frequency programmable saw tooth oscillator, three open collector top drivers, and three high current totem pole bottom drivers ideally suited for driving power MOSFETs, Can Efficiently Control Brush DC Motors with External MOSFET H-Bridge.(Figure 4)

Communication Peripheral Design Our communication system is based on the module of NRF2401. NRF2401 is a single-chip radio transceiver for the world wide 2.4 - 2.5 GHz ISM band. The transceiver consists of a fully integrated frequency synthesizer, a power amplifier, a crystal oscillator and a modulator. Output power and frequency channels are easily programmable by use of the 3-wire serial interface. This year, a new communication system has been developed to reach better results. The new communication system is based on two RF modules on the robot. They were two separate modules, respectively charging for receiving and sending. Compared to last year, data transferred from the pc to robot, is encoded to a larger packet. All of the robot on the ground can receives the same packet at the same time, and pick up the right information with the numbers of themselves. the right information picked up from the large packet includes the speed of every

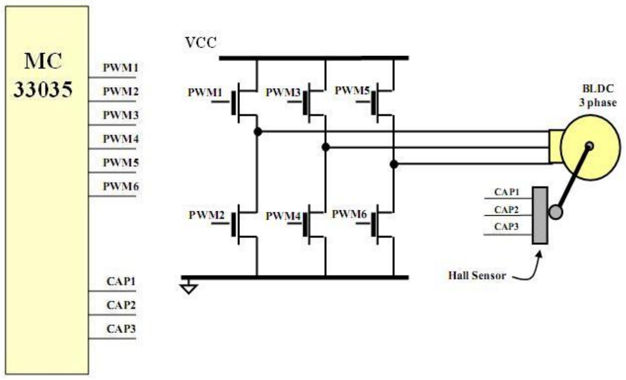


Fig. 4. Motor Drive Diagram

wheels, shoot or chip command, the number of robot. Thus, every robot can receive the order without delay. Meanwhile the robots can send a packet to the pc. However, there is only one robot can send packet that contains useful information to pc. The PC can receive information from the robot without any delays. So the command sent from the pc can be implemented more effectively and timely.

Shooting Peripheral Design Compared to previous years, our Shooting system (Figure 5)is not changed too much this year. Our boosted circuit includes a PWM control module and voltage control module, to control the boost capacitor voltage. FPGA sent chip or flat shoot signal, through control circuit for chip or flat shoot. However, we choose two smaller capacitors with higher voltage, to achieve a better result. The kicks are driven by two $4700uF$ capacitors charged to $190V$. The kicker can give the ball a maximum velocity of $10m/s$. With the increased voltage, the velocity will be faster.

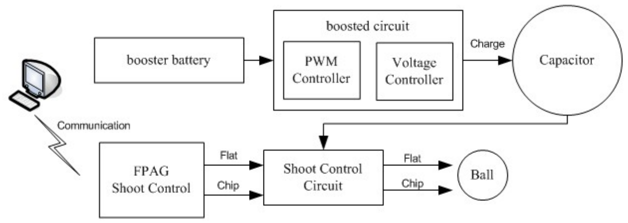


Fig. 5. Shoot Diagram

3 Emebedded Software

Since RoboCup 2007 in Atlant, Our circuit architecture use the NiosII as the central processor module which is a soft IP provided by Altera company matching at QuartusII9.1 and NiosII9.1 software programming environment. The overview of our embedded software flow is show in Figure 6.

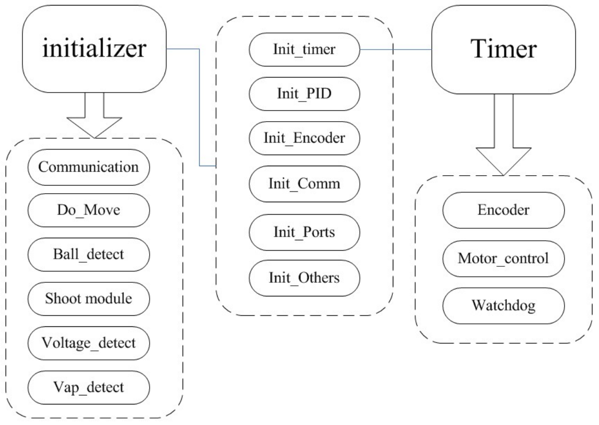


Fig. 6. Emebedded Software Flowchart

3.1 Motor PID Control

About the encoder module, we use a 512 gratings encoder with a decoder constructed by Verilog. It will count how many gratings it has detected in 2 ms, and then translated to the angular velocity of the motor, while it determine the direction of the velocity by phase difference between channel A and B which are quadrature signals shifted by 90. Based on the velocity we detect and the set velocity we desire, we can caculate the expected PWM duty with PI algorithm and the send it to the motor control module.

Based on our motor driver module (Figure 7), we use an incremental PID algorithm, real-time code set encoding to read the speed of motor for PI and PID control, so that motor speed can reach the stable speed settings. About the encoder module, we use AM512, which is a compact solution for angular position sensing. The IC senses the angular position of a permanent magnet placed above the chip. So we put the permanent magnet on the motor, when the motor rotation, FPGA board will receive the Incremental signal from the encoder module. There are two signals for incremental output: channel A and channel B. Signals A and B are quadrature signals, shifted by 90. The speed of the motor can be count by the signals.

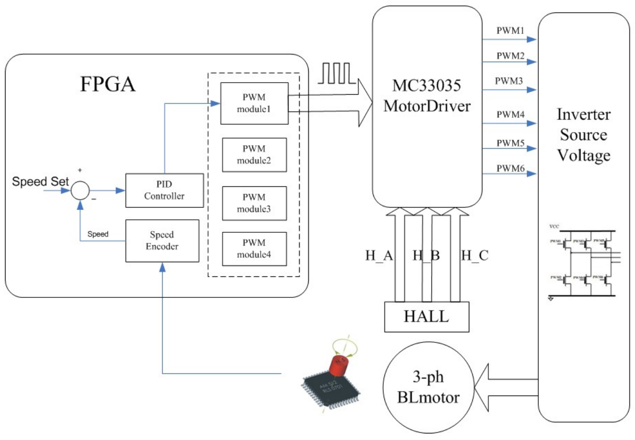


Fig. 7. Motor Control Diagram

3.2 Uploading function

This season, we add uploading function when the referee box state is STOP during the normal game. Although in normal situations, robots will not upload messages until the game state (such as ball in the mouth) is changed. However, this time, robots will have the function of uploading information about their own states. The realization of function is divided into two parts, upper computer asking and computer answer and specified as follow: When the referee box is clicked to send a STOP message, the upper computer will set a flag bit in the special position in the data packet sent to the robot, which is called upper computer asking and is totally different from our normal communication protocols. And if the robot finds the flag bit match its own car number, the robot will go into a different state of decoding data packets. They will check some of its status including whether the infra is good or bad, the value of battery voltage and the value of capacitance voltage through some hardware characteristic. For example, it checks infra state through reading electrical level state of one pin of FPGA for several times. After finishing checking, these data will be sent to upper computer. This procedure is called robot answer. This way, we will avoid many troubles to check states of robots. Firstly, we do not need to call for a suspension to check the robot of every state. Secondly, we can get the information of all robots in the same time instead of checking them one by one.

However, when checking states, we must ensure that our normal competition is not disturbed. Therefore, for the reason that the "STOP" time is short, we must limit the times of upper computer asking. Upper computer asking is as bellows: Firstly, upper computer will set that flag bit in different special position one by one in terms of robot number to make sure that all six robots will be asked. The cycle of upper computer asking is 16ms, and upper computer will ask each robot at most 10 times. Secondly, if asking one robot for 10 times without answer, upper computer will give up and ask for the next robot. After lots of tests, we find that the whole procedure can be finished in 0.5s and the percentage of getting all six robots' information can reach over 90

Using this function, when the pause of the match, we don't need to go to the playing field to check whether our robots are good or not. It's much more convenient.

3.3 LabVIEW real-time detection for robot motion

As we know, a good motion control is based on the precise and rapid data feedback. Because of the limit of the detection method, we cannot get the signal we want freely. Besides, some of the signal is parameter of the program which can never be detected by the instrument like oscilloscope. This LabVIEW mode is set up for detecting the real time situation about robot in motion, involved those parameters of each wheel, the measurement result from gyroscope, etc. Furthermore, we would find the problems and indiscoverable errors of our robot in this way. We can test motion performance by Analyzing the uploading data without using heavy instrument. That is really significance for maintenance and improvement the robot. The message sending is also supported in this mode, which means we can send Specific parameters which can be changed in real-time to test the response of the robot. It means we can adjust the parameters when the robot is working, to ensure which parameter is best for the robot, and check the result in real-time.

3.4 Movement PI Control Depending On Gyroscope

In order to make the robots' motion faster.We optimize the movenent performance depending on SD-788 gyroscope(Figure 8).

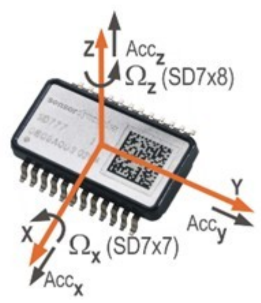


Fig. 8. SD-788 Gyroscope

We collect the uploaded data delivered by gyroscope. Then we filter the data in corresponding verilog module. The filtered data is used by the movement PI control module. This module will read the rate on Z axis and turn itself into the fit arithmetic-branch which is designed for different rate stages. To take the gyroscope's temperature drift into consideration, we revise the temperature coefficient using LabView and put the coefficient into the measure-decode function.

4 AI System

4.1 Strategy Hierarchical Architecture

The AI module for our off-board control system is shown in Fig. 9. 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.

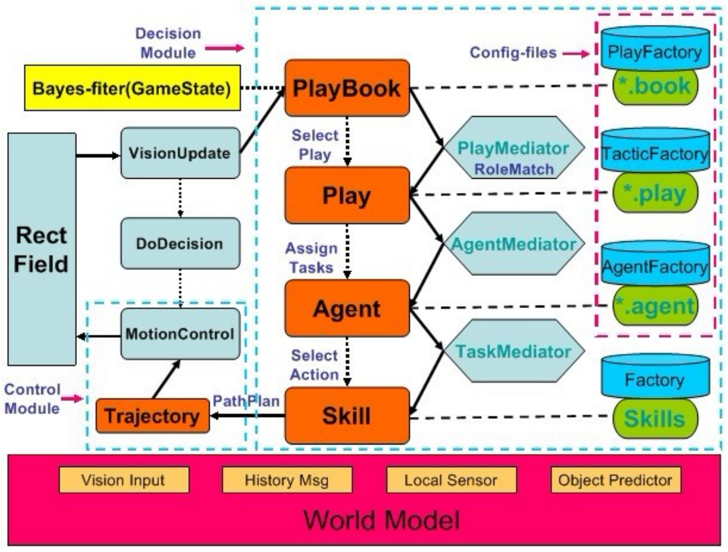


Fig. 9. Software Architecture

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 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. More details are shown in ZjuNlict2012TDP [2].

Game State Evaluation A right evaluation of game status plays an important role in the match. For the complexity of game situation, it's really a troublesome work. We consider a comprehensive view of the observation information, the strength of the rivals as well as the strategy we take to realize the evaluation. Our new method is based on the Bayes Theory [6], which gives a creative combination of the observation information and the historical strategy feedback.

```
Algorithm Bayes filter p($x_k$,$u_k$,$z_k$)
for all $x_k$ do
    $\overline{p}(x_k) = \sum_{x_k} p(x_k|u_k, x_{k-1})p(x_{k-1})$
    $p(x_k) = \eta p(z_k|x_k,)\overline{p}(x_k)$
end for
```

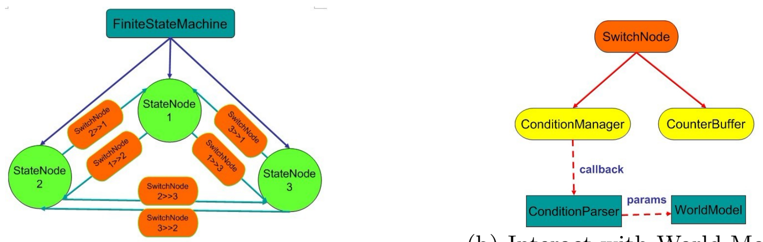
Fig. 10. General Algorithm for Bayes-filter

Fig. 10 depicts the basic Bayes Filter Algorithm in pseudo code form.
our current method owns two main features:

- **More stable:** The evaluator gives more appropriate analysis of game state, even though there come momentary errors in observation information, and helps to reduce the perturbation of the strategy while strengthens the continuity of the strategy.
- **Well targeted:** Variable initial values of prior probabilities $p(x_k|u_k, x_{k-1})$ in the algorithm accustomed to different characteristic teams, making it more convenient to configure the attacking and defending strategy in a more flexible way.

Finite State Machine This year, we mainly focus on script configuration to control robot behavior by a unified FSM-based mechanism, as is seen in Fig. 11(a). Every state node will link with the world model by using a condition parser contained in the connected switch node. The switch conditions are written in the external scripts and could be queried through internal world model.

Play Our AI module is implemented using a play-based approach. Each play represents a fixed team plan, in which each team group has a collaborating mission to perform and that group may have variable agents to execute. We can also consider the play a coach in the soccer game. The plays can transfer to each other, and the group for each agent can change too. As mentioned above,



(a) Finite State Machine (b) Interact with World Model

Fig. 11. FSM-based Mechanism

plays are designed and implemented with the unified FSM module. Plays are composed of many parts, such as applicable condition, evaluating score, roles, finite state machine and role tasks similar to CMU's STP [3]. These parts can all be configured by external scripts.

Agent Agent is performed as robot behavior which is assigned by play to control robot to perform a specific action such as manipulating ball to zone, passing the ball to a teammate, scrambling ball from opponent, getting ball. The agent first select a best proper team member according to the assigned zone and task which receive form play, then selects a proper skill for the team member performing the assigned behavior in every execution cycle and finally generates the best target for robot action. For example, in the shooting task it will be the best shoot point on opponent goal line.

Skill Skill is a set of basic knowledge for every agent, such as how to move to a point, how to get the ball and kick. Some skill generates a next target point of the specific path which will be passed to the navigation module for finally generating a avoiding collision path. Some skill will direct generate the speed trajectory for some special behavior such as pulling the ball from a opponent front. Each of skills has different main idea of generating path for robot, intercept the ball skill is different from move to point skill in many ways. We can test each skill independently for the best performance for each of skills.

4.2 New Strategy Frame with Lua

In last year, we wrote many scripts with our script system, which is actually a convenient system configured with many so-called "scripts". However, these scripts can not be compiled. If the strategy coder make some mistakes in the script, it will be a difficult work to debug.

In order to overcome this shortcoming, we develop a more flexible and robust strategy frame with Lua. By this way, we make the "script" to be a real script. We have moved some simple and repeated work to Lua, and left the complicated algorithms such as path planning, vision handling in C++. So the code is divided to two parts just like Fig. 12.Although we change our configuration file to real Lua script, our whole strategy is also build on FSM.

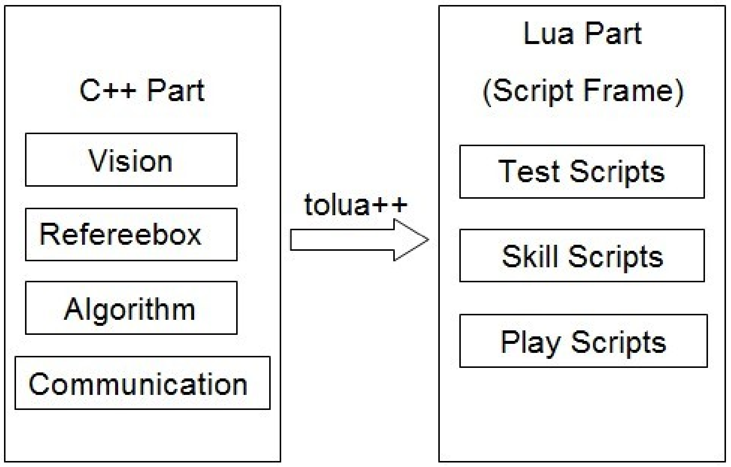


Fig. 12. New Strategy Frame with Lua

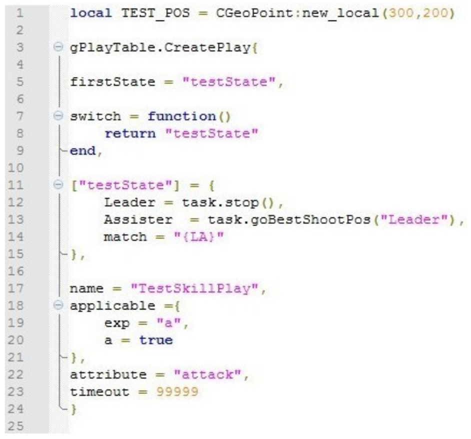
Tolua++ In Fig. 12, tolua++ is an extended version of tolua, which is a tool to integrate C/C++ code with Lua. In Lua part, we should know some variables, functions and methods which is write in C++, and tolua++ help us deal with the work efficiently by using a package file, which is a C++ cleaned header file. For more details about tolua++, refer to its Reference Manual[7].

A Simple Play Script Fig. 13 is a simple play script using Lua. This script has just one state, in this state, the role Leader will execute the STOP skill, and the role Assister will goto the best shoot pos which is calculate using Leader's pos. The "match" item means that we should choose L(short for "Leader") first, then A(short for "Assister"). The "timeout" item means that the play will at most execute 99999 frames.

A Lua script make configure and debug more efficiently, and another by-product is that we can modifying the Lua code when running C++ project, and we could see the result without restart our program.

5 Conclusion

Owing to our all team member hard work, we can obtain this result. 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.



```lua
local TEST_POS = CGeoPoint:new_local(300,200)

gPlayTable.CreatePlay{

firstState = "testState",

switch = function()
    return "testState"
end,

["testState"] = {
    Leader = task.stop(),
    Assister  = task.goBestShootPos("Leader"),
    match = "{LA}"
},

name = "TestSkillPlay",
applicable ={
    exp = "a",
    a = true
},
attribute = "attack",
timeout = 99999
}

```

Fig. 13. A Simple Play Script

References

1. Yonghai Wu, Yue Zhao and Rong Xiong, ZJUNlict Team Description Paper for RoboCup2013
2. Yonghai Wu, Penghui Yin, Yue Zhao and Yichao Mao, Rong Xiong, ZJUNlict Team Description Paper for RoboCup2012
3. Brett Browning, James Bruce, Michael Bowling and Manuela Veloso, STP: Skills, tactics and plays for multi-robot control in adversarial environments
4. Yonghai Wu, Xingzhong Qiu, Guo Yu, Jianjun Chen and Xuqing Rie: Extended TDP of ZjuNlict 2009 *Robocup 2009*
5. *Sheng Yu, Yonghai Wu. Motion prediction in a high-speed, dynamic environment.*
6. *Sebastian Thrun, Wolfram Burgard, Dieter Fox, Probabilistic Robotics, The MIT Press*
7. *tolua++ Reference Manual. http://www.codenix.com/ tolua/tolua++.html.*