TJArk

Team Description Paper & Research Report 2017

TJArk, Robotics & Artificial Intelligence Laboratory

Tongji University, Shanghai 201804, P.R.China

(TJArk.official@gmail.com)

Feb. 12, 2018

Contents

Team Description Paper & Research Report 2017

Abstract. TJArk is a Chinese RoboCup team from Tongji University. They took part in RoboCup in 2006 for the first time and entered the quarter-finals in RoboCup 2007, 2008, 2016, 2017. All the members of this team are from Control Science and Control Engineering Department of Tongji University, China. This paper will provide a concise description of this team, including information about the research interests of its team members and the improvements on each part of TJArk's project. We will talk in detail about the notable work we did during 2017 and the field of interest we want to research in 2018.

.

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.
Affiliation:	Robotics & Artificial Intelligence Lab, Tongji University	
Location：	Shanghai, China.	

1.2 Team Members

The main part of researches and coding is done by the student researchers. Main student researchers:

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

Wei Penghao. Control Science and Control Engineering M.S Student.

Yong Haohao. Control Science and Control Engineering M.S Student.

Zhou Guangliang. Control Science and Control Engineering M.S Student.

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

Deng Xiuqi, Control Science and Control Engineering B.S Student.

Zhong Xianyou. Control Science and Control Engineering B.S Student.

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

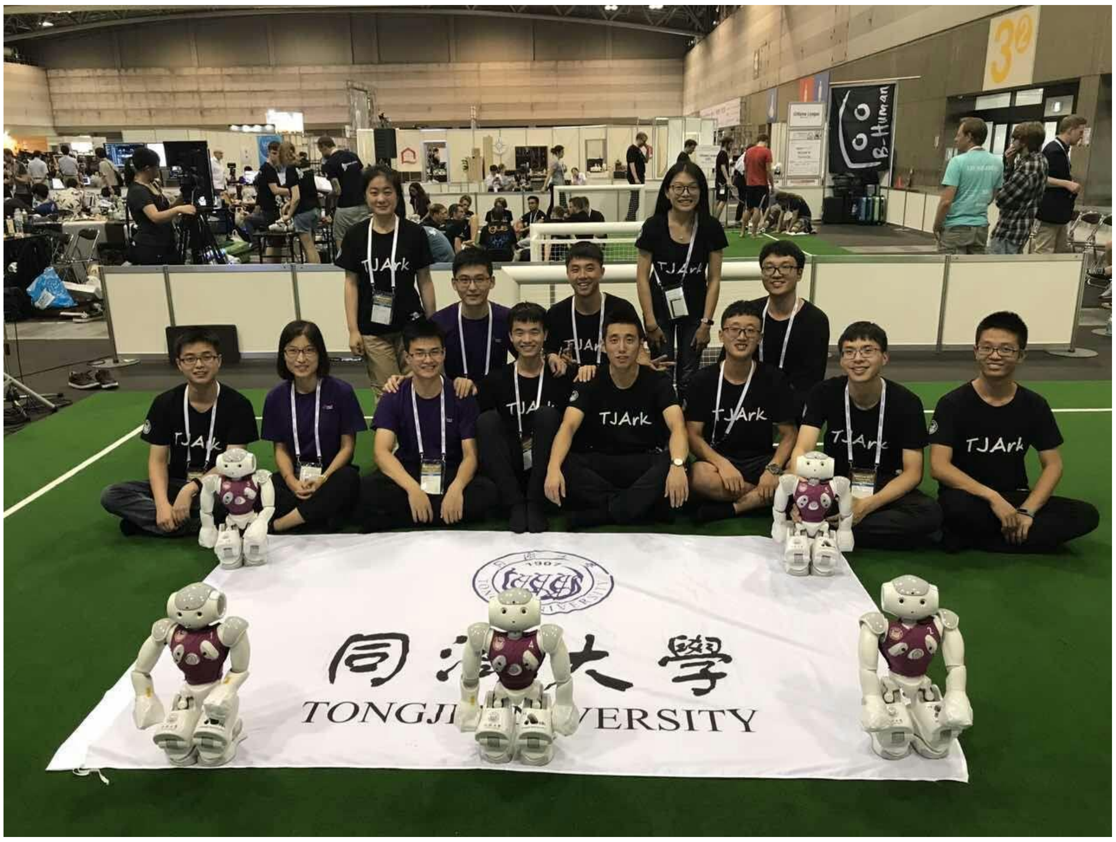
Chen Hao, Control Science and Control Engineering B.S Student.

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

Sun Xiaoxian, Control Science and Control Engineering B.S Student.

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

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



1.3 Robot information

Our team currently have 6 H25 NAO v5s, which is totally enough for us.

2. Code Usage and Own Contribution

TJArk has been participating in RoboCup SPL since 2006, and since that we keep our own code base. But since 2013, we have been using the B-Human framework of Code Release 2013 including the walking engine, kick engine, debugging tools, CABSL and simRobot, according to the license. We develop other modules from our own framework. It worked out not bad. We'd like to thank B-Human for their devotion for SPL. Our walking engine used in RoboCup 2016 and RoboCup 2017 is based on rUNSWift-2014-release, and we also would like to thank UNSW Australia for their devotion.

However, the modules listed below are mainly original developed by TJArk:

1. Vision system
2. Localization system
3. Locomotion system*
4. Whistle recognizer
5. Path planning module
6. Behavior System (with the help of CABSL developed by B-Human)

* We have been developing our own locomotion system since 2016. And now, we already have our own locomotion system, but unfortunately, we were unable to make our walking engine work well as we expected before RoboCup 2017, so we had used the walk engine based on rUNSWift-2014-release again as we did in RoboCup 2016. In RoboCup 2018, we intend to use our own walk engine if it is fully developed and debugged.

The modules above will be further described in following sections. Although some of these modules work not that excellent, we are still using them and developing them. By the gain of game experience and research ability, we believe we can build a much stronger system and finally re-build our own code base.

3. 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/coderelease_2017/TJArkTeamResearchRep-ort2017.pdf

Code: https://github.com/TJArk-Robotics/coderelease_2017

4. Publications

In this section, we list some of our notable work on NAO robot and research paper published from 2016 to 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] Liu Chengju, Xu Tao, Wang Danwei, Chen Qijun. Active balance of humanoids with foot positioning compensation and non-parametric adaptation. Robotics and Autonomous Systems, 2016.1.01, 75: 297~309.

[4] Chengju Liu, Danwei Wang, Erik David Goodman, Qijun Chen. Adaptive walking control of biped robots using online trajectory generation method based on neural oscillators. Journal of Bionic Engineering, 2016.10.01, 13(4): 572~584.

[5] 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

[6] 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.

[7] Liu Chengju, Yang Jing, Bu Wanghui, Chen Qijun. A trajectory generation method for biped walking based on neural oscillators. 13th IEEE International Conference on Networking, Sensing, and Control， Mexico City, Mexico， 2016.4.28-2016.4.30.

[8] Liu Chengju, Yang Jing, Ning Jing. Locomotion control of seven-link robot with CPG-ZMP. 35th Chinese Control Conference, Chengdu, 2016.7.26-2016.7.30.

[9] Wang Helin, Liu Chengju, Chen Qijun. Omnidirectional walking based on preview control for biped robots. IEEE International Conference on Robotics and Biomimetics (Robio) , Qingdao, China, 2016.12.3-2016.12.7.

[10] 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 (in proceeding)

[11] 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)

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

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

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

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

5. Vision System

In this part, we will describe some sub-modules in our 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.

5.1 Field color detector based on GMM

Our paper that describe the algorithm and detailed procedure has been published in 2017 IEEE Conference on Real-time Computing and Robotics, and will be indexed by EI soon.

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 using Gaussian Mixture Model (GMM) to segment every frame of images. Here are some results.

a. Normal situation:

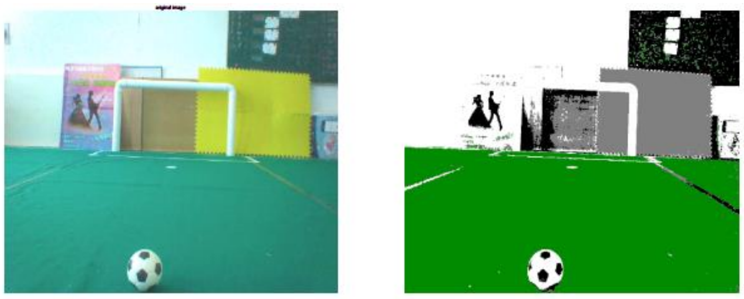


Fig. 5.1 field color detector on normal situation

b. Small field area:

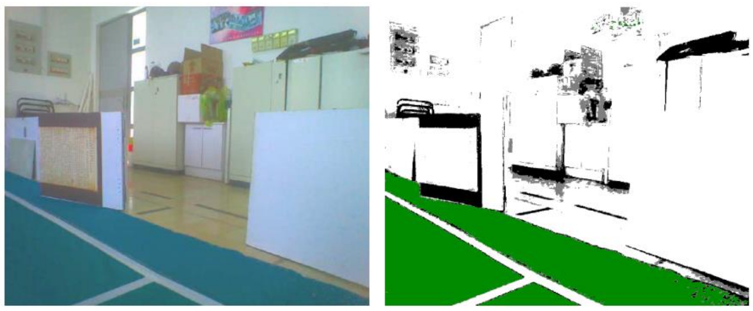


Fig. 5.2 field color detector on small field area

c. Severe uneven light:

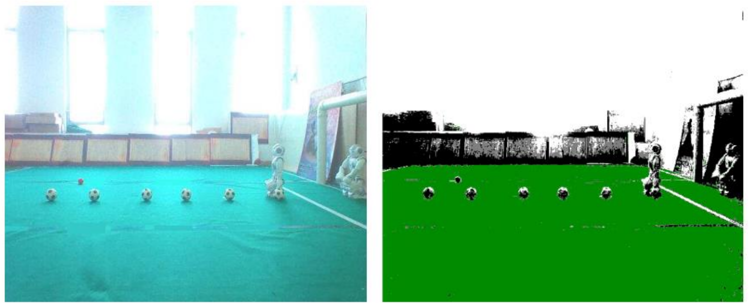


Fig. 5.3 field color detector under severe uneven light

d. Dark image and uneven light:

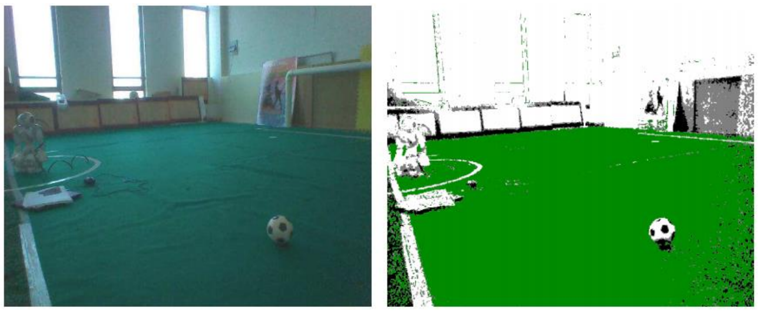


Fig. 5.4 field color detector on dark image with uneven light

e. Uneven light and bright spots:

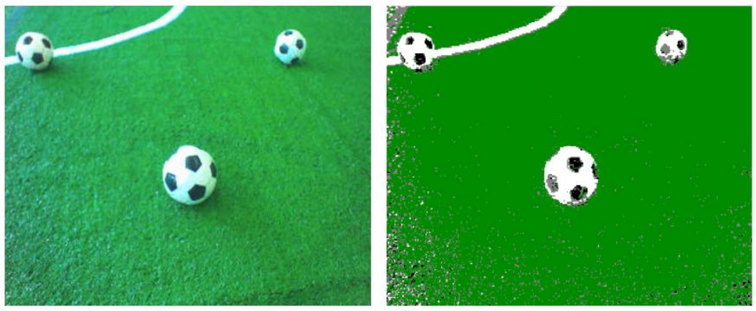


Fig. 5.5 field color detector under uneven light and bright spots

f. Two kinds of carpets (seems unlikely to happen in RoboCup):

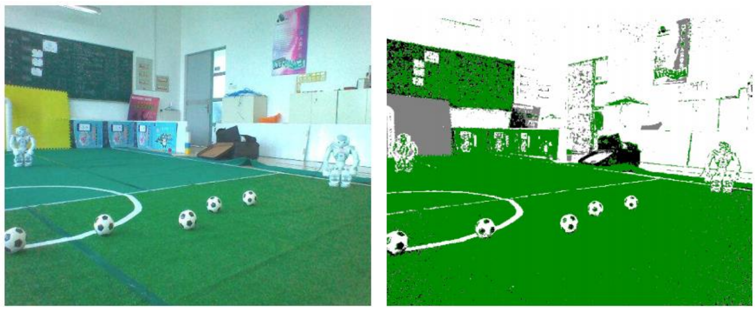


Fig. 5.6 field color detector on two kinds of carpets

Mail processing steps are described below:

a. Convert every frame of image to HSV color Space (Fig. 5.7 Image of Field).

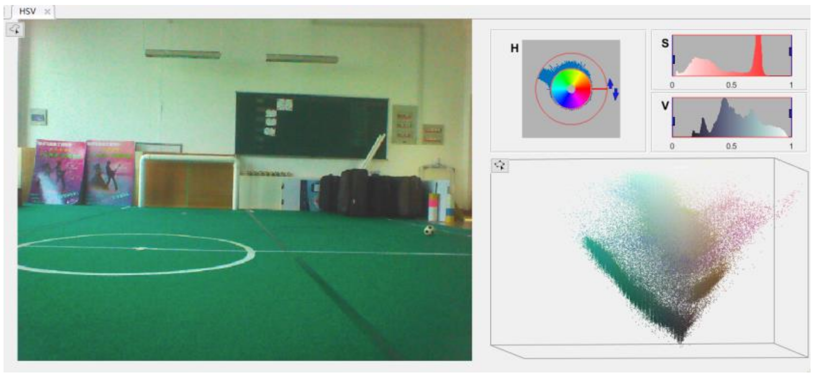


Fig. 5.7 Image of field

b. Find histogram of S channel (Fig. 5.8 Typical S channel histogram)

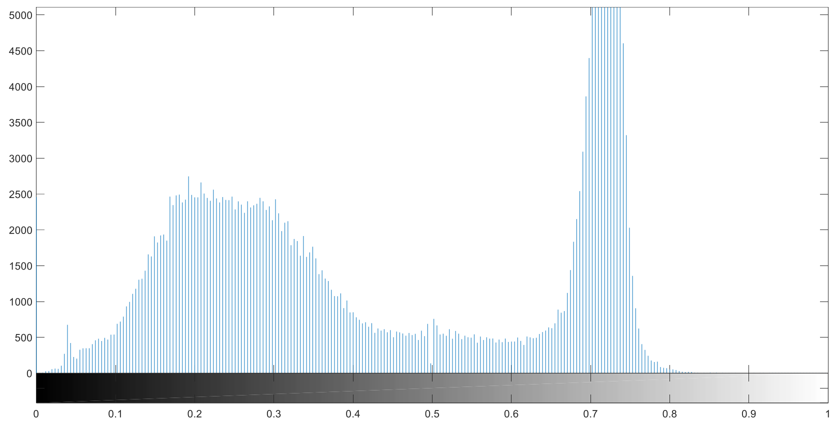


Fig.5.8 Typical S channel histogram

c. Using GMM algorithm fit the histogram (Fig 5.9. Fit with GMM (K = 3))

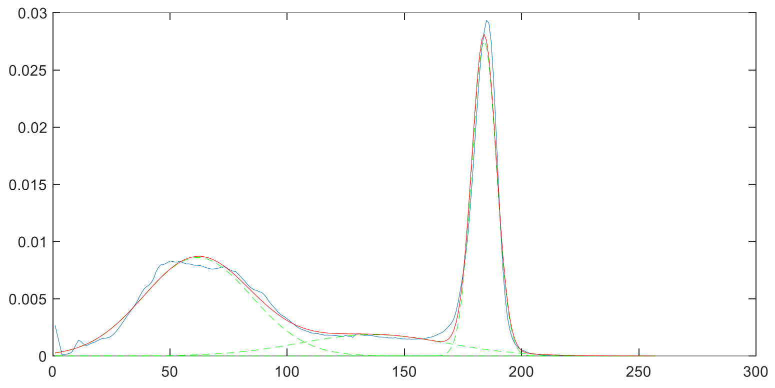


Fig. 5.9 Fit with GMM (K = 3)

d. As we know, for green color, S-channel should be greater than 100, so we choose the wave with a peak greater than 100 as potential green seeds. (Fig 4. Potential Green Seeds)

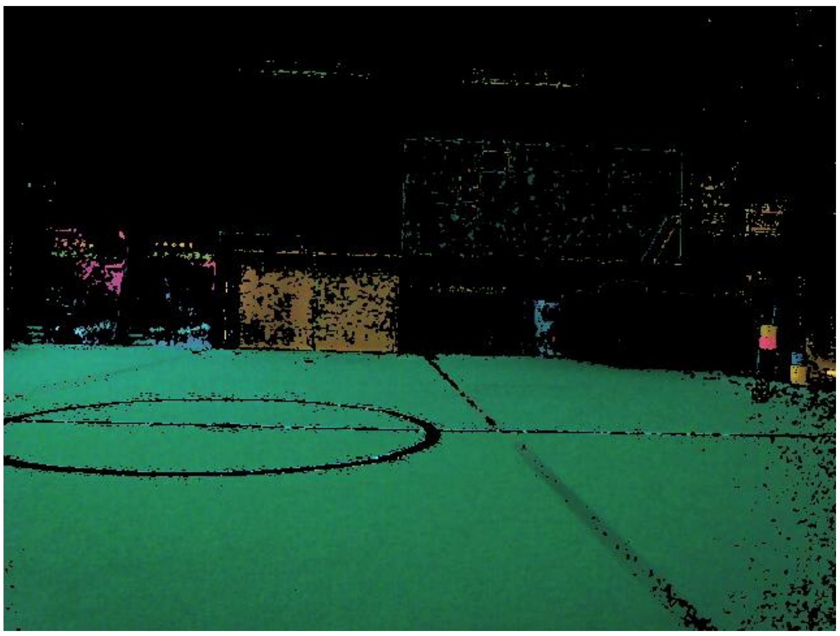


Fig 5.9. Potential Green Seeds

e. For potential green seeds, we fit its H-channel with GMM. (Fig. 5.10 Fit H-channel with GMM)

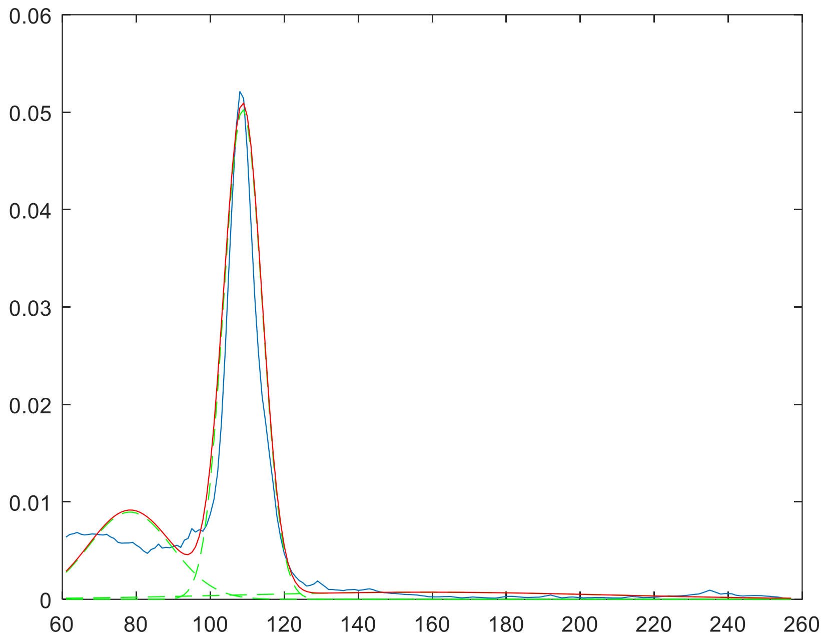


Fig. 5.10 Fit H-channel with GMM

f. We choose the wave with the highest peak as green seeds, and we get green threshold in HSV color space. (Fig. 5.11 Green parts on field)

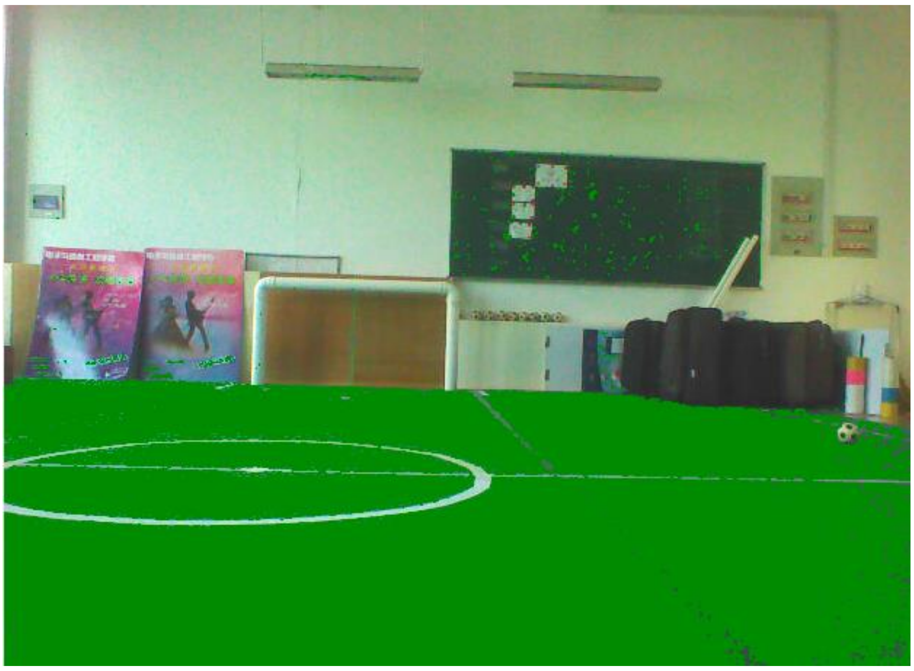


Fig. 5.11 Green parts on field

5.2 Field border detector

Field border is important in perception, because all the objects we interested in are on field. Our implementation of field border detection is based on RANSAC. Here are some of the results:

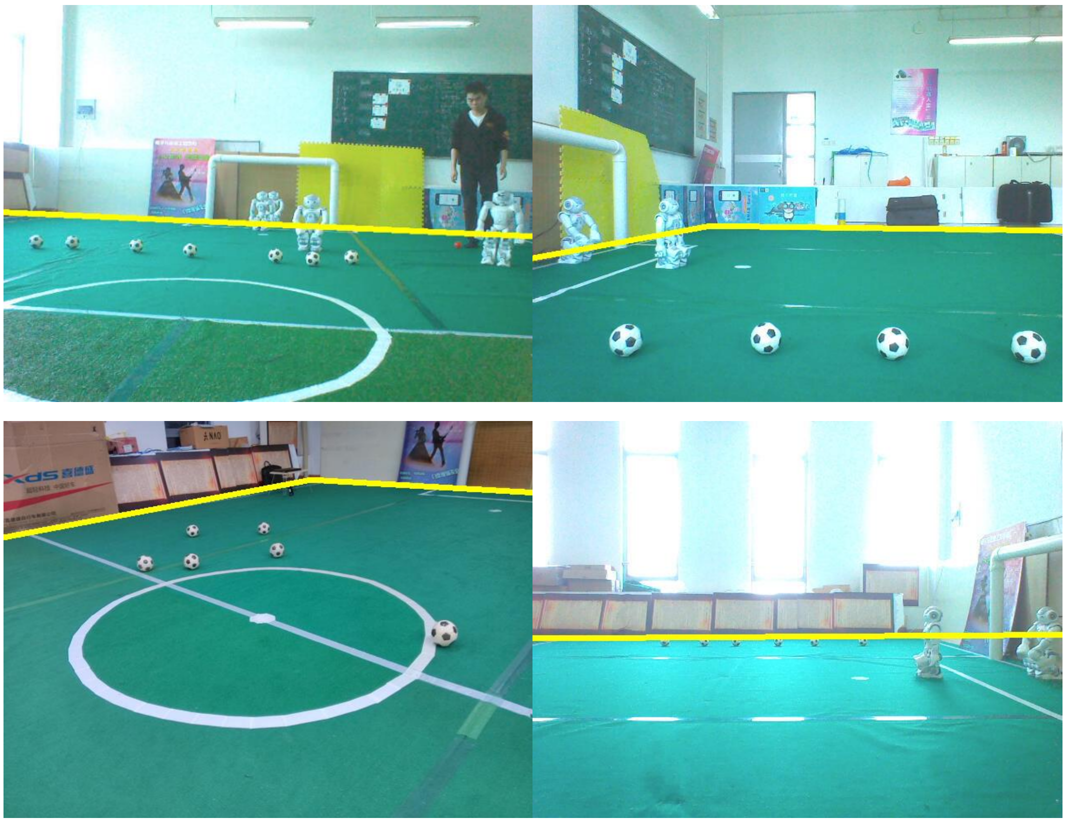


Fig. 5.12 field border detect results

5.3 Line detection and center circle detector

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

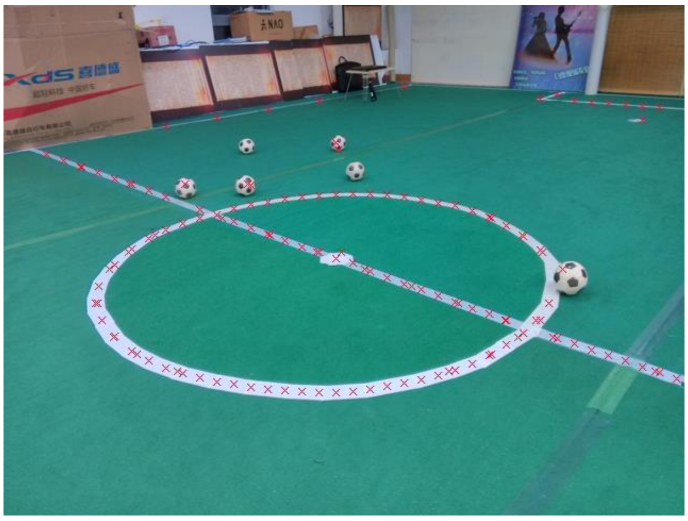


Fig. 5.13 line spots after scan lines

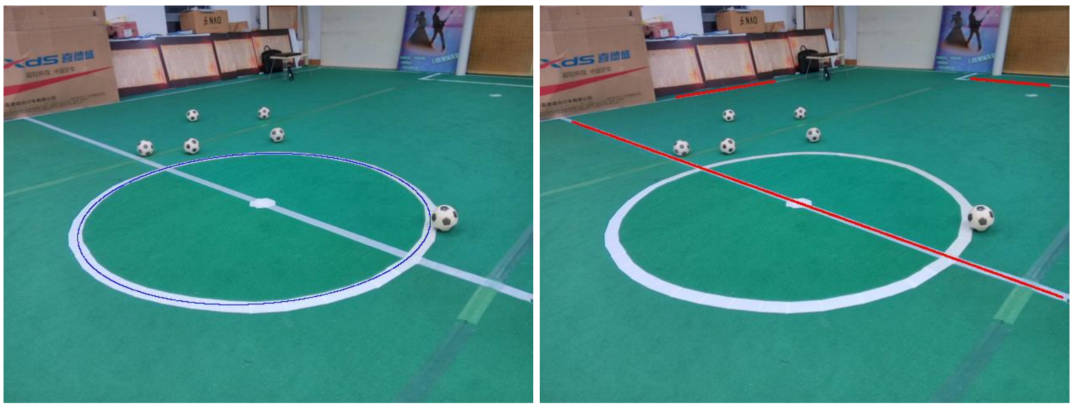


Fig. 5.14 line and circle fitting with RANSAC

5.4 black and white ball

You can find our ballPerceptor directly in our CodeRelease.(https://github.com/TJArk-Robotics/coderelease_2016).

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:

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:

First step:

choose three points called guesspoint inside the circle whose center is ballspot and radius is calculated in BallSpotProvider. 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 figures (Fig. 5.15 Search for guess points):

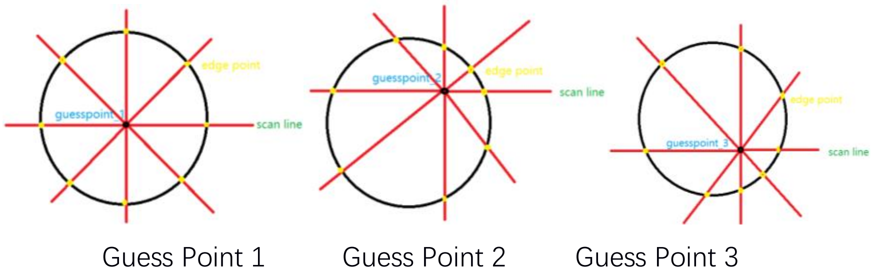


Fig. 5.15 Search for guess points

The edges found is as follows(which are yellow) (Fig. 5.16 Search for Edges):

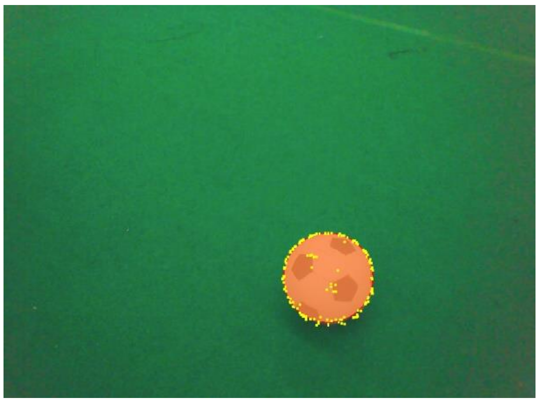


Fig. 5.16 Search for Edges

Second step:

After the first step, we get 24 edge points of a ballspot. In the second step we use these edge points to try to fit a circle. We use RANSAC algorithm to fit them. Since the method of least squares can be influenced by noise easily and the edge points found in the first step sometimes contain a few noise point, the RANSAC algorithm can get a better performance than the least squares method. Through these two steps descripted above, we get some possible ball. Then we re-calculate their center and radius and transfer them to classifyBalls2().

classifyBalls2()

This function is in charge of deciding whether the possible ball found by fitball() is valid. If valid, save it as a candidate. In the end, it 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 an valid ball. All the statistical data are listed as follow:

(1) 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.
(2) ratioTotal: the ratio of black or white pixels among the totality of pixels in the circle.
(3) 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.
(4) whitePercent: the proportion of white pixels in the circle.
(5) ratio: the ratio between black and white pixels in the circle.
(6) meanWhite: the average gray scale of white pixels in the circle.
(7) meanBlack: the average gray scale of black pixels in the circle
(8) Score: the score of possibleball can be achieved by :

$$Score = tansig(ratio, 0.3) \times 0.2$$
$$+tansig(ratioTotal, 1.2) \times 0.4$$
$$+tansig(ratioGreen, 0.7) \times 0.4$$

The tansig function can be seen in the BallPerceptor.cpp.

It is only confirmed to be a valid ball if all the 7 indexes above meet the requirements and the score exceeds the threshold. Finally, if no valid ball is confirmed, we can simply comprehend that as there's no ball in the field of vision. 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 is as follows (Fig.5.17 Perception Performance):

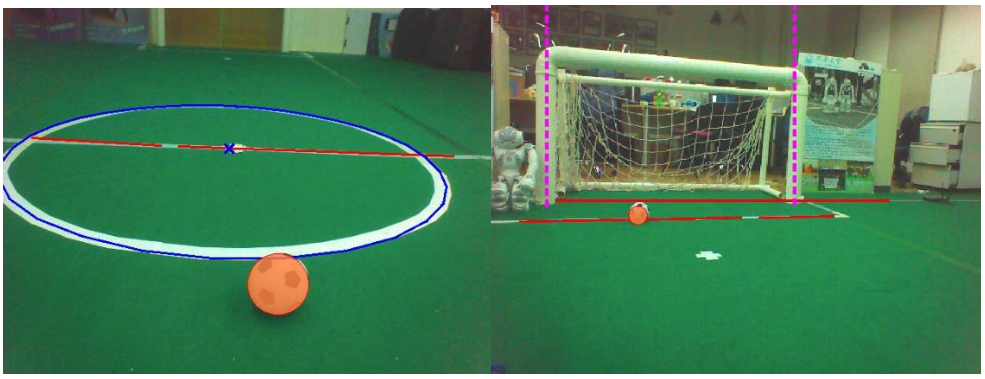


Fig.5.17 Perception Performance

5.5 Background perception

Since the goalposts at either end of the field are made the same color, a robot is not able to resolve one end of the field form the other only by the observation of field features. A robot may suffer from localization mirror when it is lost after kidnapping or a complicated fall. In RoboCup2016, the robot in our team suffer from this situation for several time which made us lost some points in several games.

In order to avoid these problem, we use the background of each goalpost as localization landmarks to distinct the goalposts of each side. We use the SURF (Speeded Up Robust Features) to extract the features of background behind each goalposts. However, the computation resource of Nao is so limited that the SURF algorithm can not be implemented in real time, we simplify the SURF to 1D-SURF which applied SURF to a single row of grey-scale pixels captured at the robot's horizon. The input to the 1D-SURF algorithm is the sum over a band of 30 vertical rows at the robot's horizon as follows:

$$I(i)=\sum_{j=horizon-15}^{horizon+15} I(i,j), i=1,2.....width \qquad (1)$$

Then, the algorithm searches for the local maximum response in the single space dimensional, in other words, a pixel is selected as a maximum response if it is greater than the pixels around it in the same scale. This relaxation ensures that sufficient feature points will be detected. The descriptor of 1D-SURF features is illustrated as Fig. 5.18. The orientation of the 1D-SURF is no longer necessary and can be disregarded. It divides the 15 pixels around the feature point into 3 sub-regions and add up the Harr response in these sub-regions as the feature descriptor, so that the feature descriptor is a 6D vector.

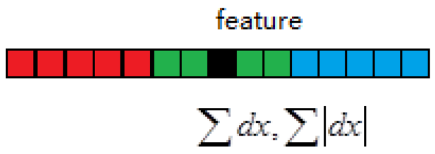


Fig. 5.18 Descriptor of 1D-SURF

After extracting the 1D-SURF features, we can match the features from two different images. We adopt nearest neighbor with RANSAC as the matching algorithm, which makes the algorithm more robust. The matching result of this algorithm is showed as Fig.5.19:

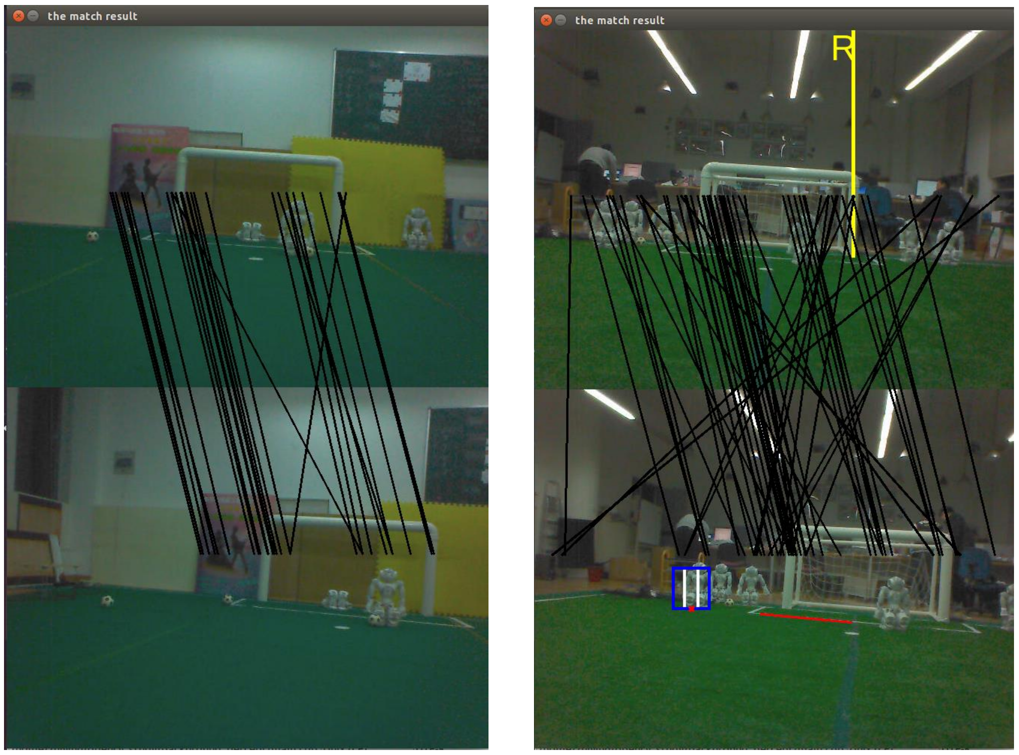


Fig. 5.19 Matching result of same background

With the introduction of 1D-SURF, we extract the features from the background of each goalposts and store them in the memory at the beginning of every games. Then, the robots are able to distinct the goalpost of us and opponent by matching the features in memory with the image captured during the game. Thanks to this algorithm, our robot rarely suffers from localization mirror situation in RoboCup2017.

6. Locomotion System

6.1 Walk control system

Since 2016, we have formed a research group, whose main work is robust and fast walking on biped robots. And we have published several papers related to biped walking on NAO robots and these papers have been indexed by EI/SCI.

You can find all these papers from Publications section and Reference section.

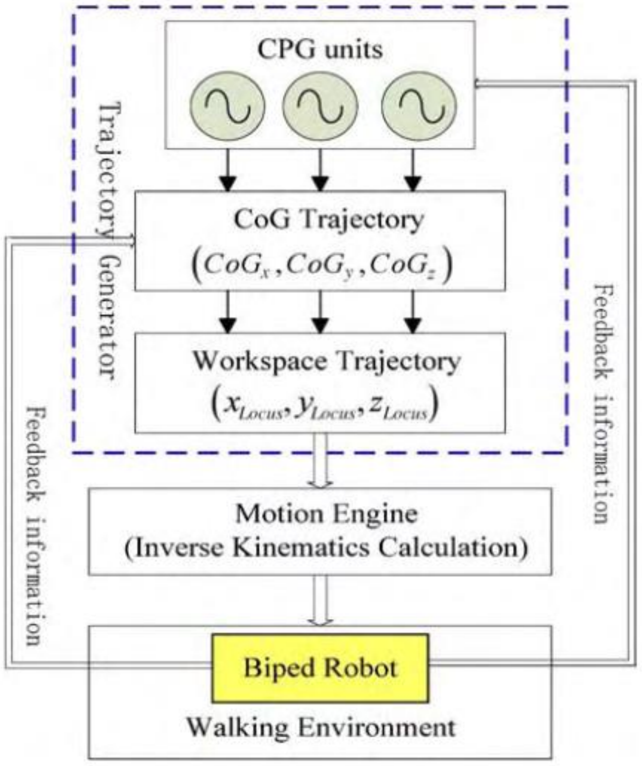


Fig 6.1 Architecture of biped walking control system

The architecture of our walking system is shown in Fig5.1, and the trajectory of a leg and a swing foot in shown in Fig5.2 and Fig5.3, respectively. In order to achieve the maneuverability, we abort our former model which results in an apparent delay when receiving a parameter alter command and use a new omni-directional foot placing scheme. For the swing foot, three parameters forward, left and turn are used to define the next foot's position. This scheme allows us to change either the speed or the direction in one step. Besides we design a parabolic trajectory to make the movement of the swing foot in the air as smooth as possible. We regard the two legs for the robot as an integral mechanism and calculate the joint values altogether using the commonly used DH parameters method in solving inverse kinematics problems. In our case the supporting foot is set as the basic link and the swing foot the end effector.

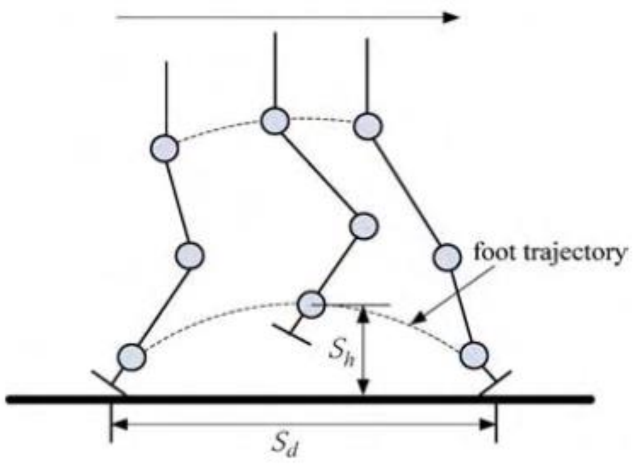


Fig. 6.2 Trajectory of a leg during walking

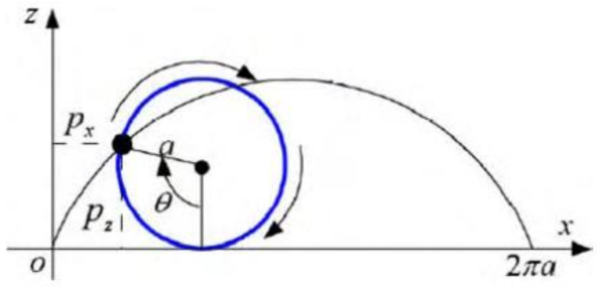


Fig. 6.3 Trajectory of a swing foot

The preliminary open-loop walking model can be achieved through the above description. We then further study an effective control policy for the purpose of a stronger stability. We borrow the idea of applying a control variable on the ankle joint [3]. The main thought is that thinking an ideal inverted pendulum, we could always make the center of mass remain over the origin by adding a force to the origin. In this way the COM will continuously move at a stable speed. In our case, we control the ankle angle of the supporting foot in order to control the COM remaining in a conservative range above the ankle (considering in the 2-dimentional sagittal plane and coronal plane separately) so that the robot will obtain a relatively small acceleration in the horizontal direction. This basic policy results in an apparently better performance comparing to the open-loop one.

In order to get a more stable effect, we use the kalman filter to make the sensor data more stable (including the gyroscope data and accelerometer data), and then these data are used to control the ankle joint in the sagittal plane and coronal plane separately. As a result, we make apparent progress in walking stability. What's more, we have also implement an online foot position compensator (FPC) to improve the robustness of the biped walking.

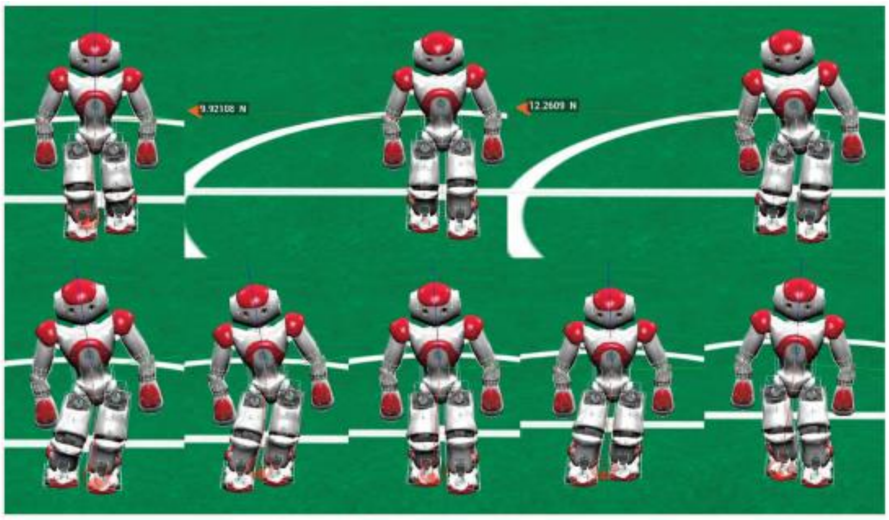


Fig. 6.4 Snapshot sequences of NAO recovering from lateral push

In addition, we design a dynamic model to control the COM in order to reduce the wobble level when the robot have a sharp acceleration or deceleration (especially in the start walking phase and stop walking phase).

6.2 Kicking actions

In order to meet the requirements of behavior control, we design a variety of kicks. These kicking motions have different objectives. On the one hand, the ball must be kicked as precise as possible with an adjustable power and omni directions. On the other hand, the kicking motion should be as fast as possible to avoid unnecessary delays within a normal walk cycle. With these kicks, we have a good performance in Kick-Off, Dribble, Penalty-Kick and so on.

A. 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.

Our current implementation is based on the Kick Engine of B-Human (thanks for their contribution to SPL). The kick process are divided into several phases (see Fig. 5.5)

- lean phase - transfer the robots weight to the support foot
- prepare phase - move the foot to a point behind the ball
- kick phase - a fast straight motion towards the ball
- prepare phase - back to the position after lean
- lean back phase – back to the original robots position

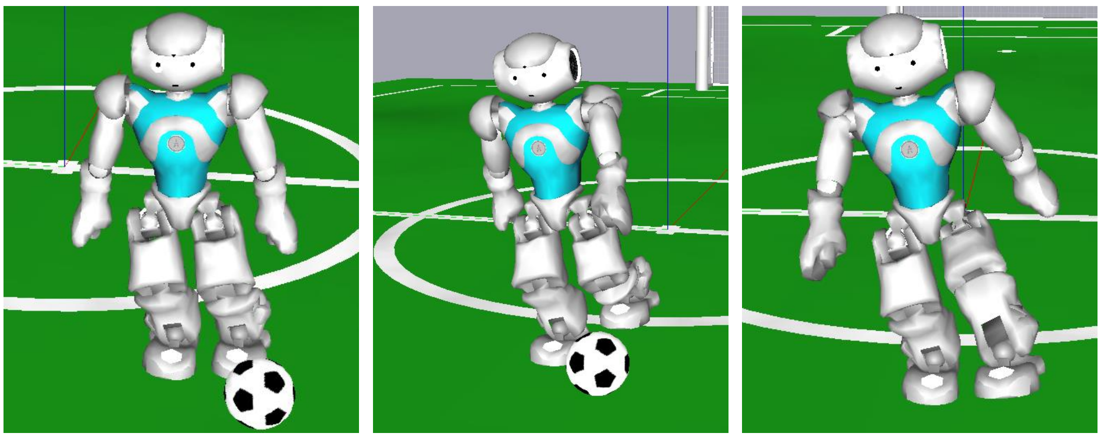

a) lean phase b) prepare phase c) kick phase

Fig.6.5 The robot’s three main phases of the dynamic kick

The separation into these five phases allow a parameterization of the kick which can be adjusted and optimized easily. During the prepare phase, we can adjust the distance and direction of kick swing foot according to the desired power and direction.

B. **Flash Kick**

Beside kicking with adjustable power and omni directions, it is sometimes important to kick as fast as possible. An illustrative situation is when opponent robots are nearby and contesting the possession of the ball. In such situation, speed is of the essence, as the first robot to nudge the ball away will typically gain the advantage. For this reason, another kicking motion exists, the “Flash Kick”.

This kicking motion is not separated from the walking engine like the normal kick. It is divided into three phases. During the first phase, the robot’s weight is transferred to the support leg, while the lean angle is smaller than Dynamic kick. During the second phase, the swing foot kicks straight forward quickly to push the ball and in the last phase, the foot is placed on the position for the next walk step. This three phases therefore replaces the normal movement of the foot during a single support phase. The visualization of the foot trajectory kicking a ball is shown in Fig. 5.7.

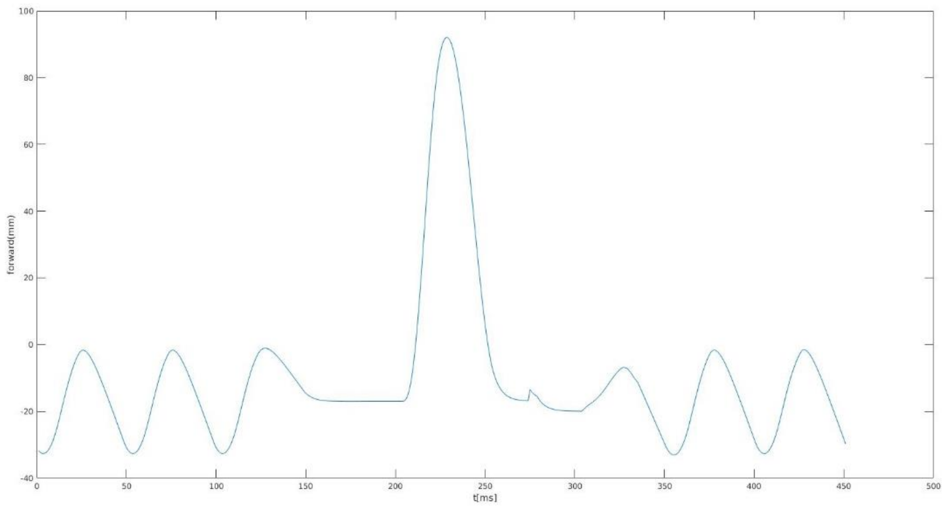


Fig.6.7 Visualization of the foot trajectory while flash kick

7. 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. We lost several points in RoboCup2016 because of this situation.

In order to fix this problem, we combine the ICP (Iterative Closest Point) algorithm with our localization system [2]. In the each iteration of the ICP algorithm, it need to minimize the mean squared position error e, which is :

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

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 $\sin(\delta\theta)=\delta\theta$ and $\cos(\delta\theta)=1$. Equating the first order partial derivatives of e with respect to $\delta\theta$, t_x, and t_y to zero, we can get the following matrix for each point pair q_i and p_i :

$$\begin{pmatrix} 2w_i & 0 & -2w_i p_{i,y} \\ 0 & 2w_i & 2w_i p_{i,x} \\ -2w_i p_{i,y} & 2w_i p_{i,x} & 2w_i p_{i,y}^2+2w_i p_{i,x}^2 \end{pmatrix}\begin{pmatrix} t_x \\ t_y \\ \delta\theta \end{pmatrix}=\begin{pmatrix} 2w_i q_{i,x}-2w_i p_{i,x} \\ 2w_i q_{i,y}-2w_i p_{i,y} \\ -2w_i p_{i,y}(q_{i,x}-p_{i,x})+2w_i p_{i,x}(q_{i,y}-p_{i,y}) \end{pmatrix} \qquad (3)$$

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:

$$\begin{pmatrix} x_k \\ y_k \end{pmatrix}=R(\delta\theta)\begin{pmatrix} x_{k-1} \\ y_{k-1} \end{pmatrix}+\begin{pmatrix} t_x \\ t_y \end{pmatrix} \qquad (4)$$

$$\theta_k=\theta_{k-1}+\delta\theta \qquad (5)$$

Overall, the steps of our ICP algorithm include :

Step1: 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 (2) and update the estimated robot pose according to equation (3) and (4).

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 improved so much in RoboCup2017.

8. Path planning module

In order to solve the problem of the robot colliding in the competition, we have designed a path planner based on an improved rapidly-exploring random tree (RRT) algorithm.

Because of the strong randomness and long path length of the basic RRT algorithm, inspired by artificial potential field, we add a gravity component in RRT construction. In addition, we select the target node by a probability model and add a smooth processing. The procedure is shown below:

a. Build a bidirectional extended random tree (see Tab. 1)
b. Select path cache point (see Tab. 2)
c. Add gravitation and smooth the path
d. Set proper speed along the path

Tab.1 Pseudo code of the bidirectional extended random tree

```
doubleTreeMain()
1   firstTree.init($x_{\text{init}}$);
2   secondTree.init($x_{\text{goal}}$);
3   useFirstTree = true;
4   while searchCount < MAXCOUNT
5       currentTree = getCurrentTree(useFirstTree);
6       theOtherTree = getTheOtherTree(useFirstTree);
7       grow(currentTree);
8       for each node $p_i \in$ theOtherTree
9           if distance(currentTree.$x_{\text{new}}$, $p_i$) $< \varepsilon_0$
10              connectFlag = true;
11              break;
12          endif
13      endfor
14      if connectFlag == true
15          break;
16      endif
17      useFirstTree =! useFirstTree;
18  endwhile
19  if connectFlag == true
20      completeTree = connectTrees(firstTree,secondTree)
21      searchTreeForPath();
22  endif
```

Tab.2 Pseudo code for selecting path cache point

```
chooseTarget()
1   $p$ = RandomReal in $[0.0, 1.0]$;
2   $i$ = RandomInt in $[0, N-1]$;
3   if $0 < p < p_{\text{goal}}$
4       return $x_{\text{goal}}$;
5   else if $p_{\text{goal}} < p < p_{\text{goal}} + p_{\text{way}}$
6       return wayPointCache[$i$];
7   else if $p_{\text{goal}} + p_{\text{way}} < p < 1$
8       return RandomState();
```

The actual result of our implementation of RRT is shown in Fig. 6.1 and Fig. 6.2.

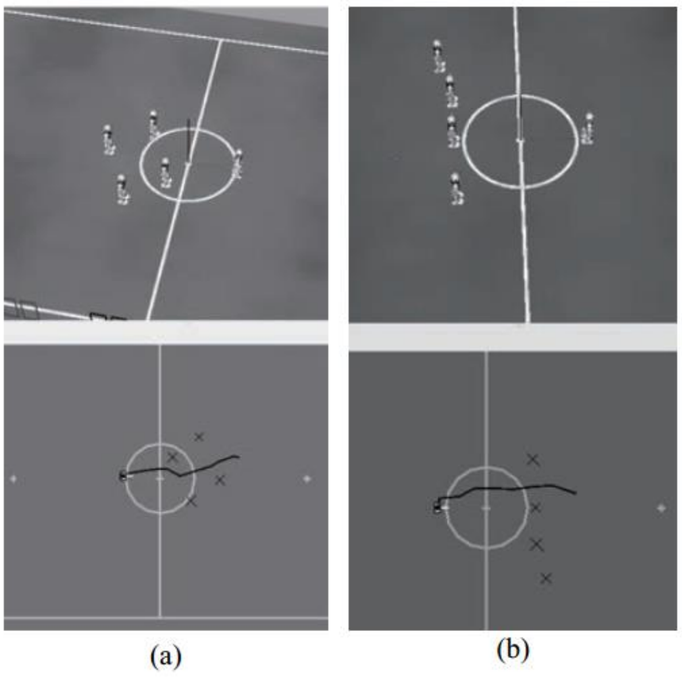


Fig. 8.1 Simulation result of the dynamically path planning in SimRobot

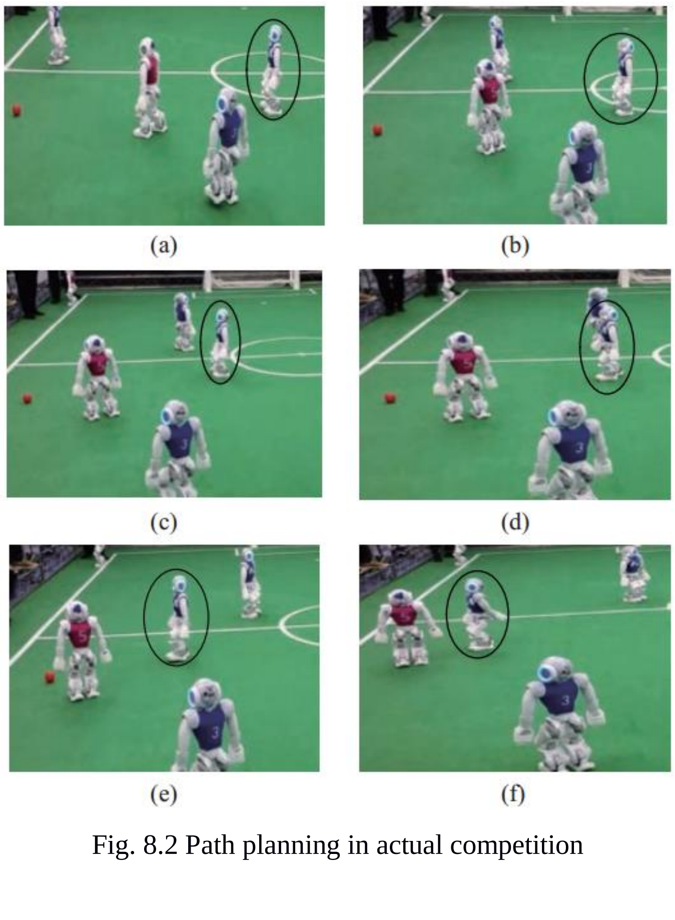


Fig. 8.2 Path planning in actual competition

9. Whistle Recognizer Module

We implement a whistle recognizer based on STFT. The main procedure is as follows. By this module we successfully recognized almost all the whistle signals in RoboCup 2106 and RoboCup 2017.

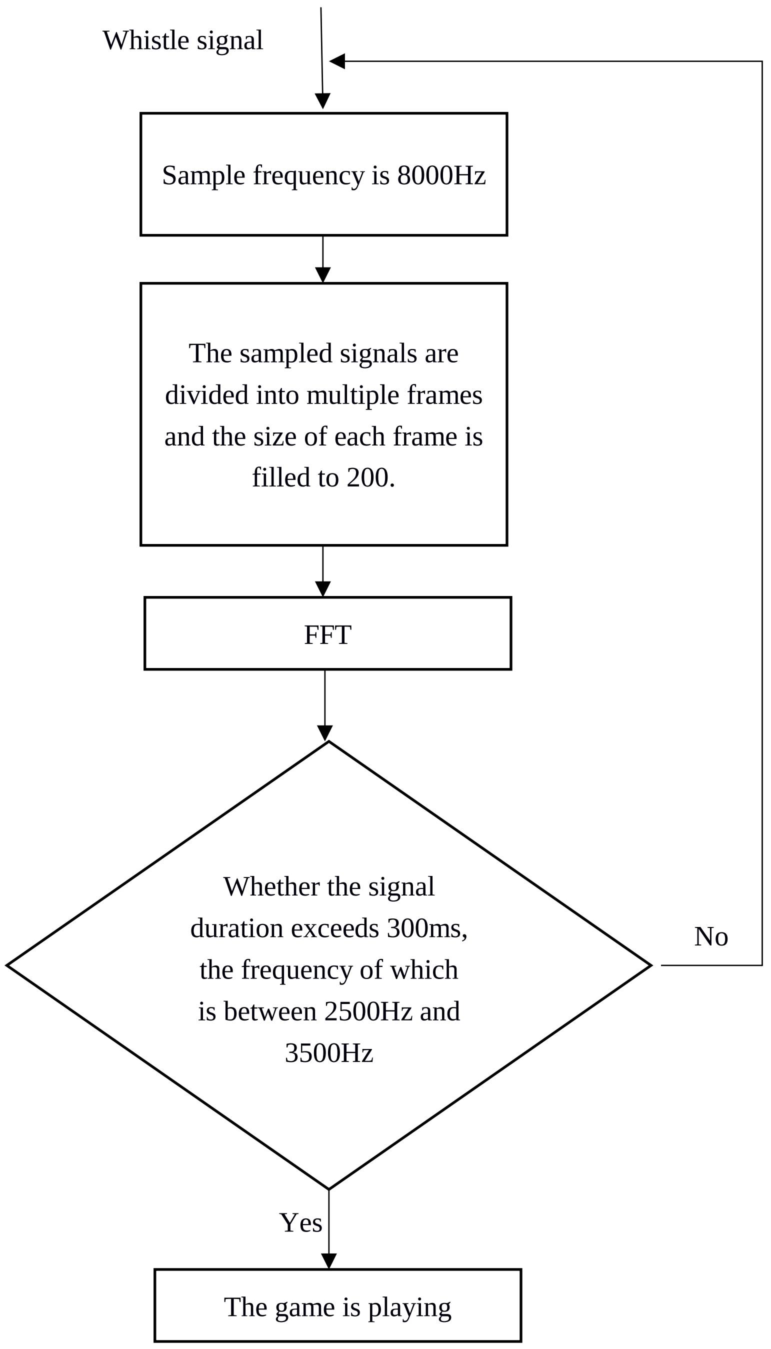

Whistle signal
Sample frequency is 8000Hz
The sampled signals are divided into multiple frames and the size of each frame is filled to 200.
FFT
Whether the signal duration exceeds 300ms, the frequency of which is between 2500Hz and 3500Hz
No
Yes
The game is playing

Fig. 9.1 Procedure of whistle recognizer

10. Research Line and Future Works

During the preparation for RoboCup2018, there are many future works to do for a better competition result.

First of all, we need to improve our strategy. We are researching on how to use deep reinforcement learning algorithm to achieve behavior control. Based on our existing behavior framework, we assign a role to each robot and set some rules to switch role between any two robots (except Keeper). However, we cannot design a policy to meet all situation in competition, so we are going to achieve the role assignment by the robot themselves.

Secondly, we are researching on how to use deep learning algorithm to detect a realistic ball. Although our BallPerceptor module has a low false positive rate, it cannot detect the ball when the background around the ball is totally white. In addition, there is some false recognition when uneven lighting or backlights occurs. In order to fix this problem, we have built an CNN network which consists of a convolution layer and a full connected layer to decide whether the ballspot is actually a real ball. This classifier has a very high accuracy. But it also need a lot of computation resource. We will make some change based on this CNN network to achieve a better BallPerceptor.

Thirdly, we are working on the walking control framework based on the central pattern generator model. At present, the main application of CPG model is to assign CPG units to each degree of freedom of robot, and suppress each other between CPG units to form a distributed CPG network to control all joints of robot to achieve coordinated movement of robot limbs. But it is very difficult for a complex foot robot to generate arbitrary waveform signals with CPG. We use the CPG model to plan the trajectory of the robot's center of gravity (CoG) in real time, and further utilize the generated CoG trajectory to plan the trajectory of the robot's working space online.

References

[1] T. Rofer, T. Laue, J. Muller, A. Burchardt, et al, B-Human Team Report and Code Release 2013, http://www.b-human.de/downloads/publications/2013/CodeRelease2013.pdf

[2] 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.

[3] Aaron James Soon Beng Tay, Walking Nao Omnidirectional Bipedal Locomotion, Bachelor of Science (Computer Science, Honours) August 2009.