TEAM DESCRIPTION PAPER & RESEARCH REPORT 2018

TJArk, Robotics & Artificial Intelligence Laboratory

Tongji University, Shanghai 201804, P.R.China

(TJArk.official@gmail.com)

Jan. 15, 2019

CONTENT

Team Description Paper & Research Report 2018

TJArk is a Chinese RoboCup team from Tongji University. We took part in RoboCup in 2006 for the first time and obtained the third place in RoboCup 2018. All the members of the team are from Control Science and Control Engineering Department of Tongji University, China. This paper will provide a concise description the team, including some basic information about the research interests of its team members and the improvements on each part of TJArk's project. Below we will talk in detail about the notable work we did during 2018 and briefly introduce what advancements we are currently pursuing.

1 Team Information

1.1 About Team

Team Name:	TJArk	
Team Leaders:	Prof. Liu. Chengju	College of Electronic and information Engineering
	Prof. Chen. Qijun	College of Electronic and information Engineering
	Mr. Li. Shu	Control Science and Control Engineering
	Mr. Zeng. Zhiying	Control Science and Control Engineering
	Mr. Zhou. Ziqiang	Control Science and Control Engineering
	Mr. Tang. Liang	Control Science and Control Engineering
Affiliation:	Robotics & Artificial Intelligence Lab, Tongji University	
Location:	Shanghai, China	

1.2 Team Members

The main researchers and programmers of our teams are students. Below show the basic information about this students researchers:

Tang Liang, Control Science and Control Engineering, M.S Student.

Chen Hao, Control Science and Control Engineering, M.S Student.

Zhou Haoran, Control Science and Control Engineering, M.S Student.

Shi Wenbo, Control Science and Control Engineering, D.S Student.

Xu Zihan, Control Science and Control Engineering, D.S Student.

Zhang Xue, Control Science and Control Engineering, M.S Student.

Sun Haoran, Control Science and Control Engineering, M.S Student.

Yan Qingqing, Control Science and Control Engineering, M.S Student.

Xiao Hongyuan, Control Science and Control Engineering, B.S Student.

Guo Xiang, Control Science and Control Engineering, B.S Student.

Huang Zhengang, Control Science and Control Engineering, B.S Student.

Zhang Hao, Control Science and Control Engineering, B.S Student.

Figure 1.1 The majority of team TJArk members in RoboCup 2018

1.3 About Robot

Our team used the V5 Nao robots in RoboCup2018, and the next year we are ready for trying using the newer version of NAO, V6.

2 Introduction

2.1 Code Usage

TJArk has been participating in RoboCup SPL since 2006, and since that we had kept our own code base. However, since 2013, we have been using the B-Human framework of Code Release 2013 including the kick engine, debugging tools and CABSL, according to the license, to develop our own codes. Anyway, we are very grateful for the contribution of B-Human for SPL.

Besides, we would also like to thank UNSW Australia for their walking engine. We have been trying to develop our own walking engine independently, but the walking effect is not ideal and is still needed to debug. So our walking engine using in RoboCup 2018 was still based on the walking of rUNSWift-2014-release. But we have made some big improvement on that by ourselves, which make robots walk in line with our vision. So let us express our heartfelt thanks to their devotions once again.

However, others modules are mainly developed from our own framework. Although some of them work not very excellent, we are still using them and constantly developing them. The modules mainly original developed by our TJArk are listed below：

1. Vision System
2. Localization System
3. Locomotion System
4. Whistle System
5. Path planning module
6. Decision System

During 2018, we have made a lot of attempts and improvements for the above modules. We will introduce them in the following chapters. Our main focus are the new works and the differences compared to the past.

2.2 Code Release and Research Report

Our code release accompanying this report and the according documentation can be found under the following links:

Documentation:

https://github.com/TJArk-Robotics/TJArK-Vision/TJArkTeamResearchReport2018.pdf

Code:

https://github.com/TJArk-Robotics/TJArK-Vision

2.3 Chapter Arrangement

As show above, the first chapter introduces the basic information of our team TJArk, and the second chapter (current chapter) mainly illustrates the code usage and main content of the document.

The following chapters will introduce the works and contributions we have done during 2018, and the main arrangements are listed as follows:

The third chapter will describe some of our vision system, including field color detector, field border detector, features detector, ball detector and background perception. Besides, we also develop a deep learning method for objects recognition, which try to classify different objects on SPL field by once, such as opponent, ball and goal, which shows the results is not bad.

The fourth chapter will illustrate our localization system. That would simply describe what methods we used to achieve the robots' localization.

Then the fifth chapter will introduce the related works on the locomotion system. In this part, we give a brief description about what changes we have done. That mainly contain two aspects, one is the walking and another is the kicking actions. Those developments are all made to make the motion system more convenient and flexible.

The sixth chapter is the whistle module. Actually, it is similar to our module in 2017. It will show the main procedure of how to identify the whistle.

The seventh chapter is the path planning module. That would mainly introduce the improvements compared to the 2017. For path planning, we still use the RRT algorithm, but we have made many improvements to the details, which promote a smoother path for the robots.

Next chapter, e.g. the eighth chapter, will introduce a brand new strategy for the decision system. In the past and even now, we usually try to set the special behavior for robots for each possible situation on the field. But this is complicated and cumbersome. So we are trying to develop the reinforcement learning (RL) method, which could make robots choose the behavior action automatically according the real-time scene. Although we haven't applied the methods to the real robots for competition, because the training for real robots is difficult and the effect on real robots is also not well now, we have achieved the simple algorithm about this and have done some testing, which prove the feasibility of this method for decision system. And this part would mainly describe the usage of reinforcement learning. We believe though our constant efforts, the developed methods would help us greatly simplify the code and make decision more accurate. Besides, we should know that the RL have a much bigger application space, not only for the upper decision system.

The last, we would show some paper that were published or written by our laboratory in recent years.

3 Vision System

These years we have developed a fully calibration free vision system. It is capable to cope with light changing conditions and save a lot of time during debug process. More important, it is fast enough for NAO robot to process images at real-time speed.

In this part, we will simply describe some sub-modules in our vision firstly, such as field color detector, field border detector, features detector, which are developed on our 2017 Team Research Report, if you want to read the previous part for details, you can turn to the 2017[1]. Then we will describe a new method for objects recognition parts by deep learning.

3.1 Field color detector based on GMM

The algorithm and detailed procedure described in this part has been published in 2017 IEEE Conference on Real-time Computing and Robotics.

The most important key point in a fully calibration vision system is to build an auto field color detector. In our current vision system, we are mainly using Gaussian Mixture Model (GMM) to segment every frame of images. Here are some results using the methods.

a) Normal situation:

b) Severe uneven light:

In addition to the above experiments, we also test in other complex environments (dark and uneven light, small field, etc.). All show a great result.

The main processing steps are described below:

1) Convert every frame of image to HSV color Space.
2) Find histogram of S channel.
3) Using GMM algorithm fit the histogram.
4) For green color, S-channel should be greater than 100, so choose the wave with a peak greater than 100 as potential green seeds.
5) For potential green seeds, fit its H-channel with GMM.
6) Choose the wave with the highest peak as green seeds, and we get green threshold in HSV color space.

3.2 Field border detector

Field border is important in perception. Our implementation of field border detection is based on RANSAC. Here are some of the results:

Figure 3.1 Results of field border detection

3.3 Line detection and center circle detector

The implementation of line and center circle detection is also based on RANSAC. After scan lines prepared, we mark non-green regions and try to fit a line or circle by RANSAC.

Figure 3.2 Results of Line and center circle detection

3.4 Ball detector

The BallPerceptor includes two function mainly: fitball() and classifyBalls2(). The fitball() function finds 24 edge points based on the ballspots and try to fit a circle using those edge points. If success, it saves the ballspot as a possible ball. Then all the possible balls are transferred to the second function called classifyBalls2(). classifyBalls2() is a classifier which uses some criteria to decide whether the possible ball is a valid ball. This classifier also assigns a score to those possible ball that meet the criteria, so that we can choose the most possible ball from them. These two function will be detailed as follow:

1) fitball()

fitball() is in charge of finding the edge points of a ballspot and trying to fit a circle based on those edge points. If success, it considers the ballspot as a possible ball. This function consists two steps.

a) First step

Choose three points called guesspoint inside the circle. These three guesspoints should not be on a same line, so that the edge points found will not repeat. Then using these three guesspoints as start points to trace from the guesspoint to the region's extrema. There are eight scan lines and they will finish when finding enough green pixels. The scan lines' scanning directions and guess points are shown by the following Fig. 3.3:

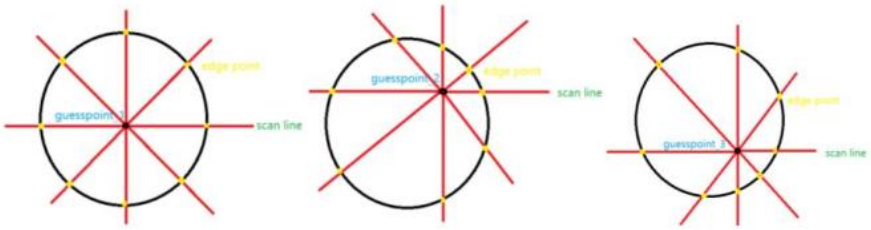
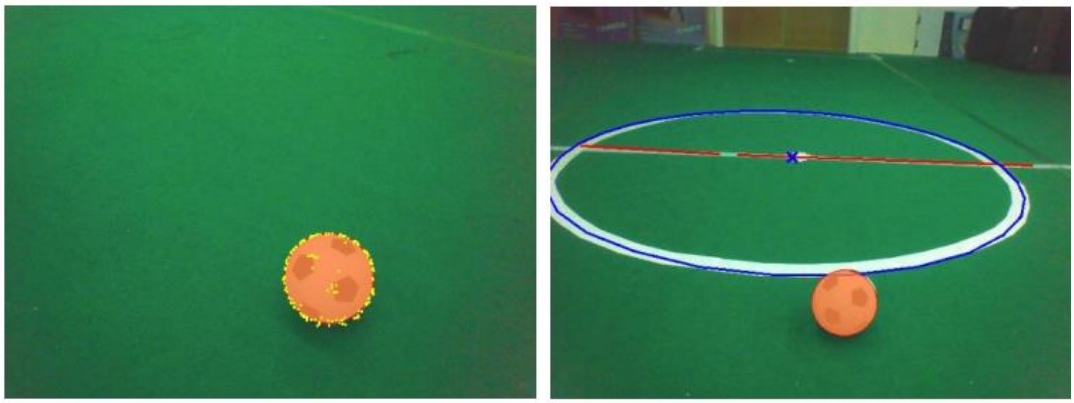


Figure 3.3 Search for guess points (from left to right: point 1, point 2, point 3)

The edges found is as Fig. 3.4 (a):



(a) Edges points (yellow points) (b) Ball detection

Figure 3.4 Results of ball detector

b) Second step

After obtaining 24 edge points of ball, we use RANSAC algorithm to fit them to a circle. The reason that we use RANSAC instead of the least squares is that the least squares method can be influenced by noise easily, while the former always could get a better performance.

2) classifyBalls2()

This function is in charge of deciding whether the possible ball found by fitball() is valid. If that's valid, save that as a candidate. Then in the end, we will choose a most possible ball as ballPercept among those candidates according to their scores.

In the beginning, it uses OTSU algorithm to find a suitable threshold so as to distinguish the black pixels and white pixels inside the possible ball. With this algorithm we can distinguish correctly in spite of the frequently changing of lighting condition. Then this function goes through all the pixels inside the square whose center is possible ball's center and side length is the possible ball's diameter. After that, it can get several statistical data to decide whether it is a valid ball or not. The statistical data includes:

a) **ratioGreen**: The ratio of green pixels which lie in the square and stay outside the circle among the totality of pixels outside the circle. It is only convinced to be a ball on condition that the ratio is larger than a certain threshold. And if a valid ball is confirmed, there must be a certain quantity of green pixels around since the ball is definitely on the ground.
b) **ratioTotal**: The ratio of black or white pixels among the totality of pixels in the circle.
c) **vary**: The variance of gray scale provided by the black and white pixels in the circle. This parameter protects the detected possibleball from being wrongly confirmed as a ball.
d) **whitePercent**: The proportion of white pixels in the circle.
e) **ratio**: the ratio between black and white pixels in the circle.
f) **meanWhite**: the average gray scale of white pixels in the circle.
g) **meanBlack**: the average gray scale of black pixels in the circle

According the statistical date, we will define a formula to calculate the **score**.

Then for a ball, if all the 7 indexes above meet the requirements and the score exceeds the threshold, that will be confirmed a valid ball. However, if more than one ball is confirmed, we'll take the one with the highest score as the convincing one. The performance of this module show as Fig.3.4 (b).

3.5 Detecting objects with convolutional neural network

During 2018, we explore a new method for classifying and recognizing several kind of common objects in the RoboCup game. The method is based on the convolutional neural network (CNN), which is constructed to identify and classify ball, robot, goalpost and the field, so that NAO robot can play well in the game. The recognition algorithm is supposed to have high-precision and real-time performance.

Firstly, we use image processing and segmentation method to find objects in the field. The object might be ball, robot, goalpost or a small plot of the field. Then, we build a high-precision CNN model to classify these four kind of objects. After that, the XNOR-Net algorithm and SSE instructions were used to compress and accelerate the CNN model, which can improve the real-time performance of the algorithm. Finally, we import the CNN model into NAO robot. The experiments prove that the CNN model works well on NAO robot during RoboCup game. Figure 3.5 shows the whole process of object detection.

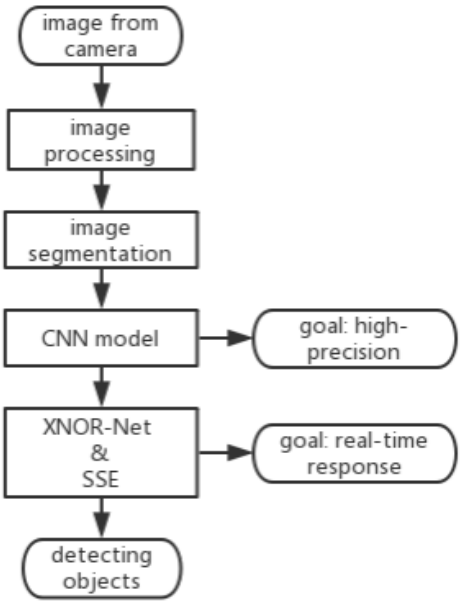


Figure 3.5 Flow chart for detecting object with CNN

3.5.1 CNN model for high-precision result

After obtaining the image from NAO's upper and lower camera, we firstly use image processing and segmentation methods to find and locate the objects, which might be ball, robot or goalpost. For more details, you can see the former chapters or turn to our Team Research Report 2017[1].

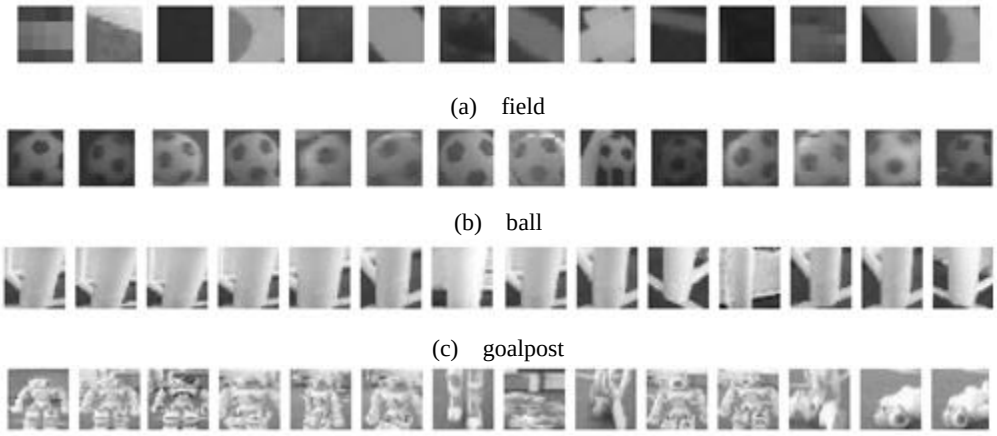
Then, based these objects, the next step is to make a dataset contained these as training set. With training and testing the CNN model on computer, the results show that the CNN model can surly provide a high-precision result for classifying four kind of objects.

3.5.1.1 Make dataset

As we can find and locate the objects, we first export a large number of patches containing the individual object. It should be noted that, considering the changes in light, we collect dataset over multiple periods of time so that our dataset is more adaptable.

Then we resize all these patches to a resolution of 32*32. During resizing, for the color is not the main information to distinguish these four kind of objects, so we transform the color image into gray image at the same time.

After that, we manually classify the objects into four folders. Figure3.6 shows some examples of the classified dataset.



(a) field

(b) ball

(c) goalpost

(d) robot

Figure 3.6 some images of each kind of objects

Besides, in order to further improve the generalization performance of CNN, we use data enhancement method for each picture, that is, to increase brightness contrast, reduce brightness contrast, and horizontal flip for each image. Figure 3.7 shows the data enhance process. In the end, we choose 5000 images of each folder as training set, and 1000 images of each folder as testing set, eg. the training set contains 20,000 images while the test set contains 4,000.

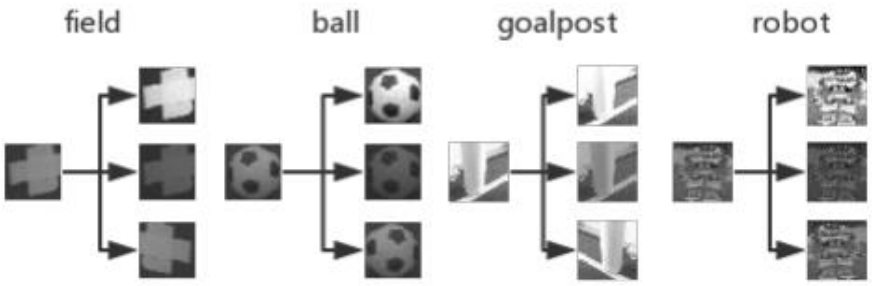


Figure 3.7 examples of data enhance

3.5.1.2 CNN model training and testing process

The convolution neural network model built in this study refers to a cuda-convnet model. Figure 3.8 shows the whole CNN atchitecture.

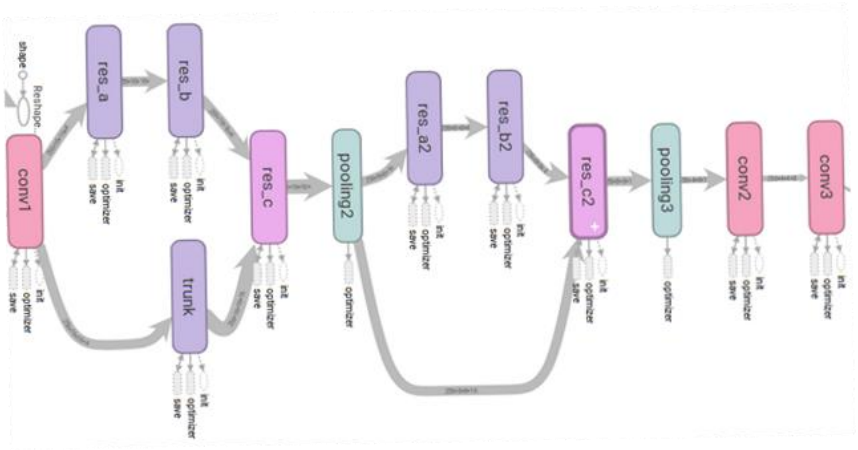


Figure 3.8 CNN architecture

In the training process, loss and accuracy of training will be output and saved every 50 training iterations, and verification will be carried out every 400 training iterations. We can see, when the training begins, the initial loss is about 1.4 and the initial accuracy is about 25%. Subsequently, the loss would continue to decline and the accuracy continues to rise. At the end of the training, the loss value of the network model was 0.012, and the accuracy rate reached 99.414%.

In the testing process, we input all the test dataset into the trained CNN network in turn for object classification. After average, the accuracy of the test dataset can reach 99.090%, which is a good result for NAO RoboCup game.

3.5.2 XNOR-Net and SSE for real-time response

Although CNN can bring high precision, its real-time performance is poor because there are many parameters and large amount of calculation during object classification. In RoboCup, real-time object detection is very important, so we use XNOR-Net and SSE method to improve real-time response so that the CNN model can work on well during RoboCup game.

3.5.2.1 Simplifying CNN with XNOR-Net

XNOR-Net is a binary convolutional neural network proposed by M. Rastegari et al. in 2016[2]. XNOR-Net binaries both the weight W and input X simultaneously, which not only reduces the storage space of the model, but also accelerates the convolution operation, achieves the compression and acceleration of the model.

After adding XNOR-Net to our model, experiments verify that could greatly increase the running speed of the model, shorten the running time and improve the real-time performance of the CNN model.

3.5.2.2 Accelerating CNN with SSE

The full name of SSE is Streaming SIMD Extensions, in which SIMD is also the abbreviation of four words, the full name is Single Instruction, Multiple Data. SSE is a single instruction multi-stream expansion, that is, after an instruction is issued, it can be put on different data at the same time to execute.

When we use SSE to speed up CNN, the time to recognize an image is further reduced by four times.

3.5.3 Experiment and result

Based on the results of simulation experiments and real experiments, the following results can be drawn for object detection algorithm based on CNN.

On the one hand, in terms of accuracy, the average accuracy of the algorithm can reach more than 98%, which fully meets the requirements of the competition. As a result, the CNN model can recognize and classify three common obstacles: robot, goal post and referee. It is worth mentioning that the algorithm classifies referee into field, because the referee will not interfere with the game after entering, so it will not affect NAO robot to make decision.

On the other hand, in terms of real-time response, the algorithm has high real-time performance. It can complete all visual tasks such as image preprocessing, object detection, and self-localization in time during the period of image capturing. It ensures that every frame of camera image can be effectively utilized.

Figure 3.9 shows the result of the CNN model with NAO's upper camera.

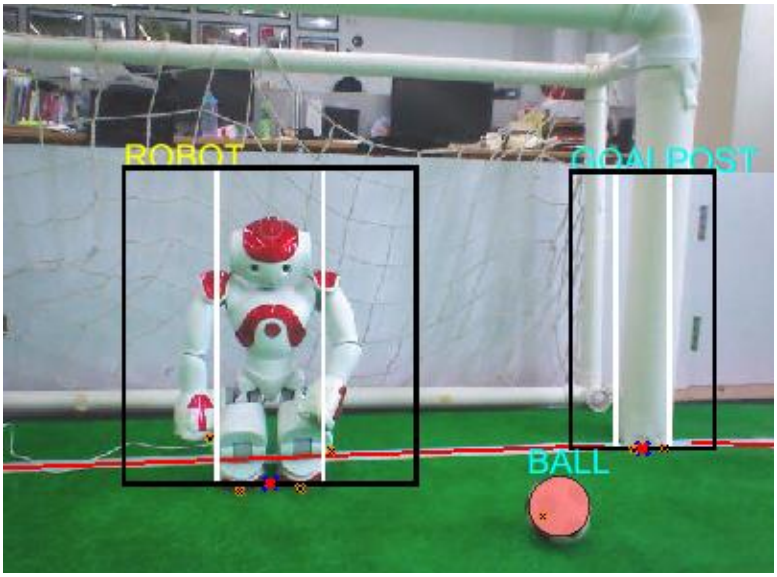


Figure 3.9 Actual performance of object detection

4 Localization System

Our team's self-localization is based on a particle filter with a low number of particles that each includes an Unscented Kalman Filter (UKF) and Iterative Closest Point (ICP) algorithm. With a more stable and reliable Vision System developed, our Localization System become more accurate and reliable. At begin, we only use UKF, but we found that the robot is hardly able to re-localized itself once it is kidnapped or lost after not seeing any field features for a long time. Since the robot transfers the field features it seen relative to its own coordinate system to world coordinate system according to its last robot pose, once it lost, in other words its last robot pose is wrong, it cannot match the field features correctly. As a result, it cannot re-localize itself. It may even "rebel" and attack our own side.

In order to fix this problem, we combine the ICP (Iterative Closest Point) algorithm with our localization system [3]. In each iteration of the ICP algorithm, it need to minimize the mean squared position error e, which is:

$$e(R,T) = \frac{1}{N}\sum_{i=1}^{N} w_i \| q_i - (R(\delta\theta)p_i + T(t_x,t_y)) \|^2 \tag{4.1}$$

where $R(\delta\theta)$ and $T(t_x,t_y)$ are the rotation matrix and translation vector between the estimated robot pose in two continuous iterations. q_i is the target points which is the ground true of the field features such as the goalposts, penalty area, T corners and L corners etc. p_i is the source points extracted from the observation of field features.

In order to apply the ICP algorithm, we linearize the above equation using a first order Taylor series approximation such that $sin(\delta\vartheta) = \delta\theta$ and $cos(\delta\vartheta) = 1$. Equating the first order partial derivatives of e with respect to $\delta\vartheta$, t_x , and, t_y to zero, we can get the following matrix for each point pair q_i and p_i:

$$\begin{pmatrix} 2w_i & 0 & -2w_i p_{i,y} \\ 0 & 2w_i & 2w_i p_{i,x} \\ -2w_i p_{i,y} & 2w_i p_{i,x} & 2w_i p_{i,y}^2 + 2w_i p_{i,x}^2 \end{pmatrix} \begin{pmatrix} t_x \\ t_y \\ \delta\theta \end{pmatrix} = \begin{pmatrix} 2w_i q_{i,x} - 2w_i p_{i,x} \\ 2w_i q_{i,y} - 2w_i p_{i,y} \\ -2w_i p_{i,y}(q_{i,x} - p_{i,x}) + 2w_i p_{i,x}(q_{i,y} - p_{i,y}) \end{pmatrix} \tag{4.2}$$

The least squares solution can be found using the SVD decomposition. Then the robot pose in the iteration k can be update use the following equations:

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix} = R(\delta\theta)\begin{pmatrix} x_{k-1} \\ y_{k-1} \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{4.3}$$

$$\theta_k = \theta_{k-1} + \delta\theta \tag{4.4}$$

Overall, the steps of our ICP algorithm include:

Step 1: Extract the source points from the observation of field features and find the corresponding target points.

Step 2: Reject some outlier.

Step 3: Solve the equation (4.1) and update the estimated robot pose according to equation (4.2) and

(4.3)

Step 4: If the mean squared position error is minimized to a threshold or we reach the maximum number of iterations, end the algorithm, otherwise, go to Step 2.

With this algorithm, when the robot sees multiple landmarks, it can match those features with landmarks on the field correctly after a few iterations of ICP algorithm. As a result, the robot can re-localized itself quickly.

In addition to the combination of ICP algorithm, we also use the z-axis gyroscope to measure the rotation of robot pose since the V5 version Nao equips with a 3-axis gyroscope. Thanks to this little modification, the robot still has a relatively accurate rotation estimation after falling down.

With the above techniques, our location performance was better in RoboCup2018.

5 Locomotion System

This part will describe the walking and dynamic kicking. The walking control system is based on the model pre-observation control, which aim to develop our own walking engine. And the dynamic kicking is mainly based on the dynamic movements primitives, which can make robots kicking ball more flexible and improve its adaptability to the external environment.

5.1 Walk control

The general problem of bipedal walking is how to place the feet and move the rest of the body. In the past, we directly generate the footprint of the robot without feedback, or use LIMP to generate CoM of the robot. Both of them can achieve a good walking. However, there will be big delays in walking. To overcome the delay of the ZMP tracking, we chose the method of preview control to generate online dynamic walking.

5.1.1 Outline of walking engine

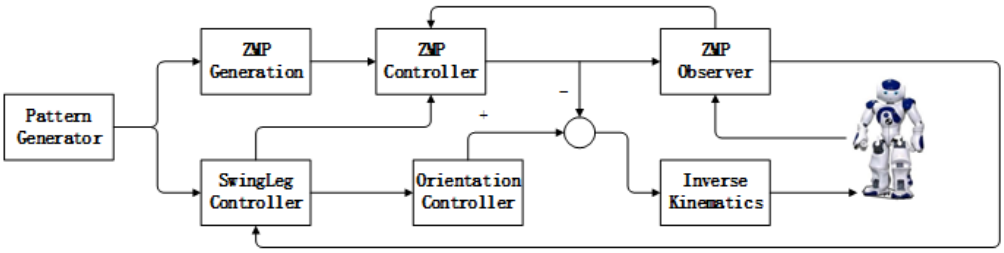


Figure 5.1 Control structure of walking engine

The concept of the zero moment point(ZMP) [4] is widely used to judge the dynamic stability of a walking bipedal robot. The ZMP is the point on the ground where the tipping moment acting on the robot, due to gravity and inertia forces, equals zero. For a stable posture, the ZMP has to be inside the support polygon. The input of our walking engine is the reference translational and rotational speed. The reference ZMP trajectory have been obtained through processing of the Pattern Generator. Then, the controller can generate the CoM of the robot. After inverse kinematics, we obtain the joint value of the robot. In order to gain a closed-loop system, we add a ZMP observer to collect sensor feedback.

5.1.2 Preview control

It is possible to measure an approximated ZMP with acceleration sensors by using the following equations:

$$p_x = x - \frac{z_h}{g}\ddot{x} \tag{5.1}$$

$$p_y = y - \frac{z_h}{g}\ddot{y} \tag{5.2}$$

Using the table-car model as the controlled object of a dynamic system, the servo system of ZMP target trajectory tracking can be constructed as shown in the following figure 5.2.

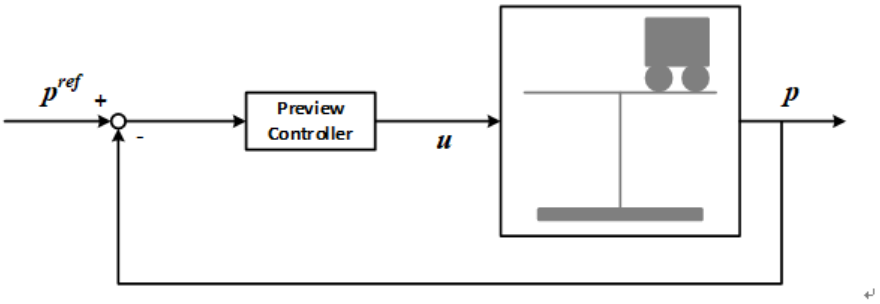


Figure 5.2 System block diagram with preview controller

The input of the system is the reference ZMP trajectory, and the output is the actual ZMP trajectory. For ease of processing, the differential definition of car acceleration is generally defined as the input of the system, the state equation of the following system can be obtained.

$$\frac{d}{dt}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u, \qquad p_x = \begin{bmatrix} 1 & 0 & -\frac{z_c}{g} \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \\ \ddot{x} \end{bmatrix} \tag{5.3}$$

Discretization the system equation:

$$\begin{cases} x(k+1) = Ax(k) + Bu(k) \\ p(k) = cx(k) \end{cases} \tag{5.4}$$

Where

$$x(k) \equiv \left[x(kT) \quad \dot{x}(kT) \quad \ddot{x}(kT) \right]^T, \quad u(k) \equiv u_x(kT), \quad p(k) \equiv p_x(kT)$$

$$A_0 = \begin{bmatrix} 1 & T & \frac{T^2}{2} \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} \frac{T^3}{6} \\ \frac{T^2}{2} \\ T \end{bmatrix}, \quad C = \begin{bmatrix} 1 & 0 & -\frac{z_c}{g} \end{bmatrix}$$

In order to establish the structure of quadratic linear programming in modern control theory, a performance index needs to be defined:

$$J = \sum_{i=k}^{\infty} \left\{ Q_e e(i)^2 + \Delta x^T(i) Q_x \Delta x(i) + R\Delta u^2(i) \right\} \tag{5.5}$$

Given a feasible optimal controller for minimizing *J*:

$$u(k) = -G_I \sum_{i=0}^{k} e(k) - G_x x(k) - \sum_{j=1}^{N} G_d(j) p^{ref}(k+j) \tag{5.6}$$

Where $G_I, G_x, G_p(j)$ is calculated in reference [5]. Hence, given the pre-planned ZMP trajectory, the optimal system output and the next frame of the robot CoM trajectory can be

obtained by minimizing *J*. Next, the motion of each joint of the robot can be calculated by inverse kinematics of the robot.

5.1.3 Observer

Predictive control is only an open-loop control method. After manually adjusting a series of parameters such as the ratio of single-foot support phase to double-foot support phase and the initial step length, the robot can also maintain balance under some minor external impact when walking on the flat ground. However, when the walking parameters of the robot are not well modulated or the robot encounters strong external shocks in the course of walking, it is easy for the robot to fall because of its inability to adjust its attitude in time, so it needs state feedback. However, since the actual velocity of the center of mass in the state vector can't be detected in practice, an observer is added to the preview control system [6]. Observers are used to compute the estimated state matrix.

The system output using the observer is shown as follows:

$$y(k) = C_m \cdot x(k) \triangleq \begin{bmatrix} q_x^{mea}(k) \\ p_x^{mea}(k) \end{bmatrix} \tag{5.7}$$

$$C_m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

Where $q_x^{mea}(k)$, $p_x^{mea}(k)$ represent the CoM and ZMP positions detected by the sensor respectively.

The state space equation of the system with feedback link is as follows:

$$\hat{x}(k+1) = A_0\hat{x}(k) + L[y(k) - C_m\hat{x}(k)] + b\hat{u}(k) \tag{5.9}$$

Gain L can be obtained by using Discrete Linear Quadratic Regulator (Discrete LQR) programming. The appropriate gain can be easily solved by using function $dlqr()$ in Matlab when it is offline.

5.1.4 Swing leg controller

The trajectory of swing foot is generated by cycloid equation.

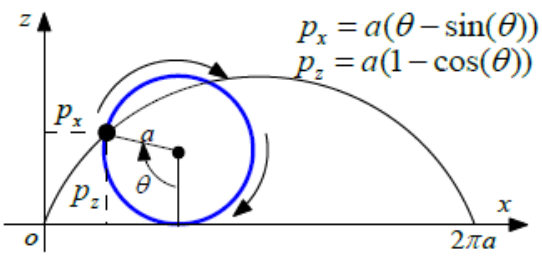


Figure 5.3 Cycloid Equation Generating Swing leg trajectory

The position of the swing foot when it leaves the ground is defined as $p_s = [x_s, y_s, z_s]^T$, and the position of the swing foot when it completes a gait movement is $p_e = [x_e, y_e, z_e]^T$, The cycloid from p_s to p_e is the trajectory generated by the swinging foot of the robot. By modifying the cycloid equation, the motion trajectories of swing feet with different elevation and walking steps can be obtained. Given the walking speed $v_c = [v_{cx}, v_{cy}]^T$ and the robot rotation angle ω, the next position of the robot can be expressed as follows:

$$\begin{cases} p^f = \left[\dfrac{\alpha}{2} v_{cx}, \alpha v_{cy} + y_s \right] \\ \theta^f = \alpha \omega_c \end{cases} \tag{5.10}$$

Where $\alpha = T/1000$, T is the walking period, y_s is the distance between robot's feet when it is standing, the omnidirectional motion of the robot is illustrated as follows:

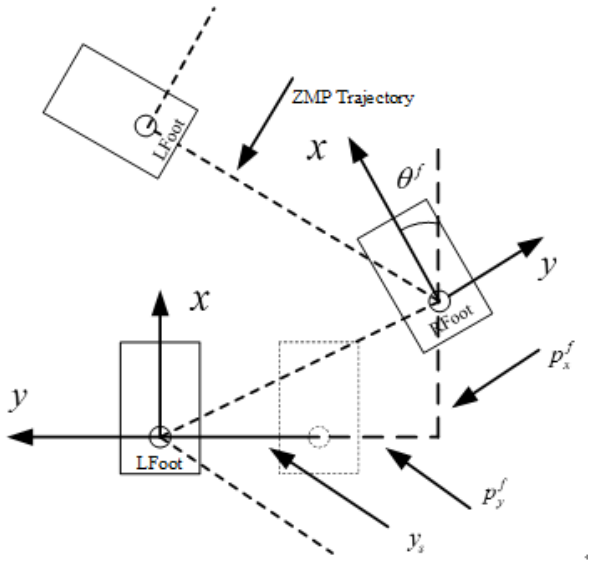


Figure 5.4 Omnidirectional walking of robot

5.2 Dynamic Kick

In every match, it is important to execute a dynamic kick for a pass or a shot on the goal. Since covering all possible kick angles and strengths with Special Actions is not possible, we should change the kick trajectory dynamically. Besides, the time and speed are key points in a match, but our previous kicking is relative slow, for that needs an alignment period before kicking. So developing the dynamic kicking is necessary, which can effectively improve these issues.

5.2.1 Dynamic movement primitives

Dynamic movement primitive [7] is actually a trajectory planning method. Because it is hard to achieve high accuracy to the ball and the ball is often sporty during kicking, the dynamic movement primitives is applied to follow the changed ball by changing the trajectory of the kicking foot, so that if robots reach the ball, the robots could execute a kicking in time.

The DMP used here is discrete, whose control strategy is point attraction. According to this, the basic model equation can show below:

$$\tau \dot{v} = K(g - x) - Dv + (g - x_0) f \tag{5.11}$$

$$\tau \dot{x} = v \tag{5.12}$$

Where x, v, $\dot{v}$ respectively represent the position, velocity and acceleration of the behavior trajectory. g indicates the ending position of the trajectory and x_o indicates the starting. Besides, f represents a forced item, which we used to change the motion trajectory while keeping the shape similar.

This method used to change the kicking trajectory is to learning the item f by initial primitives, e.g. initial kicking trajectory. With the learned f and basic equation, we can achieve the trajectory adjustment.

5.2.2 Initial kicking trajectory

The initial kicking trajectory can be set manually. It is divided into four phases:

- Lifting stage: raise the robot foot to a certain height.
- Back-swing stage: move the robot to a point behind.
- Forward-kicking stage: a fast straight motion towards the ball.
- Recovery stage: back the kicking foot to initial position.

With the four stages, the complete kicking trajectory can be gotten. And this static track is considered as the base primitive of the dynamic model.

5.2.3 Practicality

As the kicking trajectory is three-dimensional, which is independent of each other, we can use the DMP model to learn the shape of the track on this dimension in each dimension, and get the respective behavior weights w, which belongs to item f.

The implementation framework is as follows.

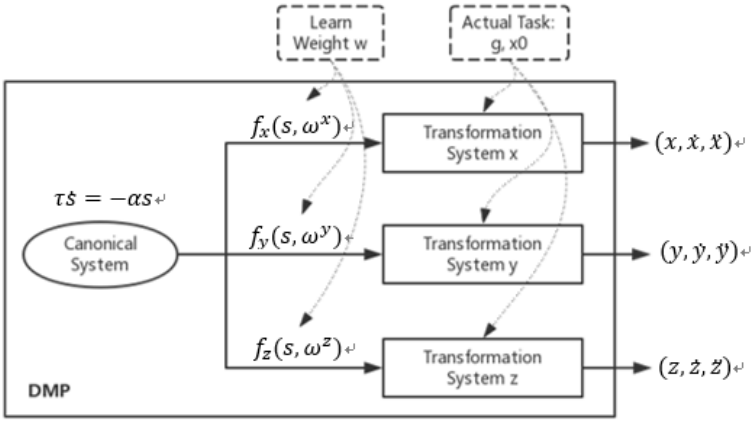


Figure 5.5 DMP framework for kicking trajectory

By the model, the kick track can be dynamically changed based on the actual ball position. Besides, if we solve the balance problem of robots, we can surly change the time of executing kicking, which can improve the kicking speed greatly.

5.2.4 Kicking action

We use DMP to learn the initial trajectory, the learning curve can be seen below figure 5.6.

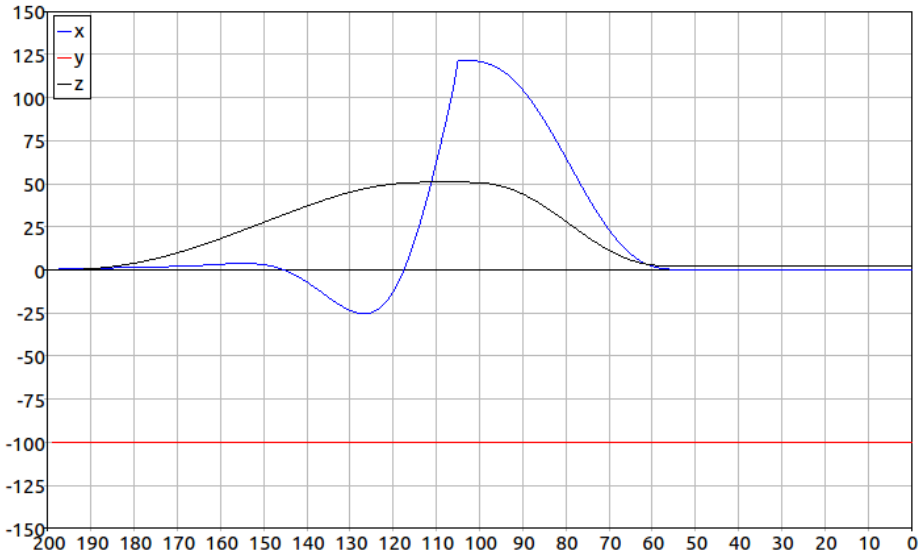


Figure 5.7 Original trajectory of kicking foot (mm)

Applying this track to kick foot, let robots carry out. We get the result as figure 5.8.

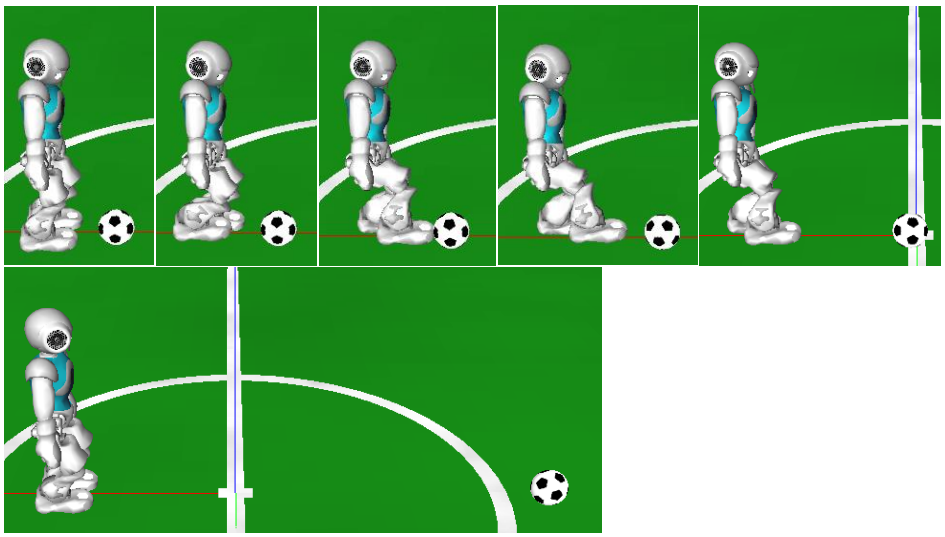


Figure 5.8 Kick action (Simulation)

When the ball position is changed, but still within the reachable range, the robot can also execute an effective kick by using dynamic kick.

6 Whistle System

We implement a whistle recognizer based on STFT. The main procedure is as follows.

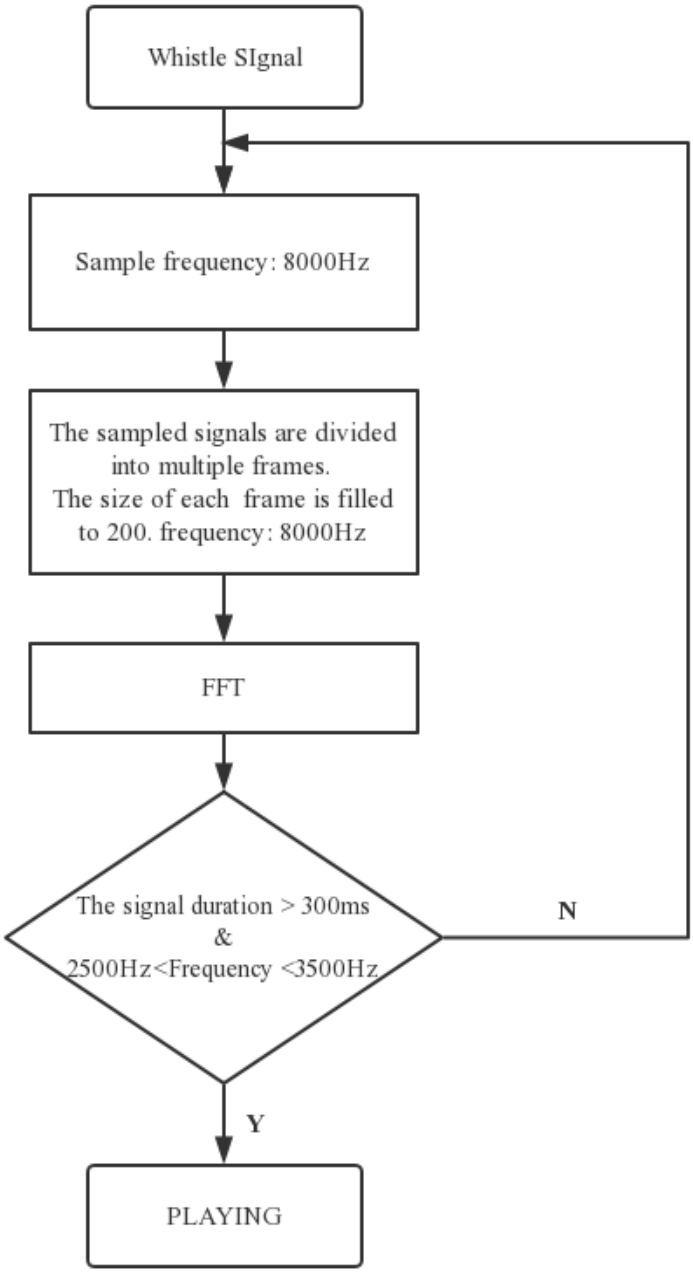


Figure 6.1 Procedure of whistle recognizer

By using this module, we recognized almost all whistle successfully in competition.

7 Path-planning Module

The significance of path planning is that the robot cannot always select the straight path in the process of going to a certain position during game. For example, there sometimes exists the obstacle robot (including the team members and the opposing members) on the straight path. Another example like that the robot cannot walk into the forbidden area in the special circumstances (otherwise it will cause an illegal action for the team). Therefore, we need an intelligent and efficient path planner. Considering the competition environment and the number of robots, we design a path planner for the environment with sparse obstacles based on the RRT algorithm [8].

7.1 Theoretical basis

The basic RRT algorithm use a series of random nodes to explore the feasible paths in the environment. There exist several severe shortcomings in the basic RRT algorithm: the strong randomness, high redundancy degree of the generate path, long search time and so on. We use some tricks to lift algorithm performance including setting goal-bias[8], introducing heuristic search[9] and smoothing path.

1) Setting goal-bias: using the target as a "proposal" growth direction of the random tree. An appropriate goal-bias probability can quickly guide the random tree to extend the target and reduce the quantity of search nodes. However, the selective probability should not be set too high in order to through the "small gap".
2) Introducing heuristic search: heuristic search is the key of A* algorithm. It will estimate the total length of the path through a suitable algorithm to influence the random search process. It plays an important role in guiding the random tree growth and smoothing the path. But it brings the challenge of computing power because its algorithm implementation is concerned to the number-sorting problems.

7.2 Results

The planner algorithm is tested repeatedly in the Simulation, which proves that the algorithm performs well in respects of calculation time and planning effect. A simulation scene example is shown as below.

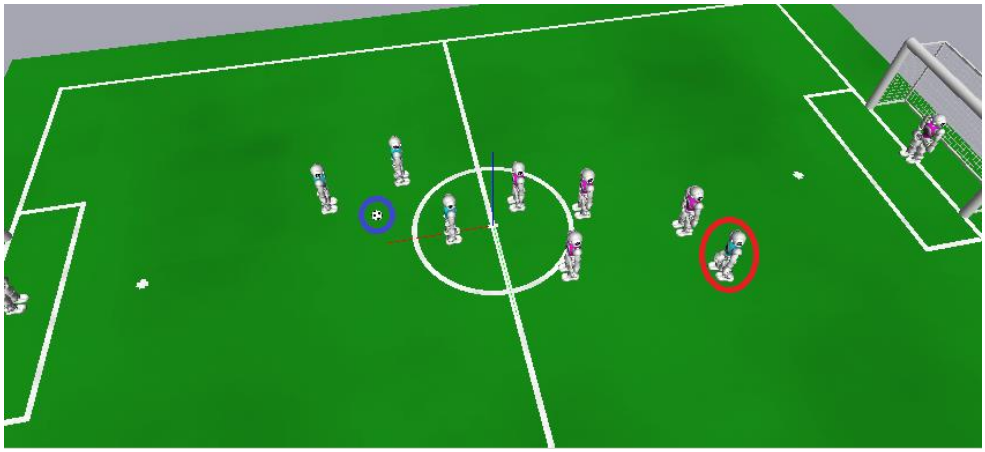


Figure 7.1 A pending planning scene

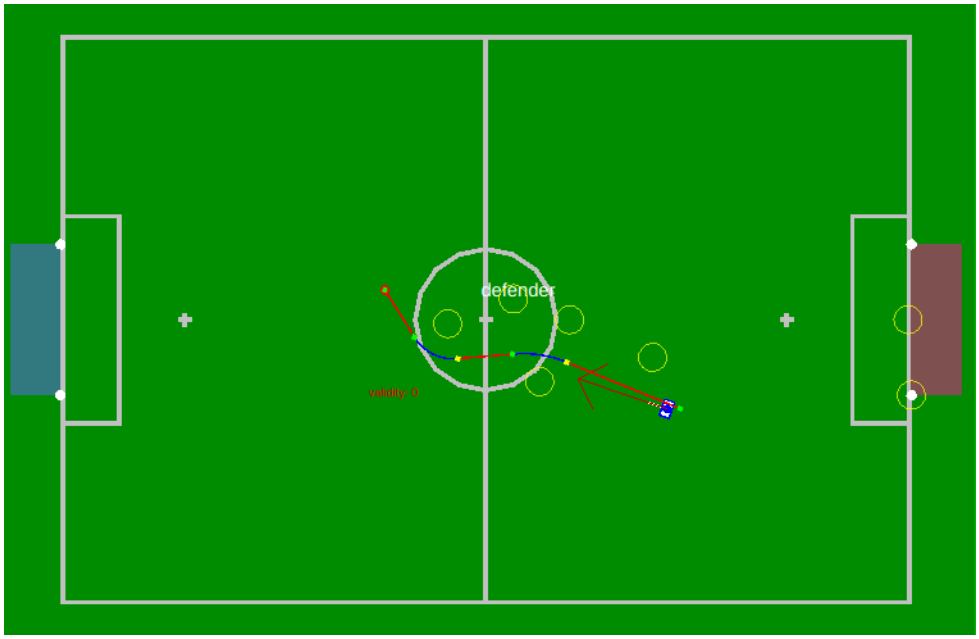


Figure 7.2 A solution of our algorithm

8 Decision System

This part will introduce a brand new strategy for the decision system. We intend to address these problems by using Reinforcement Learning. We applied a layer-study model based on SPL Game environment. In the first layer, the robot learns the base action such as dribbling and kicking. And in the second layer, the robot learns to select the action by the environment.

8.1 Base Action Layer Learning

Taking dribbling as an example, we take the model in [10] as a reference and try many RL training ways such as Q-learning and SARSA. As is shown in picture below, we build the robot-ball model based on the reasonable assumption. The state space is set to be the speed of the ball, the speed of the robot and the distance between them while the action space is set to be the speed of the robot and the acceleration of the robot. And the reward function is set to make the robot walking faster and closer to ball at the same time.

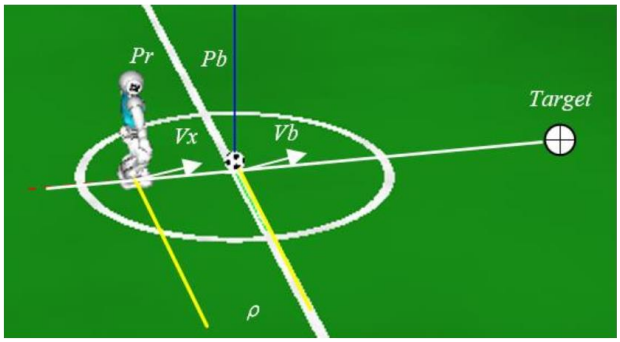


Figure 8.1 The robot-ball model in dribbling learning

After testing on real robot, the result shows that it achieves higher ball-control rate than the fixed-speed dribbling strategy.

8.2 Action Selection Layer Learning

We try to implement the action selection layer learning targeting on Half-Field Offence Task in which a striker's offence strategy is learned. The state space is simplified as the area of robot, ball and opponents, and the action space is set as dribbling and kicking at fixed angle. The reward function is more complicate here, so we build the function considering the score, the ball-control rate the opponents' ball-control rate and others.

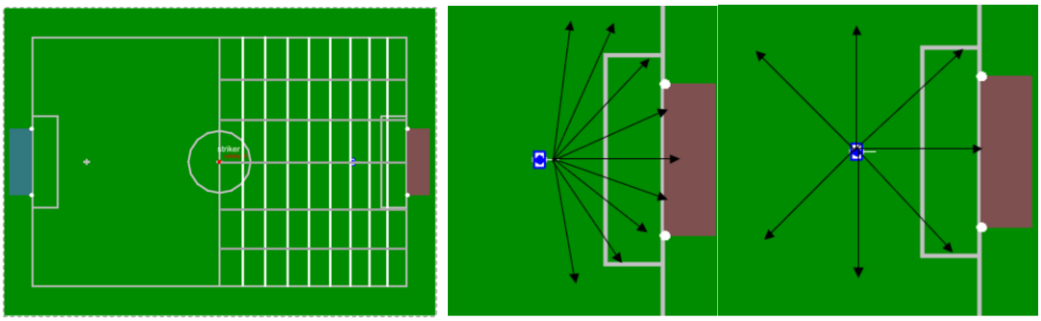


Figure 8.2 The action-selection learning model

We introduce a deep neural network to deal with the large state-action space, and the Double DQN by [11] is implemented here after modification. But the goals rate is not improved remarkably, so we didn't use our RL trained behavior selector model in RoboCup 2018. More training step on learning model is needed and the Multi-Agent Reinforcement Learning is the next problem we need to consider.

9 Publications

In this section, we list some of our notable work on NAO robot and research paper published or written in 2017 and 2018.

[1] Zhang T, Liu C, Chen Q. Rebalance control for humanoid walking based on online foot position compensation[C]// Ieee/rsj International Conference on Intelligent Robots and Systems. IEEE, 2017:4605-4610.

[2] Liu C, Ning J, An K, et al. Active balance of humanoid movement based on dynamic task prior system[J]. 2017, 14(3):172988141771079.

[3] Kang An, Chuanjiang Li , Zuhua Fang, and Chengju Liu. Effects of upper body parameters on biped walking efficiency studied by dynamic optimization. International Journal of Advanced Robotic Systems.2017: 1–13, DOI: 10.1177/1729881416682702.

[4] An Kang, Li Chuanjiang, Fang Zuhua, Liu Chengju. Efficient walking gait with different speed and step length: gait strategies discovered by dynamic optimization of a biped model. Journal of Mechanical Science and Technology.

[5] Shu Li, Qijun Chen. A Real-time Robust Calibration-free Color Segmentation Method for Soccer Robots. 2017 IEEE International Conference on Real-time Computing and Robotics.

[6] Chengju Liu, Junqiang Han, Kang An. Dynamic Path Planning Based on an Improved RRT Algorithm for RoboCup Robot. Robot Journal. 2017, 39(1):8-15. (In Chinese).

[7] Zhiying Zeng. Robot Localization based on UKF and ICP for NAO robot on RoboCup. Undergraduate thesis. (In Chinese).

[8] Dairong Li. Multi-robot behavior and communication for NAO robot in RoboCup competition. Undergraduate thesis. (In Chinese).

[9] Zhongde Chen. Action design and movement planning based on RoboCup SPL games. Undergraduate thesis. (In Chinese).

[10] Ruiming Zhang. Biped walking system based on ankle joint control. Undergraduate thesis. (In Chinese).

[11] Wenbo Shi. A Deep Learning Approach to Target Recognition for RoboCup Soccer-Robot. Undergraduate thesis, 2018. (In Chinese).

[12] Xiaoxian Sun. Design of Humanoid robot walking control method and experimental verification. Undergraduate thesis, 2018. (In Chinese).

[13] Xue Zhang. The Application of Series CPG Model in Walking Control of Humanoid Robot. Undergraduate thesis, 2018. (In Chinese).

[14] Haoran Zhou. Research and Verification of Humanoid Robot Control Strategy for Robust Walking. Undergraduate thesis, 2018. (In Chinese).

[15] Liang Tang. Research and Application of Model Learning Based on DMP in Humanoid Robot Behavior Control. Undergraduate thesis, 2018. (In Chinese).

[16] Hao Chen. A Reinforcement Learning Approach to RoboCup Soccer-Robot Behavior Control. Undergraduate thesis, 2018. (In Chinese).

Reference

[1] Shu Li, Ziqiang Zhou. TJArk Team Describe Paper & Research Report 2017, 2017. Web: http://github.com/TJArk-Robotics/coderelease_2017/blob/master/TJArkTeam-ResearchReport2017.pdf.

[2] M. Rastegari, V. Ordonez, J. Redmon and A. Farhadi. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks[J].Computer Vision - Eccv 2016, Pt Iv, 9908: 525-542.

[3] Peter Anderson, Youssef Hunter and Bernhard Hengst, An ICP Inspired Inverse Sensor Model with Unknown Data Association, in Robotics and Automation (ICRA), 2013 IEEE International Conference. May 6-10 2013.

[4] 陈启军，刘成菊.双足机器人行走控制与优化:北京.清华大学出版社,2016.

[5] Kajita S, Kanehiro F, Kaneko K, et al. Biped walking pattern generation by using preview control of zero-moment point[C]// IEEE International Conference on Robotics and Automation, 2003. Proceedings. ICRA. IEEE Xplore, 2003:1620-1626 vol.2.

[6] Oliver Urbann, Stefan Tasse. Observer based biped walking control, a sensor fusion approach. Autonomous Robots (2013)1-13.

[7] Ijspeert A, Nakanishi J, Hoffmann H, et al. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors[J]. Neural Computation, 2013, 25(2):328-373.

[8] Lavalle S M. Rapidly-exploring random trees: progress and prospects[J]. Algorithmic & Computational Robotics New Directions, 2001:293--308.

[9] Li J, Liu S, Zhang B, et al. RRT-A* Motion planning algorithm for non-holonomic mobile robot[C]// Sice Conference. IEEE, 2014:1833-1838.

[10] Leottau, David L., et al. "Decentralized Reinforcement Learning Applied to Mobile Robots." Robot World Cup. Springer, Cham, 2016.

[11] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." AAAI. Vol. 2. 2016.