TJArk

Team Description Paper & Research Report 2017

TJArk, Robots & Intelligent Systems Laboratory

Tongji University, Shanghai 201804, P.R.China

(TJArk.official@gmail.com)

Dec. 18, 2017

Team Description Paper & Research Report 2017

Abstract. TJArk is Tongji University's RoboCup team. 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. Associate Professor of College of Electronic and Information Engineering;
Prof.Chen. Qijun, Dean of College of Electronic and Information Engineering;
Mr. Li. Shu, Control Science and Control Engineering M.S Student;
Mr. Zeng. Zhiying, Control Science and Control Engineering M.S Student.

University: Tongji University, 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.

1.4 CodeRelease and Research Paper

Since 2013, we have been using the B-Human framework of Code Release 2013 including the walking engine and kick engine, 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 current walking engine is based on rUNSWift-2014-release, and we also would like to thank UNSW Australia for their devotion. Last year, we made progress in vision, location, strategy and motion modules.

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

Documentation:https://github.com/TJArk-Robotics/coderelease_2017/TJArkTeamResearchReport2017.pdf

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

2. Mixed team competition

We are very happy and with great honor to make a joint team with UNSW Australia for the mixed-team-competition next year.

Mixed team name: Swift-Ark

Mixed team jersey color: Home-Yellow, Away-Blue

Infrastructure plans: We will define an interface to share information (robot position, ball position, obstacle position, behavior intention, etc.) and share some basic skills in order to work together in the field.

Reason for cooperation: UNSW Australia and TJArk are both RoboCup teams with a long history, by this cooperation we can learn from each other, strengthen exchanges, discuss game experience and the realization of each module algorithm, so that benefit both teams. We may build up a strong team and bring excited games to the league.

3. Notable work in RoboCup2016

In this part, we will describe the changes we made during the RoboCup2016 championship

competition.

3.1 vision

This year we 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. And it is faster enough for NAO robot at a speed of 2*30fps.

3.1.1 field color detector based on GMM

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 image processing steps:

a. Convert every frame of image to HSV color Space (Fig. 1 Image of Field).

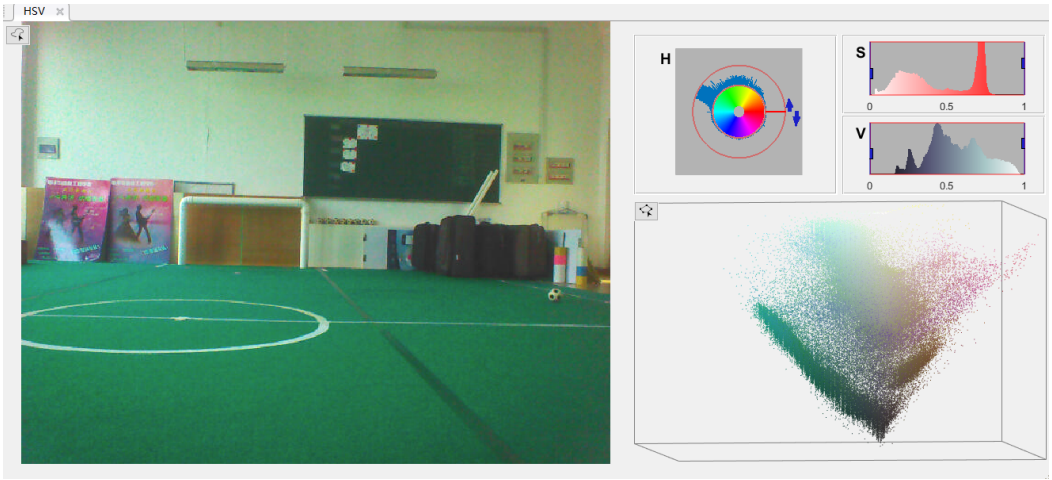

Fig. 1 Image of field

b. Find histogram of S channel (Fig. 2 Typical S channel histogram)

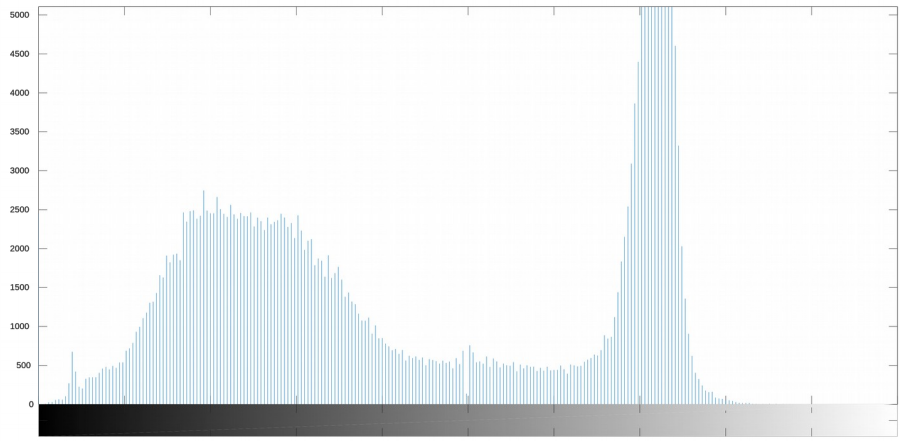

Fig.2 Typical S channel histogram

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

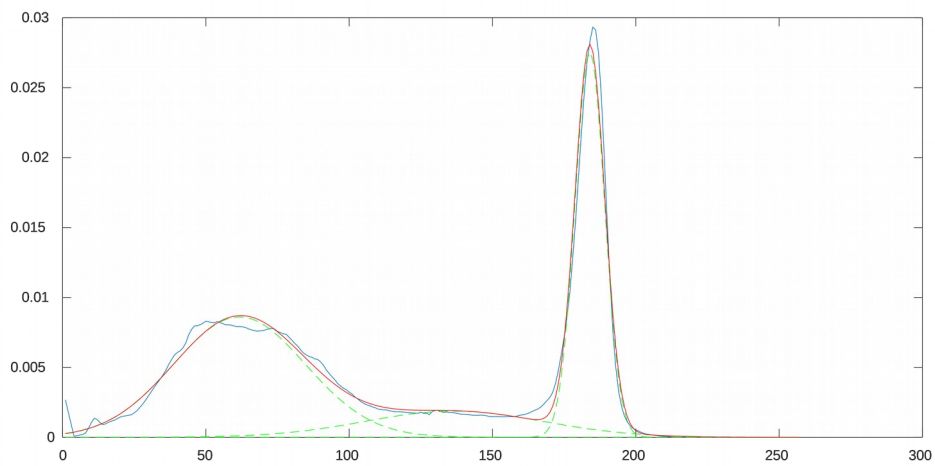

Fig. 3 Fit with GMM (K = 3)

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 4. Potential Green Seeds

e. For potential green seeds, we fit its H-channel with GMM. (Fig. 5 Fit H-channel with GMM)

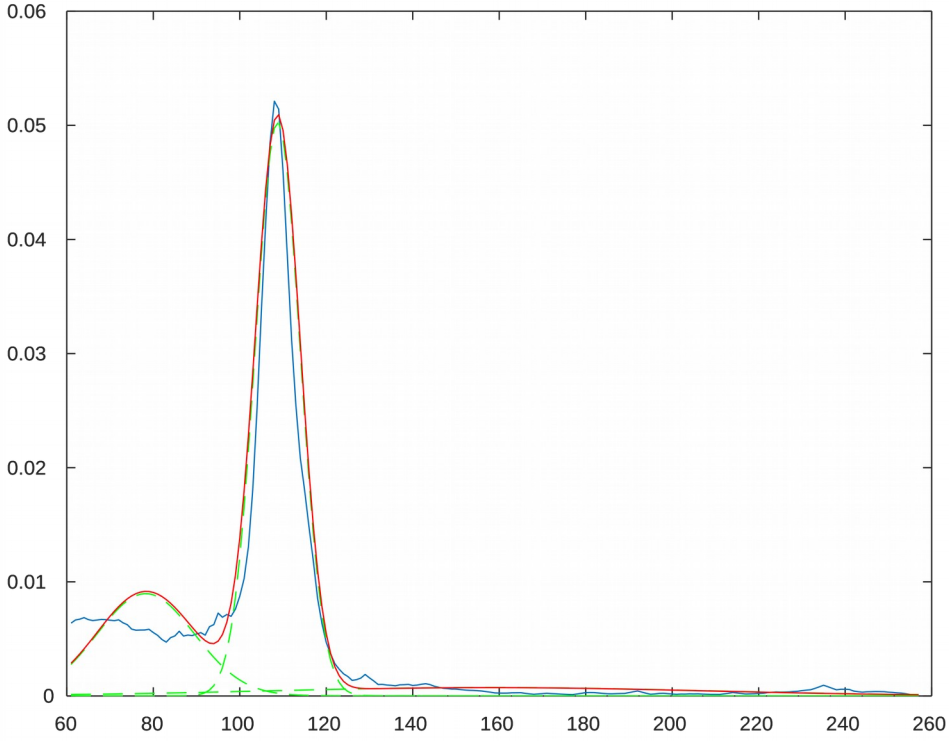

Fig. 5. Fit H-channel with GMM

f. We choose the wave with the highest peak as green seeds, and we get green threshold in HSV color space. (Fig. 6 Green parts on field)

Fig. 6 Green parts on field

3.1.2 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. 7 Search for guess points):

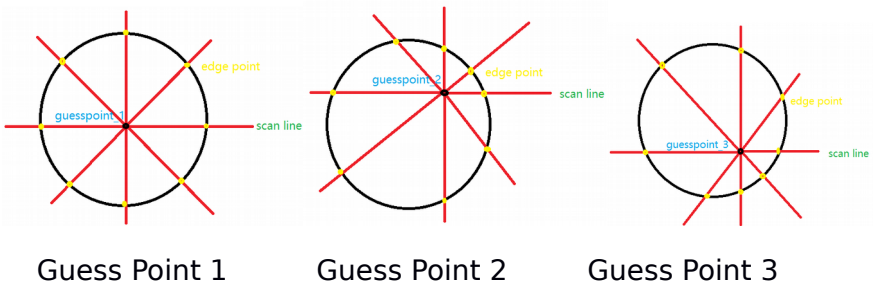

Fig. 7 Search for guess points

The edges found is as follows(which are yellow) (Fig. 8 Search for Edges):

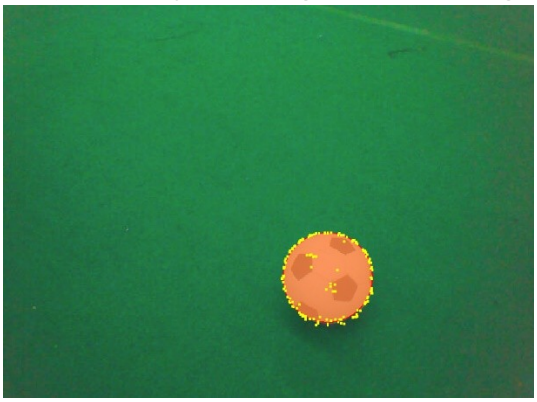

Fig. 8 Search for Edges

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.9 Perception Performance):

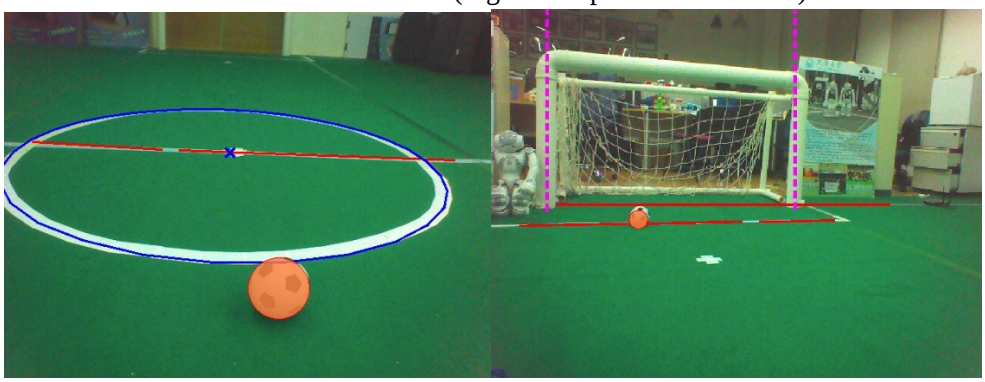

Fig.9 Perception Performance

3.2 motion

3.2.1 walking

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.

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.

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).

3.2.2 kicking

In order to meet the requirements of behavior control, we design a variety of kicks. With these kicks, we have a good performance in Kick-Off, Dribble, Penalty-Kick and so on.

3.3 localization

Our team’s self-localization is based on a particle filter with a low number of particles that each includes an Unscented Kalman Filter (introduced by B-Human, thanks for their contribution to SPL). With a more stable and reliable Vision System developed, our Localization System become more accurate and reliable. But we also find 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. In order to fix this problem, we combine the ICP (Iterative Closest Point) algorithm with our localization system [2]. With this sensor model, 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 by a sensor resetting method. 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.

4. 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 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.

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.