TEAM RESEARCH REPORT & CODE RELEASE 2019

TJArk, Robotics & Artificial Intelligence Laboratory

Tongji University, Shanghai 201804, P.R China

TJArk.official@gmail.com

Jan. 15, 2020

Contents

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.Zhou Ziqiang	Control Science and Control Engineering
	Mr.Tang Liang	Control Science and Control Engineering
	Mr.Guo Xiang	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:

Guo Xiang, Control Science and Control Engineering, M.S Student.
Huang Zhengang, Control Science and Control Engineering, M.S Student.
Xiao Hongyuan, Control Science and Control Engineering, M.S Student.
Zhang Hao, Control Science and Control Engineering, M.S Student.
Liu Chuangwei, Control Science and Control Engineering, B.S Student.
Meng Ziyu, Control Science and Control Engineering, B.S Student.
Yuan Jiayao, Control Science and Control Engineering, B.S Student.
He Mengqi, Control Science and Control Engineering, B.S Student.

Figure 1.1: The majority of team TJArk members in RoboCup 2019

1.3 About Robot

Our team used V5 and V6 Nao robots to compete in Robocup2019. Next year we will also use V5 and V6 robots to participate in the competition.

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, from 2013 to 2019, we used B-Human framework on Nao V4/V5 robot, including the kick engine, debugging tools and CABSL according to the license. We developed our own vision, locomotion, localization etc. modules based on B-Human framework. Anyway, we are very grateful for the contribution of B-Human for SPL.

Besides, we would also like to thank rUNSWift for their walk engine. Our walk engine used in RoboCup 2016/2017/2018 and RoboCup 2019 for NAO v6 was based on rUNSWift-2014-release.So let us express our heartfelt thanks to their devotions once again.

In 2019, we used both V5 and V6 robots. The detailed code usage is listed in Table 2.1.

	NAO v5	NAO v6
Code Usage:	B-Human framework (CodeRelease 2015)	HTWK LolaConnector rUNSWift walk engine (CodeRelease 2014) B-Human CABSL(iostream version)
Own Code:	Vision Walk Engine Localization Motion Planning Decision and behavior System	Code base Vision Localization Motion Planning Decision and behavior System

Table 2.1: Code usage on NAO v5 and v6 in RoboCup 2019

In 2020, we will fully use our own code base and walk engine, and apply all the new achievements we made in 2019 to the competition.

During 2019, we have made a lot of attempts and improvements for several 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/TJArkCodeRelease2019/blob/master/TJArk_TRR_and_CodeRelease2019.pdf.

Code Release: https://github.com/TJArk-Robotics/TJArkCodeRelease2019.git

2.3 Chapter Arrangement

As shown above, Chapter 1 introduces the basic information of Team TJArk, Chapter 2 (current chapter) mainly illustrates the code usage and the main content of the document.

The following chapters will introduce the works and contributions we have done during 2019, and the arrangements are listed as follows:

Chaper 3 will describe some of improvements made in 2019 based on our previous vision system, including field color detector, field border detector, features detector, ball detector(2019) and others objects detection. Among them, in view of our successful experience in deploying and classifying objects with CNN on NAO last year, we designed and implemented a Lightweight-CNN model this year, so that real-time end-to-end object detection can be performed. Besides, semantic segmentation based scene parsing result is also shown in this chapter.

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. The first is walk control. We improved the original walking control strategy and switched to preview control to achieve online dynamic walking. The other is that we updated our kicking-motion generator, which generates swing-foot motion when the robot kicks the ball.

The sixth chapter is the whistle module. In this part, we propose a computationally efficient and robust real-time 3D SSRL(Sound Source Recognition and Localiation) method which can address the sound source recognition and localize robots.

The seventh chapter is the path planning module. Actually, it is similar to our module in 2018. It will show the main procedure of our path planner for the environment with sparse obstacles based on the RRT algorithm.

The last, we would show some paper that were published or written by our laboratory in last 2 years.

3 Vision System

In 2019, based on last year, our vision system added a new CNN-based ball detector to perform a end-to-end detection, which increased the accuracy of ball detection and adaptability to complex environments.

In this part, we will simply describe some sub-modules in our vision firstly, which are developed on our 2017 and 2018 Team Research Report, if you want to read the previous part for details, you can turn to the 2017 [tjark1] and 2018 [tjark2]. Then we will describe a new method for improving ball detection 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.

Normal Situation:

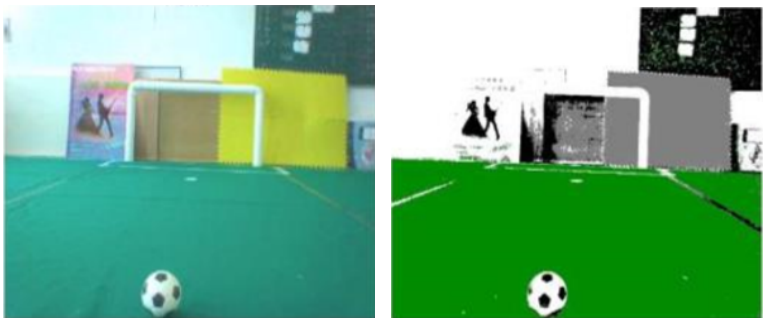


Figure 3.1: Normal Situation

Severe uneven light:

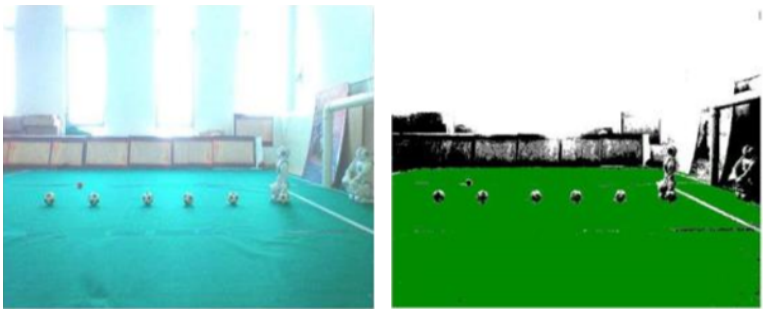


Figure 3.2: Severe uneven light

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 Scene Semantic Segmentation

Besides *GMM Field Color Detector*, we also designed a semantic segmentation network for field color classification, but the running time is about 25ms per frame which means that the algorithm can not meet the real-time requirements. So we didn't use it in RoboCup 2019. Recently, we have made new progress in the semantic segmentation network, which can run in real time (14ms). We will use it in RoboCup 2020. Figure 3.3 shows some typical segmentation results.

3.3 Field border detector

Field border is important in perception. Our implementation of field border detection is based on RANSAC. Figure 3.4 shows some of the results:

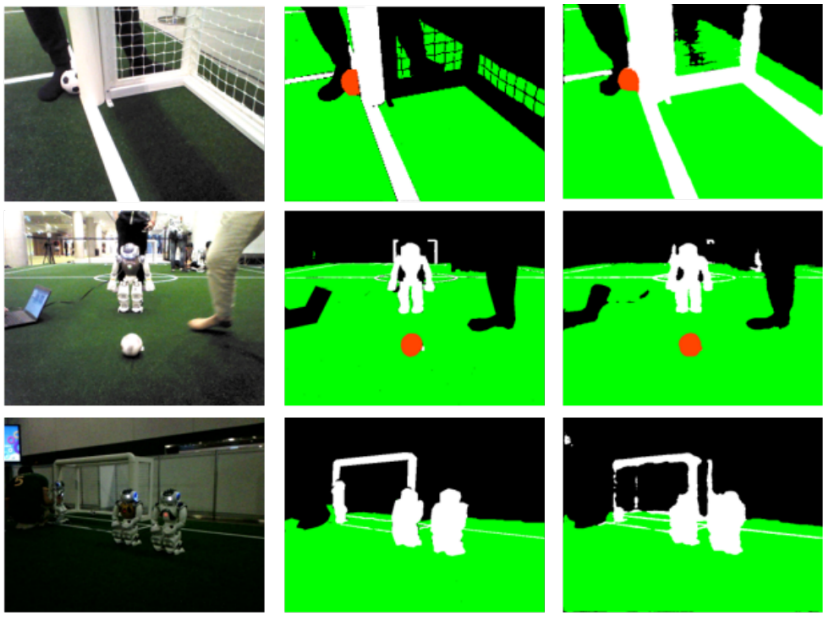


Figure 3.3: Field semantic segmentation

Figure 3.4: Results of field border detection

3.4 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 3.5 shows the detection results.

3.5 Detecting ball with convoluntional neural network

Compared to using a convolutional neural network to make a simple ball classifier in 2018, in 2019 we directly designed a CNN lightweight model directly to handle the end-to-end detection task of the ball. The following is a detailed description.

Figure 3.5: Results of line and center circle detection

Considering the hardware limitations of NAO robots, we propose a lightweight CNN object detection method for the CPU. At the same time, it guarantees that the detection process still performs well in the problems of small size, blurred images, and occlusion.

The entire implementation mainly includes four steps, namely making a data set, building a network structure, network optimization, and model deployment. The following is a detailed description.

3.5.1 Make Dataset

The proposed dataset was collected in our lab and real RoboCup competition fields, consisting of 1008 unique images with ball. Generally, the original images captured from the NAO's cameras is YUV format and the size of the images are lowered to 640 ×480 pixels and 320×240 pixels from the upper and lower camera, respectively. In order to speed up the operation process and improve the robustness under various scenarios, only the luminance (Y) channel of each image was extracted from NAO in action with various light conditions. Only when the original dataset obtained, were the ball pixels manually labelled. For the purpose of acceleration while ensuring detection accuracy, we resized the input images with label to a middle size of 416×416 pixels for later training and testing. An example of the proposed dataset is shown as Figure3.6. If you are interested in our dataset or pre-trained models, please contact us `TJArk.Official@gmail.com`.

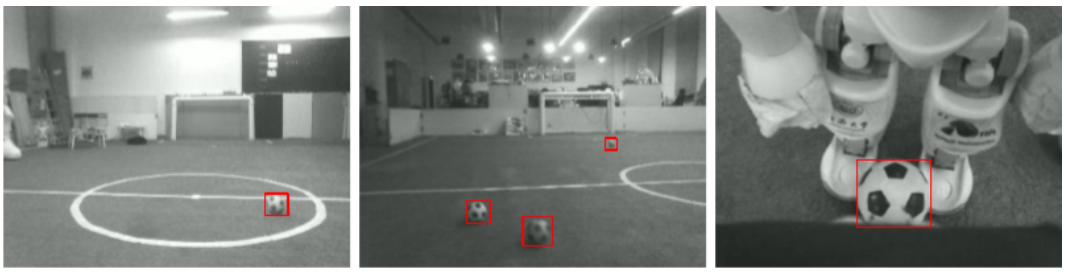


Figure 3.6: Examples of ball dataset

3.5.2 Design network structure and Optimaization

We no longer only apply CNN to classifier, we hope to use CNN to achieve end-to-end object detection. As consequence, we first build a backbone(Figure 3.7) with sufficient feature extraction and generalization capabilities. In order to deal with tiny size problem, we use anchor mechanism and design three anchors for different size objects. And under the premise of maintaining the accuracy for detection, we compress the standard CNN model as the backbone by reducing the number of layers and filters as much as possible.

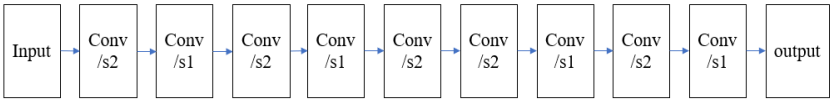


Figure 3.7: Backone Network

We refer to the idea of mobilenet. Under the premise of ensuring higher accuracy, the separable convolution is used to improve the real-time performance of the network. The core formula is as follows:

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F} = \frac{1}{N} + \frac{1}{D_K^2} \tag{3.1}$$

Where $D_K \cdot D_K \cdot M \cdot N$ is the size of a parameterized convolution kernel K,and $D_F \times D_F \times M$ is the size of the feature map taken as input.

We also use weight binarization to accelerate the computation procedure. Floating-point operation is time-consuming for device CPU, which is one of the most important factors restricting CNN running on CPU. Weights binarization can convert complex floating-point operations into simple XOR operations to accelerate the computation procedure. The core formula is as follows:

$$\begin{aligned} \mathcal{A}_k &= \tfrac{1}{n}\left\|\mathcal{W}_{lk}^t\right\|_{l1} \\ B_k &= \operatorname{sign}\left(\mathcal{W}_{lk}^t\right) \\ \mathcal{W}_{ik} &= c\mathcal{A}_{lk}\mathcal{B}_{lk} \end{aligned} \tag{3.2}$$

Where $\mathcal{W} \in \mathbb{R}_n, l, k$ represent kth filter in lth layer.

3.5.3 Model Deployment

After optimizing the model, we need to rewrite the network to integrate the code into our vision system. We use the SIMD instructions provided by Intel CPU to accelerate the operation on NAO robots to further enhance real-time performance. SIMD stands for Single Instruction Multiple Data. It can copy multiple operands and package them in a set of instructions in a single register. SSE is one of the instructions sets of SIMD which is supported by NAO's CPU. NAO uses 32-bit Intel CPU with 128-bit register length and 8-bit unsigned integer for CNN image input. Therefore, the operation of 32 pixel values can be processed at one time with SSE, leading to several times faster CNN calculation. We rewrite the convolution operation, batch normalization operation and ReLU nonelinear activation operation of CNN network with SIMD instructions.

In the end, we realized the real-time detection of the ball with high accuracy using a convolutional neural network. The following figures are our detection results on the PC and NAO robots.

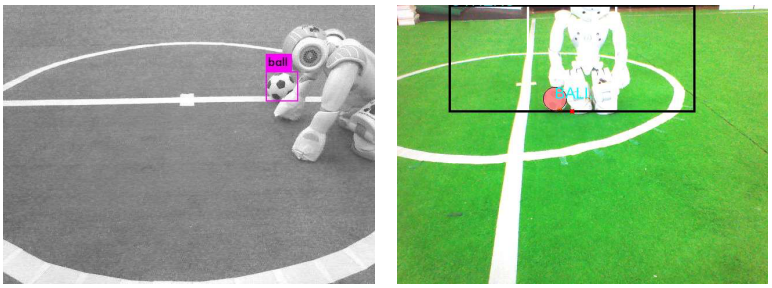


Figure 3.8: test result

3.6 Detecting other 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. But we implemented a new ball detector, so the classification for ball is no longer needed. 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 robot, goalpost or a small plot of the field. Then, we build a high-precision CNN model to classify these three 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 3.9 shows the whole process of object detection.

3.6.1 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 robot or goalpost. For more details, you can see the former chapters or turn to our Team Research Report 2017[tjark1]. 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.

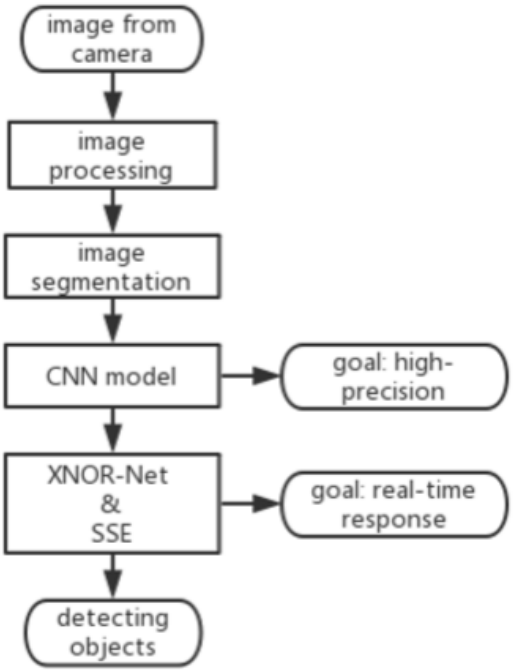


Figure 3.9: Flow chart for detecting object with CNN

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 three kind of objects, so we transform the color image into gray image at the same time.

After that, we manually classify the objects into three folders. Figure3.10 shows some examples of the classified dataset.

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. Figure3.11 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 15,000 images while the test set contains 3,000.

CNN model training and testing process

The convolution neural network model built in this study refers to a cuda-convnet model. Figure 3.12 shows the whole CNN atchitecture.

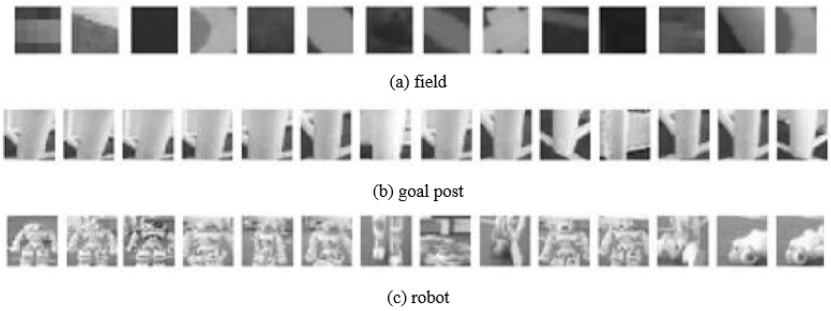


Figure 3.10: Some images of each kind of objects

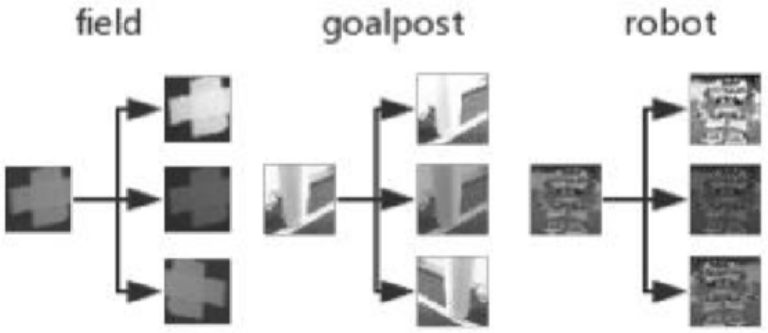


Figure 3.11: Examples of data enhance

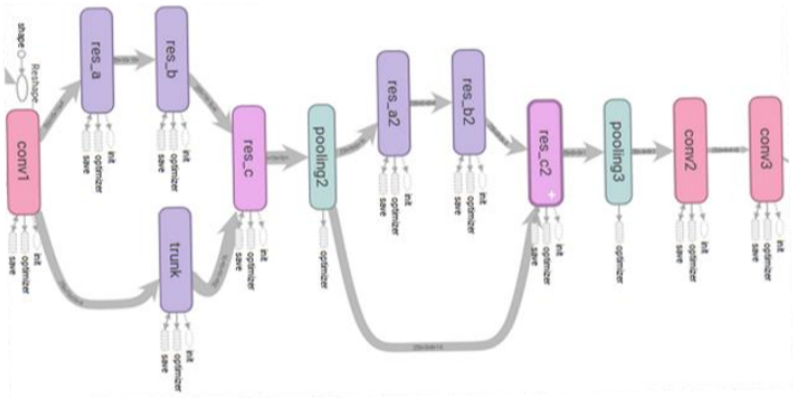


Figure 3.12: CNN architecture

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.

3.6.2 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 o bject 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.

Simplifying CNN with XNOR-Net

XNOR-Net is a binary convolutional neural network proposed by M. Rastegari et al. in 2016 [xnor]. 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.

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.

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 rec-

ognize 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 3.13 shows the result of the CNN model with NAO's upper camera.

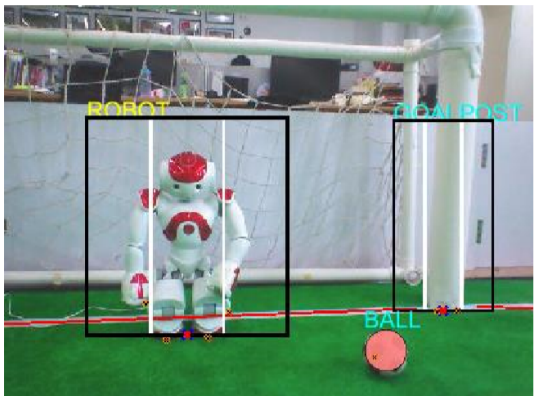


Figure 3.13: Actual performance of object detection

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 [Anderson2013An]. I n 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 $\mathrm{R}(\delta\theta)$ 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:

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

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:

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

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

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 equation 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

5 Locomotion System

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 choose the method of preview control to generate online dynamic walking.

5.1.1 Outline of Walking Engine

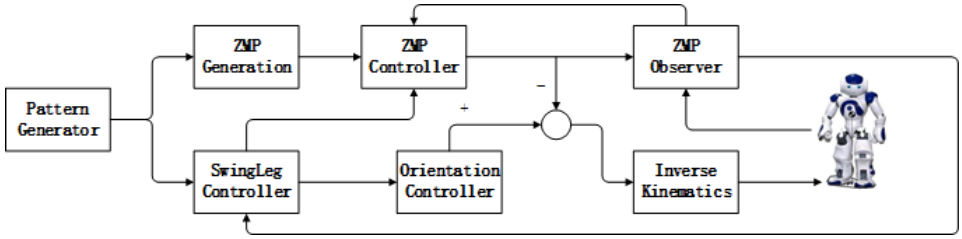


Figure 5.1: Control structure of walk engine

The concept of the zero moment point(ZMP) 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 is supposed 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 make our robots walk more stably, some strategies are used to improve the walking stability and anti-interference ability of walking. Firstly, in order to gain a closed-loop system, we add a ZMP observer to collect sensor feedback. Secondly, we plan the swing leg trajectory considering the biped support stage. Thirdly, an upper body posture controller is used to adjust the body posture online.

5.1.2 Preview Control

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

$$\begin{aligned} p_x &= x - \frac{z_c}{g}\ddot{x} \\ p_y &= y - \frac{z_c}{g}\ddot{y} \end{aligned} \tag{5.1}$$

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.

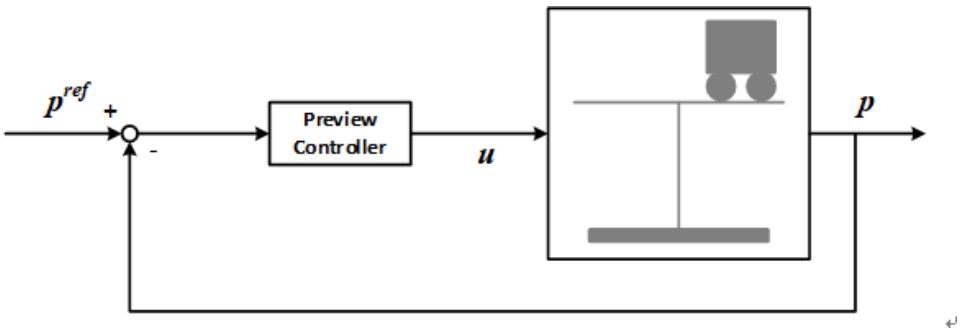


Figure 5.2: System block diagram with preview controller

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.

$$\begin{cases} x_{k+1} = A_0 x_k + b u_k \\ p_k = C x_k \end{cases} \tag{5.2}$$

Where

$$x_k = [q_x(k\Delta t), \dot{q}_x(k\Delta t), p_x(k\Delta t)]^T \tag{5.3}$$

$$u_k = u(k\Delta t) \tag{5.4}$$

$$p_k = p_x(k\Delta t) \tag{5.5}$$

$$\mathrm{b} = \begin{bmatrix} 0 & 0 & \Delta t \end{bmatrix}^T \tag{5.6}$$

$$C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T \tag{5.7}$$

$$A_0 = \begin{bmatrix} 1 & \Delta t & 0 \\ \frac{g}{z_c}\Delta t & 1 & -\frac{g}{z_c}\Delta t \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

p_x and q_x is the position of ZMP and CoM on the x-axis respectively. The state equation of motion in the y-axis is similar due to symmetry.

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

$$J = \sum_{i=k}^{\infty} \{ Q_e e(i)^2 + \Delta x^T(i) Q_x \Delta x(i) + R \Delta u^2(i) \} \tag{5.9}$$

Given a feasible optimal controller for minimizing

$$u(k) = -G_I \sum_{i=0}^{k} e(k) - G_x x(k) - \sum_{j=1}^{N} G_d(j) p^{rf}(k+j) \tag{5.10}$$

Where $G_I, G_x, G_p(j)$ is calculated in reference [KAJITA2003Biped]. 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 [Urbann2013Observer]. Observers are used to compute the estimated state matrix.

The system output using the observer is shown as follows:

$$y(k) = C_m \cdot x(k) \doteq \begin{bmatrix} q_x^{ma}(k) \\ p_x^{ma}(k) \end{bmatrix} \tag{5.11}$$

$$C_m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.12}$$

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

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

$$\hat{x}(k+1) = A_0\hat{x}(k) + L[y(k) - C_m\hat{x}(k)] + b\hat{u}(k) \tag{5.13}$$

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

5.1.4 Upper Body Posture Control

During the walking, we would like to keep robot's upper body upright. However, the robot's posture will change under disturbance, especially the upper body will tilt or even fall. Hence, a useful method is used to adjust the body posture online. An upper body posture controller is used to correct the body angles with the robot's hip joints.

For upper-body posture control, the correction angles of hip joints (u_{pitch},u_{roll}) using a PID controller are expressed as:

$$u_{pitch} = K_{p1}\Delta\theta^k_{\text{pitch}} + K_{i1}\sum_{i=1}^{k}\Delta\theta^i_{\text{pitch}} + K_{d1}\left(\Delta\theta^k_{\text{pitch}} - \Delta\theta^{k-1}_{\text{pitch}}\right) \tag{5.14}$$

$$u_{\text{roll}} = K_{p2}\Delta\theta^k_{\text{roll}} + K_{i2}\sum_{i=1}^{k}\Delta\theta^i_{\text{roll}} + K_{d2}\left(\Delta\theta^k_{\text{roll}} - \Delta\theta^{k-1}_{\text{roll}}\right) \tag{5.15}$$

where K_p, $K_i$and K_d are proportional, integral and differential gains of the PID controller. $\Delta\theta^k$ is the error between the ideal body angle θ^{ideal} which is set to 0 and the real body angle θ^{real} which is got by the sensor of the robot NAO. This control method is executed at every moment of walking process.

5.1.5 Swing Leg Controller

When the robot is walking, the support foot is on the ground and the swing foot is moving. During the single support stage, the trajectory of swing foot can be generated by cycloid equation. During the double support stage, two feet should be on the ground and the ZMP is from one foot to another foot smoothly. The cycloid could be shown in Figure 5.3.

The cycloid equation is:

$$\begin{aligned} p_x &= a(\theta - \sin(\theta)) \\ p_z &= a(1 - \cos(\theta)) \end{aligned} \tag{5.16}$$

Where θ is the function of time:

$$\theta(t) = \frac{t - t_b}{t_e - t_b} \cdot 2\Pi \tag{5.17}$$

Where t_b, t_e is the beginning time and ending time of one step period.

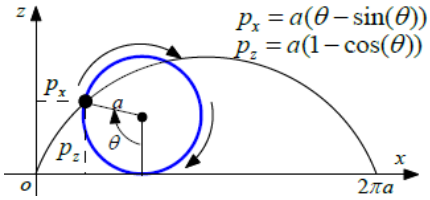


Figure 5.3: Cycloid Equation Generating Swing leg trajectory

5.1.6 Omnidirectional Walking

The omnidirectional motion is necessary for the robot. 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 $p_s = [x_s, y_s, z_s]^T$, 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:

$$\begin{cases} p^f = \left[\frac{\alpha}{2} v_{cx}, \alpha v_{cy} + y_s\right] \\ \theta^f = \alpha\omega_c \end{cases} \tag{5.18}$$

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:

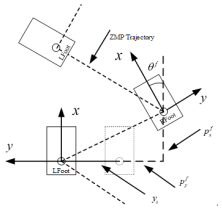
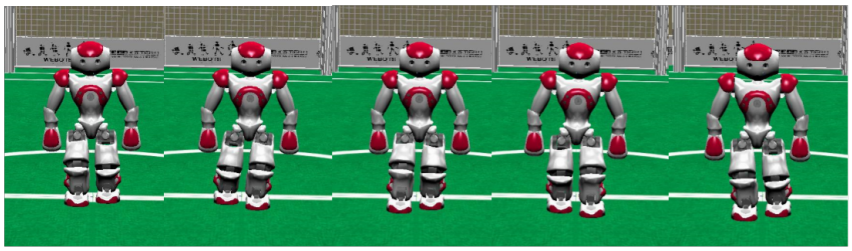


Figure 5.4: Omnidirectional walking of robot

The snapshot sequences of omnidirectional walking could be shown in Figure 5.5. Walking forward, walking sideways and rotating are shown respectively.



(a) Walking forward

(b) Walking sideways

(c) Rotating

Figure 5.5: Snapshot sequences of omnidirectional walking

5.2 Dynamic Kicking-motion Generator

5.2.1 Motivation

When robots are playing soccer on the field, their kicking-motions are crucial to scoring. The competitiveness of RoboCup requires more from our robot players, thus we have to update our kicking-motion generator, which generates swing-foot motion when the robot kicks the ball.

5.2.2 Method

This year, we applied a more dynamic way to plan the swing-foot motion. The following method is based on Speed-Variable Dynamic Movement Primitives(SV-DMP), a enhanced version of the

original DMP. This kind of motion primitive can not only plan trajectories according to given ball-position, but also speed-commands. It's quite helpful when the robot has to "control" the distance of its kick roughly.

The mathematical description of SV-DMP can be simplified as follow:

$$\begin{aligned} \tau\dot{y} &= z \\ \tau\dot{z} &= \alpha_z \cdot (\beta_z(g-y) + \dot{g}\tau - z) + \ddot{g}\tau^2 + \eta f \end{aligned} \tag{5.19}$$

Where g is a time-dependent polynomial and f is a weighed-sum of RBFs to imitate the topology of a given trajectory.

5.2.3 Offline Leanring

The first step of application is offline learning. Initially, the whole kicking-motion is simplified into three stages, which are the Shift Stage, the Lift-Kick Stage and the Reset Stage, separately. Swing-foot's trajectories during the latter two stages are sampled and refined, then learnt by VS-DMP. The results are as shown:

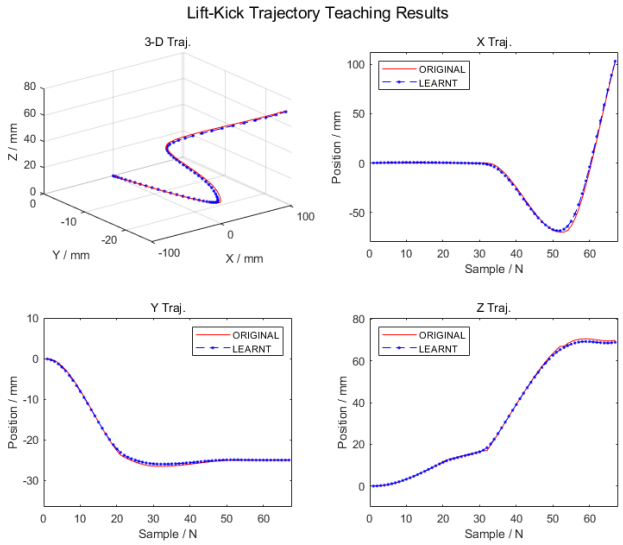


Figure 5.6: Lift-Kick trajectory Teaching Results

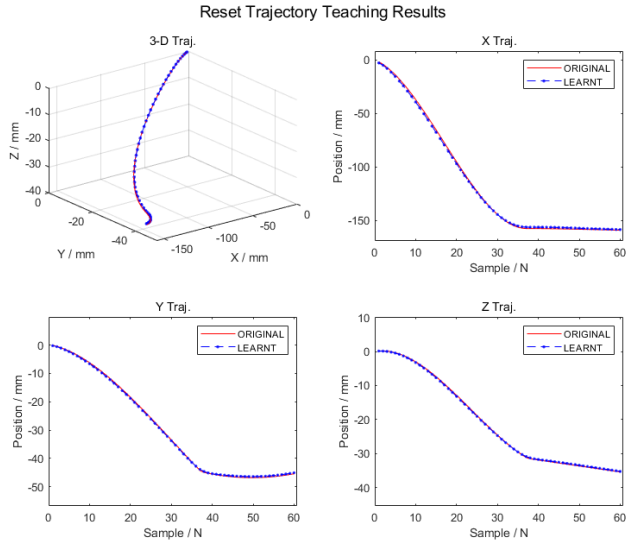
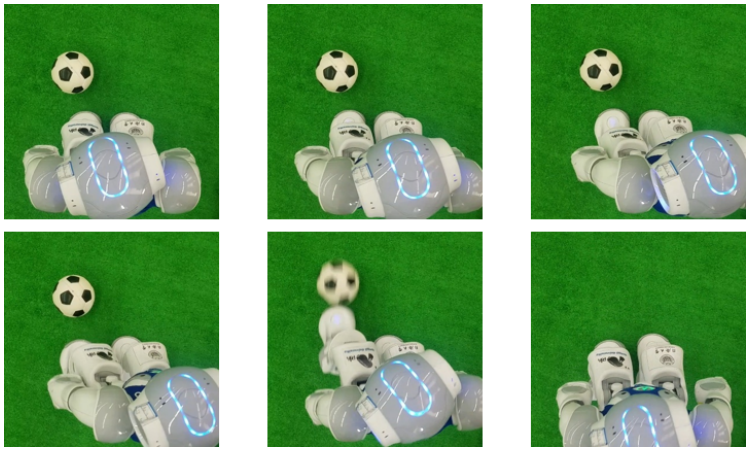
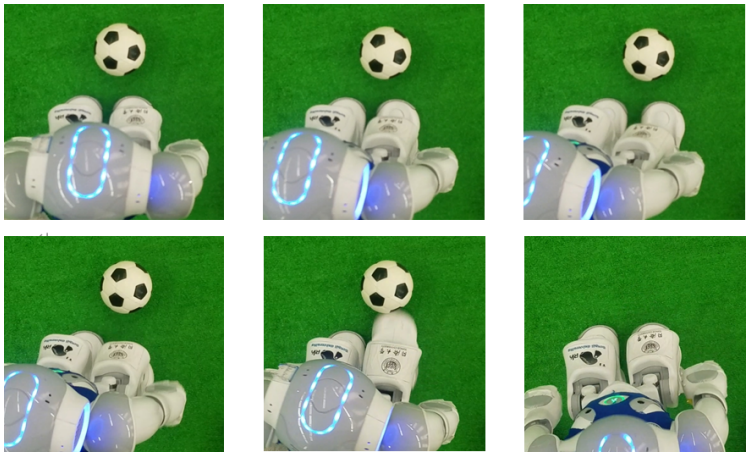


Figure 5.7: Reset Trajectory Teaching Results

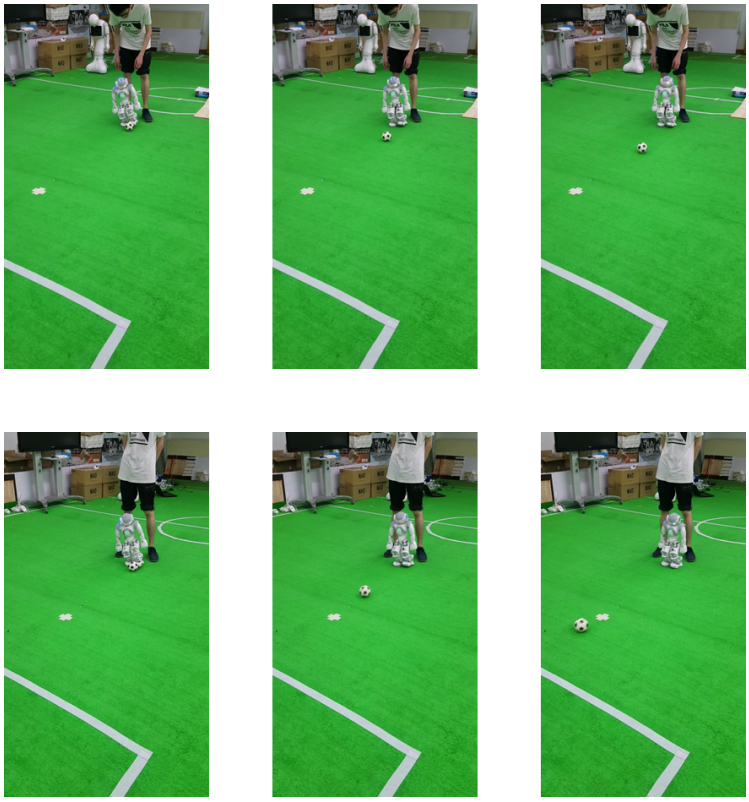
5.2.4 Real-Robot Application

The position command validity is experimented in two tries as follow. Different initial positions of the ball is set and a series of capture are shown in time-sequence.





Another two tries of kicking verifies our robot's ability to react to the speed command. The first try has a lower speed command, while the second try is commanded at a higher speed. To compare ball's speed, captures of each try start at the same time, hold the same time-interval.



6 Whistle System

Humanoid robots in RoboCup competition need to recognize the whistle bowled by referee as a signal to start the game. In this situation, the technology of sound source recognition and localization (SSRL) is employed to classify if the audio type is whistle and localize the accurate sound source position in order to avoid misidentification of whistle from other filed. Due to the fixed sensor arrays in humanoid robots and the fact that a sound event occurs at an arbitrary direction in 3D space with noise and reverberation, there comes challenge for SSRL. In addition, a lot of techniques like visual recognition need to be carried out simultaneously with the given computation resources limited humanoid robots. As a result, the computationally efficient and robust real-time 3D SSRL method is increasingly required.

In this part we propose a SSRL framework addressing the sound source recognition and localization in humanoid robots. A cross-correlation based recognition way is given to classify the audio type and a multi-robots collaboration system is used to locate the sound source. The single robot sound source direction estimator (SSDE) estimates the source direction by searching maximum SRP-PHAT function in discrete angle space while 2D Multi-robots sound source localization algorithm (SSL) and 3D SSL determine the source position. The SSRL aims at resolving the sound source recognition and localization problem under indoor and outdoor circumstances. Figure 6.1 shows the architecture of SSRL.

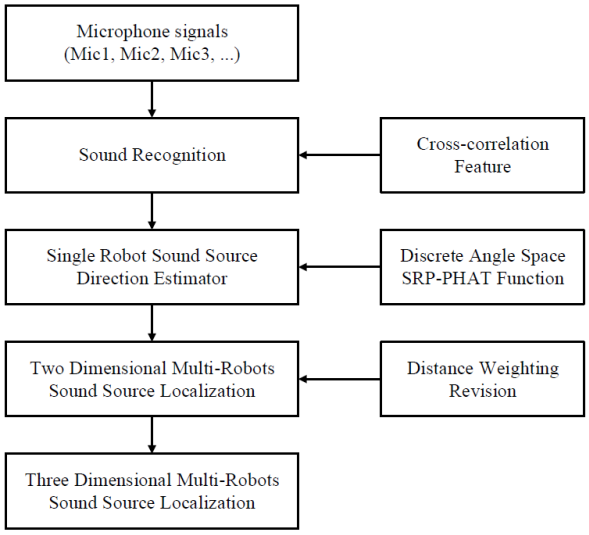


Figure 6.1: The SSRL framework

The contribution is as follows: (1) A robust and anti-noise sound recognition algorithm based on cross-correlation feature. (2) Improved SRP-PHAT function and simplified discrete angle space search with less computation cost. (3) Multi-robots collaboration localization system attaching distance weighting revision for more precise localization result. (4) Real-time application on Humanoid robots Nao equipped with microphone arrays like human.

6.1 Sound Source Recognition Algorithm

Considering the existence of noise and reverberation mixed in sampled audio data, the misidentification may occur if we classify the audio type by only detecting the principal frequency components.

The main procedure of sound source algorithm in this paper is demonstrated in Algorithm 1. The audio data is acquired and sampled from microphone in robot's head. After transferring the time-domain signal to frequency-domain signal using Fast Fourier Transform, a cross correlation function between the signal and pre-recorded reference signal is calculated to identify the certain audio type if its value is over the presetting threshold.

Algorithm 1 Sound Source Recognition

1: Sample audio data $X(n)$ from microphone signal $X(t)$ with sampling frequency f;
2: Load $X(n)$ to ring buffer $X(k)$ each T time;
3: Execute Fast Fourier Transform on $X(k)$ and recorded reference signal $R_f(k)$, thus get $F_1(w)$ and $F_2(w)$;
4: Process signal with band-pass filter $f_L - f_H$;
5: Calculate the Cross-Correlation function between two signals $R(\tau_n) = \frac{1}{N}\sum_{w=0}^{N-1} F_1^*(w)F_2(w)e^{j\frac{2\pi}{N}w}$;
6: If $\frac{\max(R(\tau_n))}{\max(R'(\tau_n))} >$ threshold $_T$, certain audio type is confirmed as T.

The empirical value of threshold is set to be 0.4 in use. As a result, we can deduce the audio type accurately based on the cross-correlation feature. The cross-correlation based way can be more robust comparing with principal frequency components detection based way.

6.2 Sound Source Localization Algorithm

6.2.1 Single robot sound source direction estimator

As is depicted in Figure 6.2, the humanoid robot NAO is equipped with four microphones. The array configuration is analogous to human-ears as they are distributed on left and right sides. An important cue for sound source localization is the TDOA , but it's not enough to locate the sound source based on the TDOA-model between two microphones because the cone of confusion will lead to a mirror-image position. In this section, we propose a single robot SSDE using all the four microphones to distinguish the source direction.

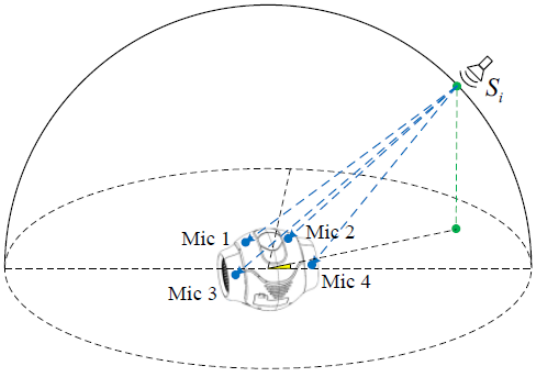


Figure 6.2: The microphone configuration in humanoid robot NAO and the straight sound source propagation path, the actual path may be more complicated

The coordinates that describe the source direction is shown as Figure 6.3.

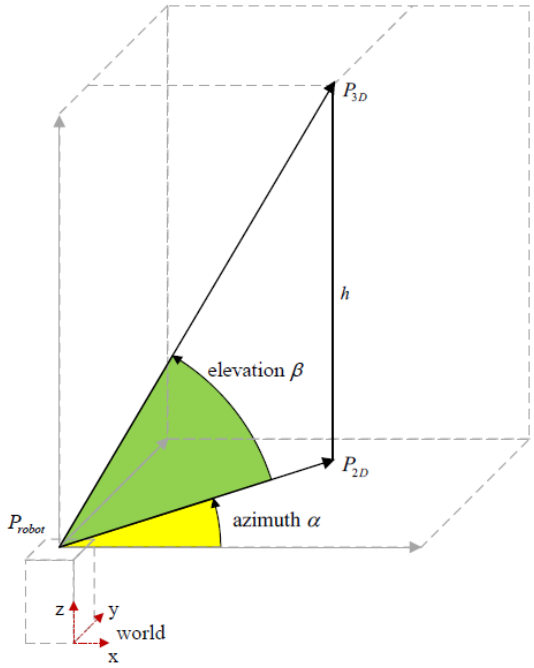


Figure 6.3: The coordinates used in SSDE. 2D Position and 3D Position in the figure are hypothesis points as the height is assumed to be the height of a referee in robot soccer game.

The SSDE algorithm is concluded as Algorithm 2.

Algorithm 2 Sound Source Direction Estimator

1: Get audio data segmentation $X_1(n), ..., X_M(n)$ from all M channels ring buffer $X_1(k), ..., X_M(k)$;
2: Execute Fast Fourier Transform on $X_m(k)$ and get $F_m(w), m = 1, ..., M$;
3: Calculate $\hat{R}_{lm}(\tau_n) = \frac{1}{N}\sum_{w=0}^{N-1} \frac{F_l^*(w)F_m(w)}{F_l^*(w)F_m(w)|} e^{j\frac{2\pi}{N}\tau_n}(l = 1...M; m = l+1)$ cross-correlation function between each two audio data segmentations;
4: Calculate SRP-PHAT function $\hat{P}(q'(\alpha,\beta)) = \sum_{l=1}^{M}\sum_{m=l+1}^{M}\hat{R}_{lm}[\tau_{lm}(q'(\alpha,\beta)]$ in discrete azimuthelevation angle space $Q'(\alpha,\beta)$;
5: Calculate the Cross-Correlation function between two signals $R(\tau_n) = \frac{1}{N}\sum_{w=0}^{N-1} F_1^*(w)F_2(w)e^{j\frac{2\pi}{N}w}$;
6: Search $\hat{q}_s(\alpha_s,\beta_s) = \arg\max_{q\in Q}\hat{P}(q(\alpha,\beta))$ and get certain sound source direction $\hat{q}_s(\alpha_s,\beta_s)$.

6.3 Multi-robot 2D sound source localization using distance weighting revision

SSDE provides the estimated sound source direction relative to single robot. SSRL introduces a multi-robot collaboration sound source localization algorithm using SSDE. As is shown in Figure 6.4, a 2D position can be estimated by crossing the azimuth angles and distance weighting revision. After combining with the elevation angle, the height can be revised and we can get a more precise 3D position. The 2D-SSL algorithm based on distance weighting revision will be presented in this section.

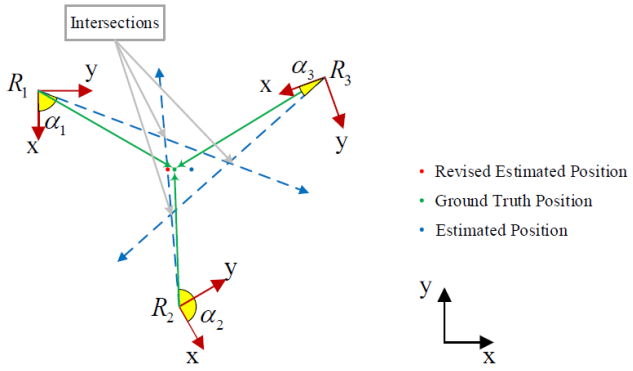


Figure 6.4: An approach for multi-robot two-dimensional sound source localization using distance weighting revision. (The blue point is estimated position as the average of three cross points, the green point is the ground truth of sound source and the red point is the revised estimated position)

To illustrate 2D-SSL, let's assume there are R humanoid robots NAO (R = 3 in this part) with initial pose $L_r = [x_r, y_r, \theta_r], \quad r = 1...R$ .For each robot, the relative sound source direction $q_r(\alpha_r, \beta_r)$ is calculated using SSDE, where α_r is the azimuth, and β_r is the elevation.

Intersect R robots' azimuth angle rays, and we can get C_R^2 intersections $P_i, i = 1 \dots C_R^2$. After taking average, an estimated 2D position is achieved as equation 6.1:

$$P_{\text{uncorrected}} = \frac{1}{C_R^2} \sum_i P_i \tag{6.1}$$

When the sound source distance is far greater than the microphone interval, we can analysis the sound propagation by assuming microphones are configured at the head chain origin point. For angle resolution sector region, the larger the radius, the longer the arc will be. ($L = r \times \alpha$). All points in the arc will be regarded as the same angle in current angle resolution. That is to say, in the same angle resolution zone, the closer the microphone is to the sound source, the smaller the arc length L and the smaller the area it represents. Correspondingly, the small shifting of sound source Δq will be reflected in small deviation of angle $\Delta\alpha$. Thus, closer distance to microphone means higher credibility of the identified sound source direction.

Using this criterion, 2D-SSL revises the estimated 2D position by distance weighting revision. For each robot, the distance between it and uncorrected sound source position is calculated as d_r , $r = 1, \dots, R$. Then we select the closest one to revise the position. The corrected position can be acquired by rotating the uncorrected pose to the azimuth ray of the closest one.

$$P_{2D} = P_{\text{corrected}} = L_{r_i} + d_\pi \cdot \begin{bmatrix} \cos(\alpha_{r_i} + \theta_{r_i}) \\ \sin(\alpha_{r_i} + \theta_{r_i}) \end{bmatrix} \tag{6.2}$$

Where r_i is the index of the robot with closet distance to sound source.

The corrected 2D position combines the information of distance, and it's more reliable than the uncorrected one. 2D-SSL in this section provides the 2D estimated position of sound source, and we will infer the 3D estimated position in next section.

6.4 Three-dimensional sound source localization

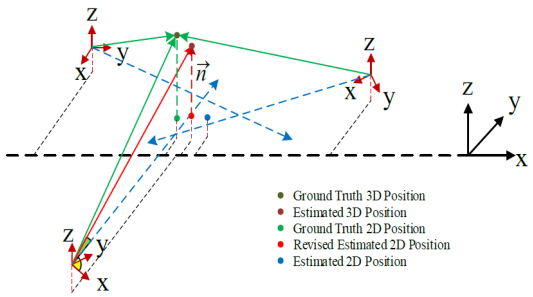


Figure 6.5: Three-dimensional sound source localization coordinates (The dark green point is the ground truth 3D sound source location, and the dark red point is the estimated 3D position. The green point, red point, and blue point have the same meaning as described in figure 6.4)

This section is meant to deduce the 3D estimated position of sound source. Once we get the 2D position from 2D-SSL, we can combine it with the elevation and work out the height of sound source to revise the height assumed before.

The whole 3D-SSL algorithm is concluded as Figure 6.6. SSDE works out azimuth and elevation using acquired microphone signal firstly. Combining angle information and robots' position, 2D position is deduced by 2D-SSL. Next, the elevation angle is used to revise the height and get 3D position by 3D-SSL.

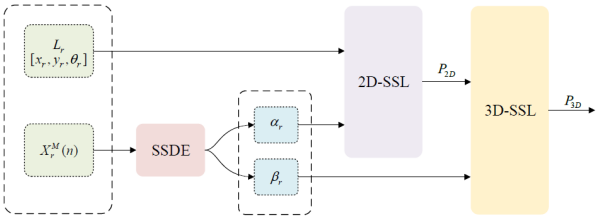


Figure 6.6: 3D sound source localization algorithm structure.

The main procedure is described in Algorithm 3.

Algorithm 3 Sound Source Localization

1: For robots located at $L_r = [x_r, y_r, \theta_r], r = 1 \dots R$, get their corresponding direction $q_r(\alpha_r, \beta_r), r = 1, \dots, R$, using Sound Source Direction Estimator ;
2: Intersect all robot-to-source direction line $\overrightarrow{L_r q_r}, r = 1 \dots R$ and generate intersecitons $P_i, i = 1, ..., C_R^2$;
3: Estimate uncorrected 2D sound source position as$P_{\text{uncorrected}} = \frac{1}{C_R^2} \sum_i P_i$;
4: Calculate distance $d_r = \| L_r - P_{\text{uncorrected}} \|$ and selct r_i s.t. $d_{r_i} = \min_{r=1,\dots,R} d_r$;
5: Get distance weighting revised 2D location as $P_{2D} = P_{\text{corrected}} = L_{r_i} + d_n \cdot \begin{bmatrix} \cos(\alpha_{r_i} + \theta_{r_i}) \\ \sin(\alpha_{r_i} + \theta_{r_i}) \end{bmatrix}$;
6: Intersect $\vec{n} = (P_{\text{corrected}}, \vec{n})$ and $\vec{\beta} = \left(L_n, \vec{\beta}_n\right)$, thus get the crossed 3D sound source position as P_{3D};

6.5 Experiments

We apply SSRL for 3D Sound source localization on humanoid robots Nao. In order to test the recognition and localization simultaneously, we design a whistle recognition and localization experiments in robot soccer competition field.

The experiment environment configuration is shown as Figure 6.7, robots are put in the predefined initial position and stay still. Once the referee blows the whistle in any place, the robot will

detect the signal and recognize the whistle. Meanwhile, the audio data segments from all microphones will be used to locate the whistle. The robots will contact with each other via Wi-Fi and share their SSDE result. Finally, the robot will calculate the 3D sound source position using 3D-SSL, and the console will present the result.

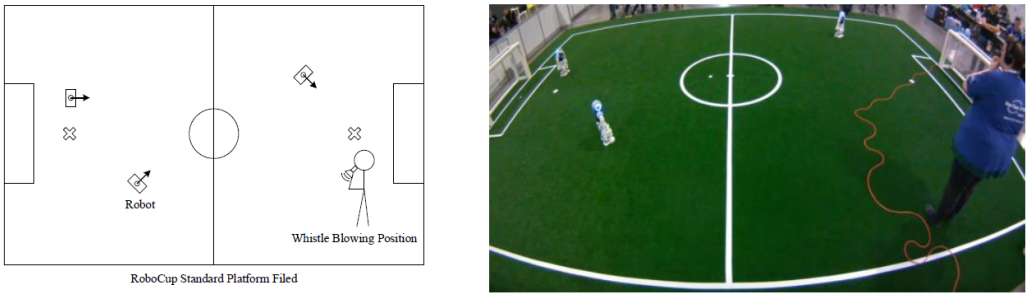


Figure 6.7: SSRL Test Environment. The robot is standing on the robot soccer standard platform competition field and the whistle will be blown by the referee.

Three indexes are used to measure the localization result in this paper: One is the threedimensional absolute position error, another is the relative distance error to the robot with closest distance to whistle and the other is the relative azimuth rotation error.

Suppose the actual position of whistle is q_{GT} , and the SSRL estimated position is q_s , then three indexes will be defined as equation 6.3:

$$\begin{aligned} e_1 &= \|q_{GT} - q_S\| \\ e_2 &= \|q_{GT} - L_{r_{\min}}\| - \|q_S - L_{r_{\min}}\| \\ e_3 &= \alpha_{GT} - \alpha_S \end{aligned} \tag{6.3}$$

The average error of three indexes are shown as Table 6.1 .

Table 6.1: Test result average error

	Average Error
$e_1(m)$	0.8636
$e_2(m)$	0.6760
$e_3(^\circ)$	4.8425

As we can see from the result, using humanoid robots microphone arrays, the SSRL framework proposed in this paper, can locate the sound source in both indoor and outdoor environments. The three-dimensional distance absolute error is approximately 0.8636m, the relative distance error is 0.6760m and the relative azimuth angle error is 4.8325°. In directional whistle technical challenge , the minimum distance error can reach to 0.049m and the minimum angle error can achieve 1.351879°.

Table 6.2: Data excerpts of RoboCup2019 SPL Technical Challenge

	Ground Truth	Estimated position	Score
1	(0, 3.35)	(0.532, 3.973)	1.54128
2	(4.85, 1.1)	(4.801, -0.187)	1.66435
3	(7.3, 5.9)	(10.506, 5.181)	1.51826
4	(2.25, -2.5)	(0.158, -1.874)	2.000

The sound source recognition and localization algorithm proposed in this paper is proved to be able to locate the sound source in indoor and outdoor environments on humanoid robots NAO with limited computation resources and critical real-time requirement after application testing. Compared with other methods, the SSRL achieves higher efficiency and robustness.

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 [10009988450].

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[10009988450], introducing heuristic search[Li2014RRT] 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.

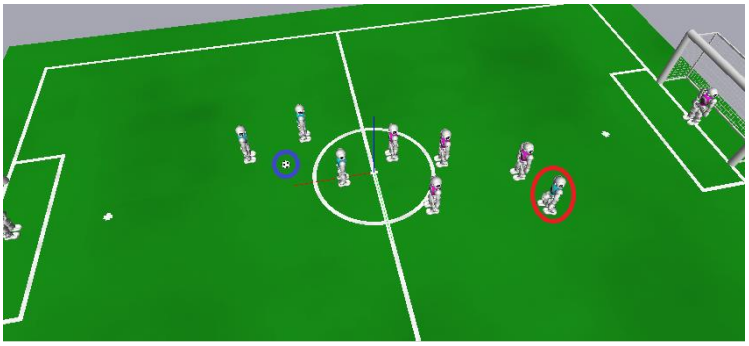


Figure 7.1: A pending planning scene

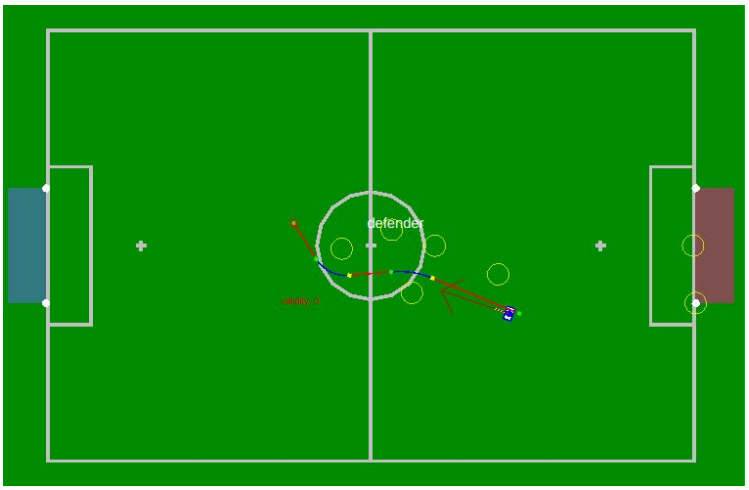


Figure 7.2: A solution of our algorithm

8 Publications

[1] Yan Q., Li S., Liu C., Chen Q. (2019) Real-Time Lightweight CNN in Robots with Very Limited Computational Resources: Detecting Ball in NAO. In: Tzovaras D., Giakoumis D., Vincze M., Argyros A. (eds) Computer Vision Systems. ICVS 2019. Lecture Notes in Computer Science, vol 11754. Springer, Cham

[2] Chen H., Liu C., Chen Q.(2019) Efficient and Robust Approaches for 3D Sound Source Recognition and Localization using Humanoid Robots Sensor Arrays

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

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

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

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

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

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

[9] Xiang Guo. Heatmap-based Robot Soccer Recognition System, 2019. (In Chinese).

[10] Hongyuan Xiao. A kicking Motion Programmer for the Humanoid Robot NAO Based On Variable End Speed Dynamical Movement Primitives, 2019. (In Chinese).

[11] ZhenGang Huang. Fall Detection and Control of Humanoid Robot