

# The TJArk' Team Description for 2010



Robot & Intelligent System Laboratory  
Tongji University  
Shanghai 201804, P.R.China  
TJArk.official@gmail.com

**Abstract.** TJArk is Tongji University's Legged-league RoboCup team. They took part in RoboCup in 2006 for the first time and entered the semi-finals in RoboCup 2007, 2008. 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 new improvement on each part of TJArk's project.

## 1. Introduction

TJArk was established in 2004 as part of Lab of Robot & Intelligent System of Tongji University. We participated in RoboCup for the first time in 2006. By 2006, research interests of TJArk were focused on color-blob-based object recognition[1] and manual design of trajectories-based locomotion, leaving the behavior and localization modules really fragile. We lost a lot of goals in RoboCup 2006, but managed to the Second Round due to a vital success in the Intermediate Round. TJArk succeeded into the quarter finals in RoboCup 2007. In 2008, we carried out many more scientific research attempts, including automatic color-table generation and adaptation, combination of Monte Carlo Localization and Kalman Filter Localization in the RoboCup self-localization problem, quadruped walking learning and on-line adaptation using walking rhythmic pattern, central pattern generator(CPG) based locomotion etc. We spent a lot of time on these trials, but many of them unfortunately failed and this finally left us little time to prepare for the competition. The result of TJArk in RoboCup 2008 is still entry of quarter finals. In 2009, we shift our research platform from quadruped robot to humanoid robot, and some locomotion and behavior optimization related research. Since we got the robots in late April, this left us only two months to build the framework and carry out trials. In RoboCup 2009, this was our first experience in two-legged competition, and we got the entry of second round. We found out that there's still much to do on our locomotion and vision module, and after the competition, we feel the urgency to add the selflocation module. Detailed information can be found at the team's website: <http://see.tongji.edu.cn/TJArk/English/index.html>.

## 2. Background of TJArk Team

### 2.1 Team Leadership

*Prof. CHEN Qijun* (Team Leader) is now the chancellor professor in the College of Electronic and Information Engineering (CEIE) of Tongji University. He is the Dean of the CEIE of Tongji University. Prof. Chen has been to University of Hagen in Germany in 2002 as a Guest Professor and

UC Berkeley in United States in 2008 as a Visiting Professor. He is the member of National professional Standardization Technical Committee; the Standing Member of Intelligent Manufacturing Committee and the Standing Member of Intelligent Robot Committee of Chinese Artificial Intelligence Association. He had more than 80 papers published in journals and conferences. Prof. Chen's research interests are robotics control, environmental perception and understanding of mobile robot, bio-inspired control and Humanoid Robot, etc.

## 2.2 Team Members

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

- *JIA Lin* (Control Science and Control Engineering graduate)
- *WANG Wenjia* (Control Science and Control Engineering graduate)
- *HU Shiling* (Control Science and Control Engineering graduate)
- *CAI Zhiqiang* (Control Science and Control Engineering graduate)
- *ZHU Zhenjiao* (Control Science and Control Engineering undergraduate)
- *XIE Jing* (Control Science and Control Engineering undergraduate)
- *CHEN Guangmou* (Control Science and Control Engineering undergraduate)
- *WANG Qiang* (Control Science and Control Engineering undergraduate)

## 3. Robot Vision

Our implement of humanoid vision module is based on combining algorithm of RLE and raster-scan feature abstraction. Using the color look up table, we scan the image with variational jumpiness to form some segmentation with the same color, then using run length encoding algorithm to merge the segment into blobs. The blobs can be used in the following vision recognition with high reliability.

### 3.1 Goal Detection

Because sometimes the scan method based on horizon line can cause false perception, we develop our strategy that rarely depends on the horizon line instead of the vertical line to collect more information before deeper recognition. For goal detection, we only adopt horizontal scan, which will save some horizontal run with goal color. Then the horizontal run will be connected to blobs based on their positions. And some supplement scanning is made to decide the goal post blob. After some sanity checks, we can make sure this is a post. When there is only one eligible post left, we use the crossbar and the corner in the forbidden area to decide whether the post is on the right or left. The new method of detecting goal has advanced the efficiency of vision and reduced the possibility of false detection.

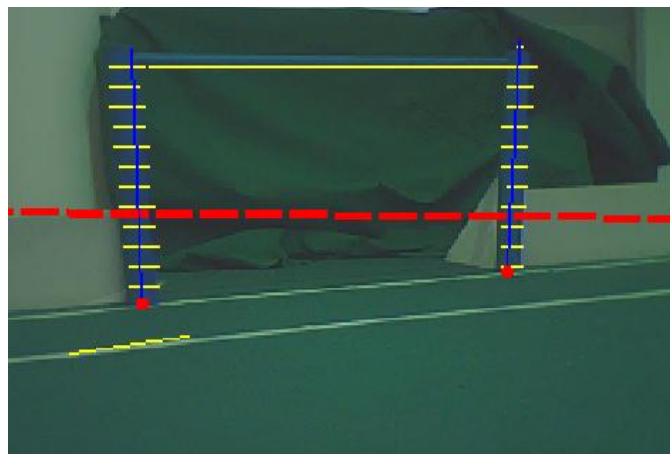


Fig3.1 The result of goal detection(The red dots express that goal cognition is right and return the coordinate of goal in image)

### 3.2 Line detection

In consideration of adding location module to our project, it's highly required to calculate the center of kick-off circle quickly and accurately, formerly we take Hough Transformation into consideration, but the result of experiment depressed us, it's time-consumed. So we replace points by blobs, we group points of the same color into a blob, then we can calculate the characters of these blobs instead of calculating the gradient of the points that is not accurate with non-straight lines, so it's more quickly and accurately to recognize the lines and kick-off circle. Fig 2 is the result of the line detection, the left one is the line and the right one is the center circle, the result is drawn with yellow line in the image.



Fig3.2 the result of the line detection

### 3.3 Tool research

We have improved our color classify tool last year by transferring manually to semi-automatic. Inspired by resample method, we use the run-length encoding algorithm to detect the target region, then fit the sample to the Gaussian model or Histogram model respectively. If the light condition changed, we choose the MAP method to adapt the model to accommodate the current light condition. The method releases us from the process of learning color look up table, and can adjust immediately according to the mutative light.

## 4. Probabilistic State Estimation and ball tracking

Our vision module processes the camera image, do object recognition, and provide the bearings and distance to visual objects in the robot's coordinate system. For a robot soccer game, it is important to know the position of the robot and the location of other objects like the ball on the field is important. Especially in SPL field, when the robot is walking near the goal, the bottom camera is unable to catch the goal in sight. The robot cannot score without knowing where the goal is. Our self-localization and the ball tracking are both based on Monte Carlo Localization with Particle Filters implementation. It's more robust to observation error and systematic error. The belief of the robot is represented using a group of particles.

To perform belief update, we construct a motion model and an observation model[2]. The motion models use the calculated odometry to predict the probability distribution of robot pose compared to last frame. In observation update, we use all the objects perceived, like the goalposts, the centre circle, lines and the intersections, to correct the belief. The history information of unique observation information, such as the goalpost and center circle, will be stored. The latest stored information will help to generate new particles. Visual location produce good result when the sensor information is accurate. Next task is to do some correction of the vision results since they are easily get incorrect when the robot is turning due to the inaccurate camera matrix.

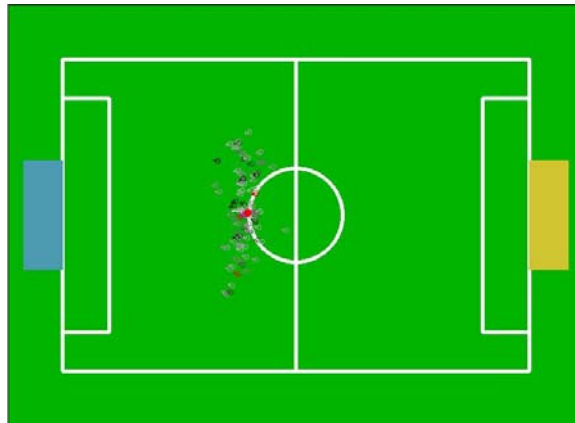


Fig4.1 the particles on field represent the pose of the robot. It generates a final pose estimation which is shown in red.

The new ball is smaller than the former one, which reduce the chance of perceiving a ball. And knowing the velocity and direction of the ball can help the goalie defending the goal. Ball tracking is our major task currently . The method we used on Aibo is Kalman filter, but we shifted to MCL on Nao. The ball tracking produce the position of the ball and its validity. When the validity is high, it may be shared with teammates. Yet this module is under improvement.

## 5. Locomotion

### 5.1 Improvement on Walking Engine

Walking Engine of TJArk is mainly based on the open-loop Cart-Table model[3] and the ZMP preview control paradigm. In the PreBufferGenerator class, we pre-plan for the walking state of the future 80 frames, including supmod, walkStatus, pSwingFoot, pSupFootZMP etc, while it can generate all kinds of walking paths and fulfill the motion requirement of RoboCup soccer game. With a newly written program for preview buffer generator, the robot is able to walk faster and more stably. In our simulation platform, the robot can walk forward with full speed of about 8 cm/s, which is much faster than last year. We also improve the rotation performance. The walking locus of y axis generated when the robot walk forward with 8cm/s and rotate with 12 deg/s is shown in Figure X.

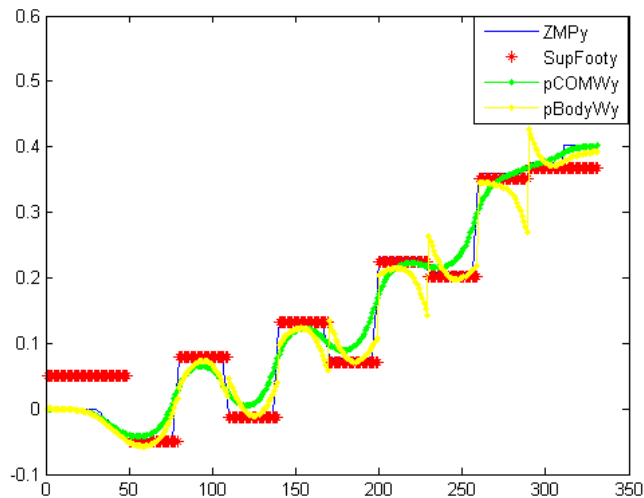


Fig5.1 The walking locus of y axis generated when the robot walk forward and rotate

Moreover, we also do the locomotion research on how to solve the closed-loop walking problem. The resolution now is to adjust foot step to compensate disturbances. When the robot is about to lift its foot, we would calibrate next foot step according to calculating step-error between ZMP measured from LIPM and ZMP pre-planned from PreBufferGenerator. As a result, we can enhance the stability of walking with sensed feedback when the robot is encountered with a small disturbance.

The NAO has plenty of sensors which are useful in monitoring the robot as various feedback signals. Foot Pressure Sensors are used to detect whether the robot is falling down, determine which foot is landing, and calculate center of pressure of the robot. Doing this research is very helpful for us to solve the problem of the gait calibration learning.

## 5.2 Calculation of Body Angles

In the generation of CameraMatrix, we import body angles (roll and tilt) to get the precise camera pose according to the supporting foot.. The calculation of the body angles depends on the the inertial unit inside the chest of Nao, which consists of two axis gyrometers and three accelerometers. Calculating the posture by accumulating the outputs of gyrometers gives good dynamic feature, however the results drift from the true values because of the accumulation. On the other hand, the outputs of accelerometers can only be precise in the static state of the robot, however highly affected by the robot motion and mechanical vibration in walking states. Considering those characteristics of sensors, we decided to combine both information to get accurate sensor data. A complementary filter [4] is introduced here with a high-pass filter to handle the gyrometer data and a low-pass filter to handle the accelerometer data. The results can be plotted via the MATLAB engine in our project just as follows.

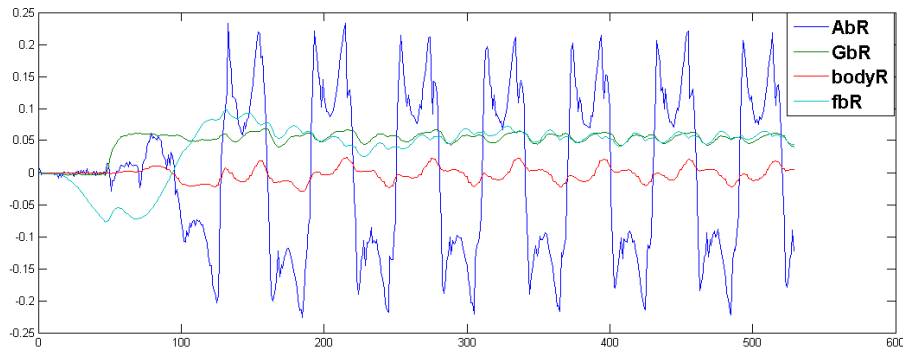


Fig5.2 Output of body roll angle.

## 5.3 Specail Action Debug Tools

### 5.3.1 Sensor Viewer

In order to observe data of sensors in a convenient way, we open up this widget. It can accelerate the debug of parameters. The left widget support the joints data, the other gives other sensors of Nao, it makes the sensors directviewing, easily operated and high in safety.

Fig5.3 Other Sensor Data

Sensor Name	Value	Unit
accX	0	m/s <sup>2</sup>
accY	0	m/s <sup>2</sup>
accZ	0	m/s <sup>2</sup>
angleX	0	rad
angleY	0	rad
compassX	0	m
compassY	0	m
fsrLBL	0	N
fsrLBR	0	N
fsrLFL	0	N
fsrLFR	0	N
fsrRBL	0	N
fsrRBR	0	N
fsrRFL	0	N
fsrRFR	0	N
gyroX	0	rad/s
gyroY	0	rad/s
gyroZ	0	rad/s

Fig5.4 Joints Sensor Data

Sensor Name	Angle	Stiffn
HeadYaw	0	0
HeadPitch	0	0
RShoulderP...	0	0
RShoulderR...	0	0
RElbowYaw	0	0
RElbowRoll	0	0
LShoulderP...	0	0
LShoulderR...	0	0
LElbowYaw	0	0
LElbowRoll	0	0
REHipYawPitc	0	0
REHipRoll	0	0
REHipPitch	0	0
RKneePitch	0	0
RAnklePitch	0	0
RAnkleRoll	0	0
LHipYawPitc	0	0
LHipRoll	0	0

### 5.3.2 Joints Control Panel

The panel gives many useful buttons and sliders, so that we can change the value or the power of any joint easily, it saves much time which we use to make our special action much more better. The left part is the control group, it completes the basic control of the robot and the tool, while the right part is the slider groups for each joint of Nao.

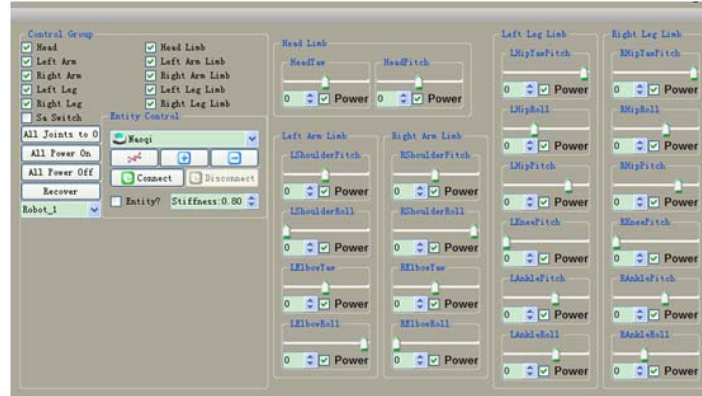


Fig5.5 Joints Control Panel

## 6. Behavior

Our behavior module is programmed in python using an external python interpreter instead of Python Bridge.

### 6.1 Finite state Machine

#### 6.1.1 Define of FSM

This year, TJArk2010's behavior module has huge changes. Based on the last year's behavior module, we add a finite state machine to the Decision-making System which named DFSM. The machine includes nodes(states), edges (from one state to another state) and so on. In this machine, all skills are defined as different states. A state has some attributes such as the current state name, the next jump node, the number of the branches. In the current Decision-making System, three subclasses have inherited the base class DFSM which are GameController class, FallController class and NaoPlayer class.

#### 6.1.2 Styles of FSM

We have two styles of FSM in the system, since there is not just one state sometimes. For example, when the robot sees the ball, there should at least be running the trackBall state and the goToBall state simultaneously. And the trackBall state just relates with whether the robot sees the ball. So we decide to define two styles of FSM. One is that all states participate in state cycle(there is only one state in this style at any time), and the other is that some states do not participate in state cycle, and these states will be pushed in the second state machine when some conditions are met. So at each frame of the system, it will run the current state of the first state machine and the states in the second machine.

#### 6.1.3 Views of FSM

In order to form grapy visualization of state transfer machine, we use yapgvb module of Python which provides Python bindings to Graphviz, with an intuitive Python interface. It is convenient and intuitive for us to check out our states process whether is right or not. A simple example as follows:

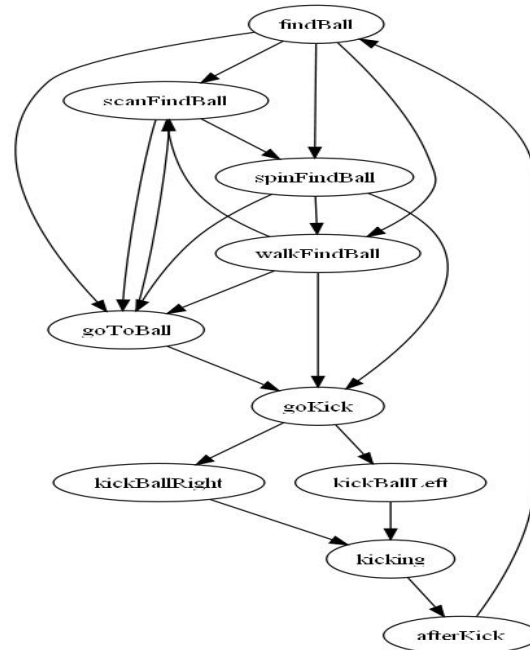


Fig6.1 Grapy of States Machine with yapgvb

## 6.2 other improves of behavior module

After the localization module added into the TJArk2010, the behavior module begins to add teammate information to communicate and role-switching to make decisions. And we also continue to strengthen all base skills to adapt to the dynamic environment.

### References:

- [1] Qijun Chen, Huali Xie, Pengyung Woo. Vision-based Fast Objects Recognition and Distances Calculation of Robots.2005 *IECON*.
- [2] T. Laue and T. R ofer. Particle \_lter-based state estimation in a competitive and uncertain environment. In Proceedings of the 6th International Workshop on Embedded Systems. VAMK, University of Applied Sciences; Vaasa, Finland, 2007.
- [3] Shuuji KAJITA, Fumio KANEHIRO, et al. Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. *ICRA 2003*
- [4] K. Nishiwaki and S. Kagami, "Walking control on uneven terrain with short cycle pattern generation," in Proc. IEEE-RAS Int. Conf. on Humanoid Robots, 2007.