RO:BIT Team Software Description Paper for Humanoid KidSize League of RoboCup 2023

Joon Ho Koh, Yong Yeon Kwon, Ji Hun Park, Dong Hui Jo

Jong Yeop Jeong, Seung Won Jang, Dae Kyum Kim, So Jeong Yoon

Robot sports game team of Humanoid Robot
School of Robotics, Kwangwoon University, Republic of Korea

joonho_koh@naver.com

Abstract. This document describes overall system of how robot works, image processing and dynamic control of humanoid. Our team strengthened competitiveness based on the experience of RoboCup 2019 and RoboCup 2022. We participate Humanoid KidSize League of RoboCup 2023 with improved technology compared to 2022's robot.

1 Introduction

Team RO:BIT is a professional robot game team of Kwangwoon University in Republic of Korea, established in November 2006. Humanoid team in RO:BIT research autonomous humanoid and participates in several domestic and international competitions. We basically research in 4 fields: image processing(vision), robot design, control of biped walking, and control system circuit.

2 Vision

2.1 Object Detection

We adopted deep learning for detecting a ball.

Fig.1. Deep Learning detection result

We used the YOLO[1] model. Since the main PC we are currently using does not have external graphics, we recognize the object using the YoloV4-tiny [2] model, which is the lightest model on Yolo. Originally, YoloV3-tiny was used, but YoloV4-tiny model is used for speed and accuracy. Learning was run on an external PC and only Learned data was imported and used. We succeeded in recognizing the crossing line of the field (X-Cross) and the ball as a result of learning (refer to **Fig. 1**).

We detected penalty-mark through image processing (refer to **Fig. 2)**. The recognized data will be used as the data required for localization.

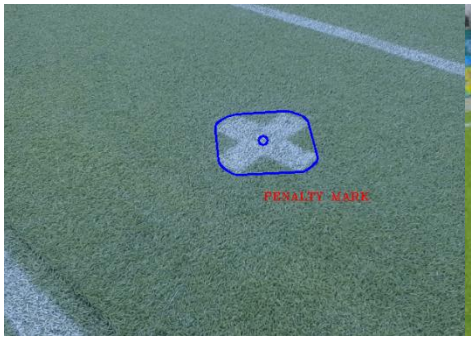

Fig. 2. Penalty-Mark detection result

Fig. 3. Result of line detection algorithm

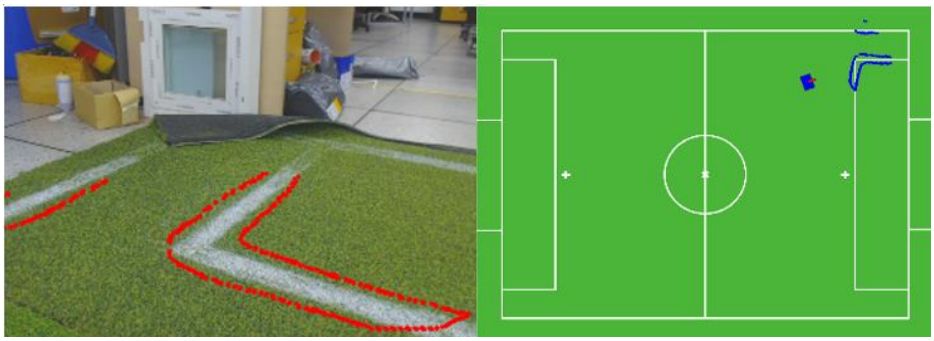

Fig. 4. Localization using vision data

Line was detected through image processing. First, convert the RGB image into HSV color space, and after canny edge detection, connect each edge through the contour and finally detect the line through the Hough Transform (refer to **Fig. 3**). So, we detect the edge of the white line and pick some points from the outside lines. This is used as data needed for localization (refer to **Fig. 4)**. And We will try to develop new vision system to recognize the ball and line through deep learning using semantic segmentation model.

2.2 Camera calibration

All objects in the field of play are perceived using camera sensor. To recognize robot's state in the field distance and orientation of interested object from robot is required. However, objects are projected on a 2D coordinate frame of camera in pixels. Geometric method is used to convert objects position in 2D coordinate frame in pixels to 3D world coordinate frame. This method is executed using intrinsic camera parameters(focal length and principal point) and using extrinsic camera parameters(height of camera from the field and tilt of camera). Extrinsic parameters are given that we know robot's height and tilt is controlled by main-controller (mini-pc). Intrinsic parameters are products of camera calibration [3]:

$$\mathbf{A} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{1}$$

where (f_x, f_y) is focal length, and (c_x, c_y) is principal point. Afterward, 2D coordinate frame in pixels is converted to coordinate frame of image plane (u, v):

$$\begin{aligned} u &= (x - c_x)/f_x \\ v &= (y - c_y)/f_y \end{aligned} \tag{2}$$

where $p(x, y)$ is object's coordinate in pixels. Using coordinate of image plane (u, v), height of camera h and tilt θ_{tilt}, distance and orientation of interested objects(ball, X-cross, Penalty-Mark) from robot are calculated:

$$\begin{aligned} &CC' = h \\ &C'P' = CC' \times \tan\left(\frac{\pi}{2} + \theta_{tilt} - \operatorname{atan}(v)\right) \\ &CP' = \sqrt{(CC')^2 + (C'P')^2} \\ &Cp' = \sqrt{1 + v^2} \\ &PP' = u \times CP'/Cp' \\ &d = \sqrt{(C'P')^2 + (PP')^2} \\ &\theta = -\operatorname{atan2}(PP', C'P') \end{aligned} \tag{3}$$

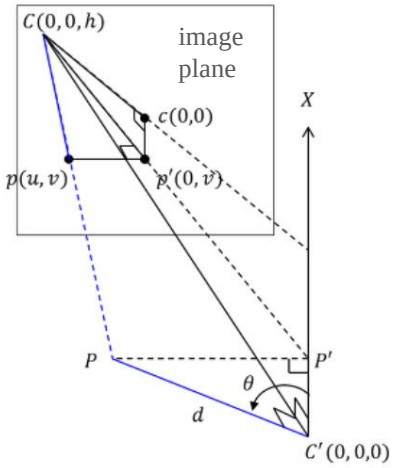

Fig. 5. Geometric method

where $C(0,0,h)$ is camera's position in 3D world coordinate frame, $c(0,0)$ the principal point in image plane, P position of object in 3D world coordinate frame, and d and θ are distance and orientation of object from robot. Last two products of equations (3) are used in our localization system.

3 Localization

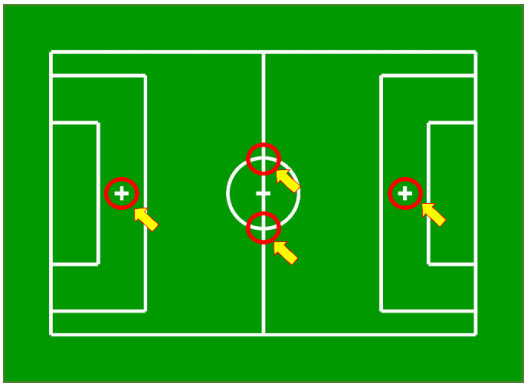

Fig. 6. Feature Points in Field

Our team's localization system based on Monte-Carlo-Localization [4] using walking odometry and location prediction with feature points. (Refer to **Fig.6)** Location with highest probability is predicted as exact location of the robot on field through Motion Model-Observation Model-Resampling process [5].

3.1 Motion Model

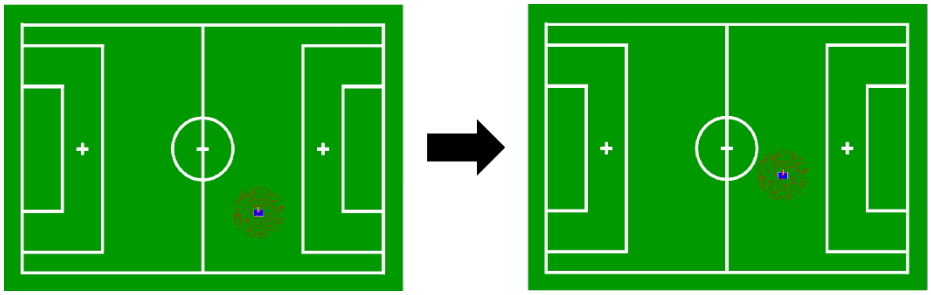

Fig. 7. Localization UI (Start/Playing)

The Initial Position of robot is specified. (Refer to **Fig.7**, Set Initial Position in Localization UI when game start.) We can know about direction and distance to travel of robot using IMU sensor and odometry model based on walking of robot. However, in this method has poor accuracy over time due to accumulated errors. So, it goes through the process of correcting the location using MCL.

Since MCL's motion model uses the odometry model, there are accumulated errors between the theoretical travel distance and the actual travel distance. For this reason, whenever the robot moves, particles are spread throughout the map. Using gaussian noise, we make the particles spread further.

Fig.7 is the localization UI. Red points are particles that mean a location where the robot may be. Blue square is the most likely place to have robot among particles are spread. At time goes by, you can see that the particles are scattered throughout the map by gaussian noise

3.2 Observation Model

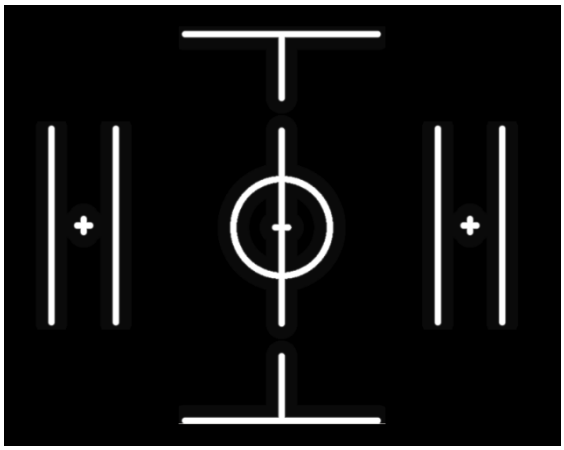

Fig. 8. Likelihood Field (characteristic lines)

Likelihood Field is a model that stores weights values according to each location. We can compute weights and create likelihood field applying likelihood-field-range-find model Algorithm [6]. Where (x_k, y_k) is k_{th} point, and (x', y') is characteristic line points. Afterward, nearest Euclidean distance is calculated between random point and line points(1). And distance is checked by max range reading variance z_{max}(2). Finally, weight q is computed using zero-centered gaussian normal distribution and uniform distribution with variance σ_{hit} (3):

$$q = 1$$
$$for\ all\ k\ do$$
$$dist = min\left\{\sqrt{(x_k - x')^2 + (y_k - y')^2}\right\} \quad (1)$$
$$if\ dist < z_{max} \quad (2)$$
$$q = q \cdot z_{hit} \cdot \frac{1}{\sigma_{hit}\sqrt{2\pi}} \cdot e^{-\frac{dist^2}{2\sigma_{hit}^2}} + \frac{z_{random}}{z_{max}} \quad (3)$$
$$return\ q$$

To create a likelihood field with the appropriate weights, we should control four parameters ($z_{hit}, z_{random}, z_{max}, \sigma_{hit}$). z_{hit} and σ_{hit} are variables related to measurement accuracy, z_{random} and z_{max} are variables related to measurement noise.

In the current model, the farther away from the center of the white line in the field, the lower the weight. We used the likelihood field to give weight to the particles to correct the robot's position. There is a slight difference in how data obtained through image is applied to the Monte-Carlo Localization (MCL) method, and in how we give weights. When measuring data for a line is obtained, each particle is weighted using the pre-calculated likelihood field (refer to **Fig.8**) with the distance and direction to the start and end points of the line received by the vision module

In the past, we start correcting the location when robot recognize any white lines anywhere. However, it was confirmed that the robot's position accuracy was poor because there were many white lines in the field. To solve this problem, we changed likelihood field with only characteristic line as shown in **Fig.8**.

3.3 Resampling

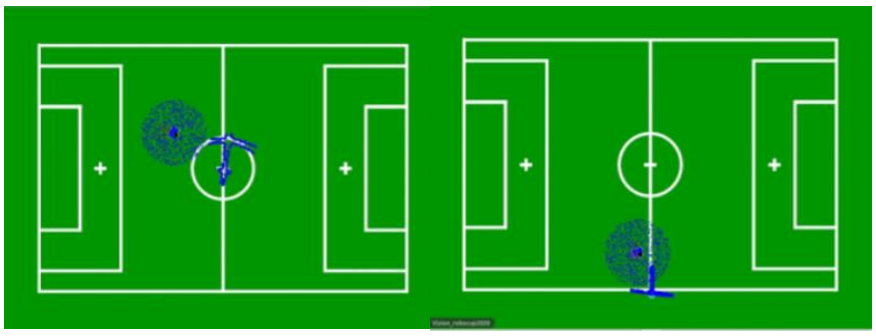

Fig. 9. Resampling Particles and Correct Location

Fig.9 is the resampling process in playing. First, we spread particles used motion model. Second, give the weights used observation model. Finally, we are proceeding resampling process that chosen particles have high weight. When estimating the robot's position using only the Motion model, the robot's position may not be correct over time. So, the robot's position is corrected through the correction process mentioned **3.2** and the Resampling process.

4 Walking Control

4.1 Kinematics

Our team solves the inverse kinematics of our robot's both leg with 12 degrees of freedom. Both leg with 12 degrees of freedom is divided into the left leg (6 degree of freedom) and the right leg (6 degree of freedom) from the center of the pelvis to solve inverse kinematics [7]. Use both ends of the robot's legs as end effectors and determine the angle of each motor based on the center of the pelvis. The motor angle can be determined according to the change of coordinate of the end effector.

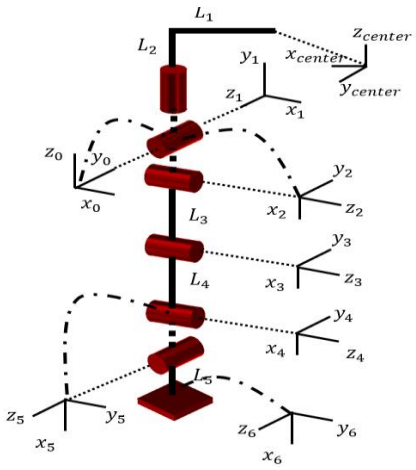

Fig. 10. Link Coordinate Frames of the Right Leg of a Our Humanoid Robot

Right Leg link D-H Parameter

Joint	θ_i	α_i	a_i	d_i
1	0	90º	0	0
2	-90º	-90º	0	0
3	0	0	L3	0
4	0	0	L4	0
5	0	90º	0	0
6	0	0	L5	0

Fig. 11. D-H Parameter

4.2 Walk Pattern

Our team generates moving patterns on the end effectors that are both humanoid ankle joint. The patterns are geometrically created on the x, y, and z coordinate planes, and the trajectory is generated using a fifth polynomial. Using a fifth polynomial, we draw a continuous trajectory without discontinuity of elements such as velocity, acceleration, etc. The pattern consists of eight variables that we can adjust passively: first time, first position, first velocity, first acceleration, later time, later position, later velocity, and later acceleration. These walking parameters can be adjusted passively to create the detailed shape of the orbit. Our team uses a manual tuning program that can easily adjust parameters and quickly test responses to adjustments and finds optimal parameter values that make walking the most stable through trial and error.

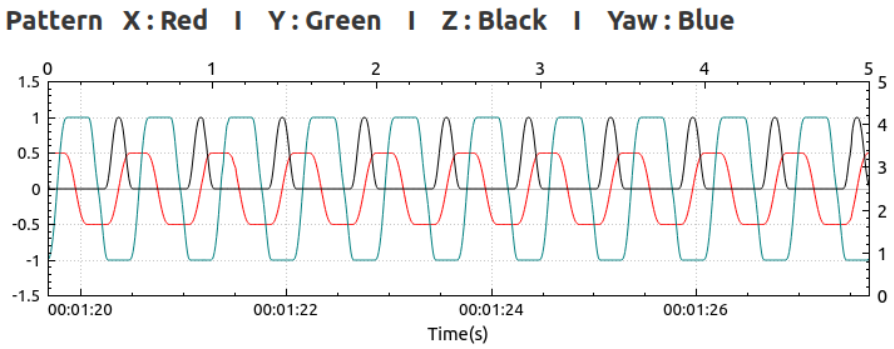

Fig. 12. Our Walk Pattern Right Leg x, y, z Coordinate when walking

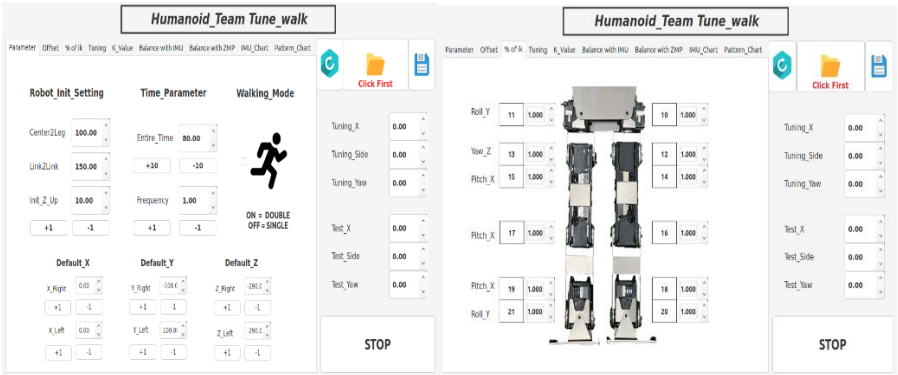

Fig. 13. Our manual tuning program

4.3 Push Recovery

In order to withstand collisions with other robots and walk stably on artificial grass field, humanoids must have the ability to respond to various obstacles. Therefore, our team improved walking stability by using the IMU (Initial Measurement Unit) to detect disturbance and allow the robot to control PID and feedback appropriately. To estimate the angle of the robot's torso and use it for feedback control, the IMU is attached to the body part the closest to the center of mass of the robot such as between the hip joints.

Our team measures ZMP (Zero Moment Point) from the robot's COP (Center Of Pressure) using feet with four load cells attached [8]. Based on the ZMP value measured in this way, the target ZMP can be reached through PID control so that the robot's ZMP does not deviate from the support polygon (foot supporting the robot) [9].

The PID controller controls the position of the pelvis, knee, and ankle joints. If the estimated value of the ZMP value and the robot's torso angle each exceeds the set value. threshold value, the final compensation value is determined through the PID controller [10]. And using that value, the pelvic, knee, and ankle joints are controlled to compensate for the errors in original inverse kinematics and restore the robot's torso angle and ZMP value to the normal range. This process helps prevent the robot from losing balance and falling and keep the ZMP in the support polygon. The controller's parameters and thresholds are also set as manual program through trial and error.

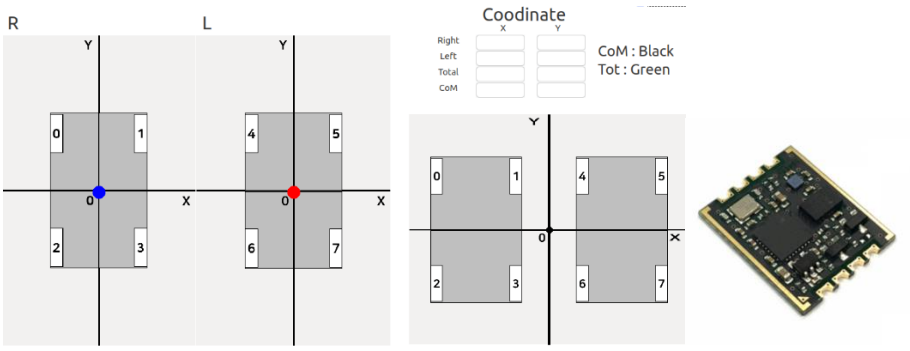

Fig. 14. Our Load Cell Control & IMU

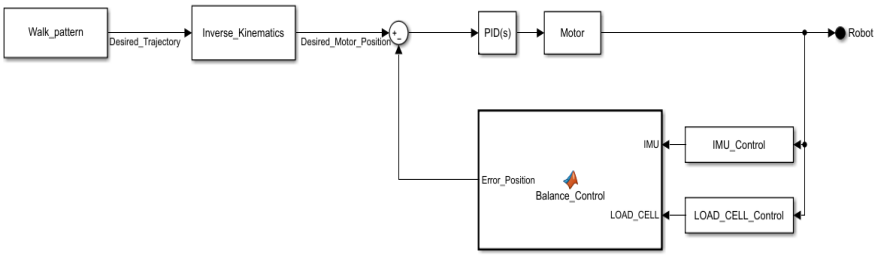

Fig. 15. Closed loop controller

Behavior

4.4 Overview of system

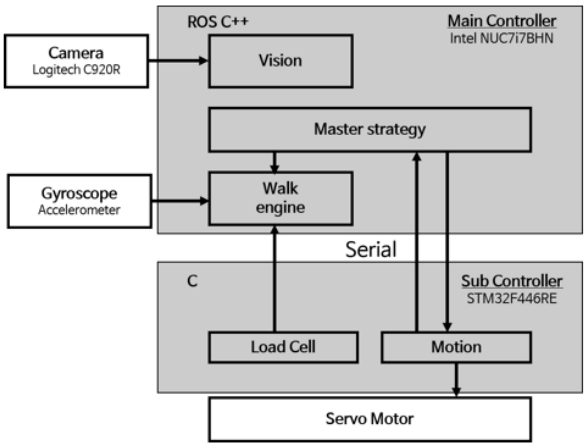

Fig. 16. overview system of robot

Our robot system consists of a C920R Logitech USB camera, a NUC7i7 main controller. Main controller executes overall algorithms and image processing on Linux using ROS (Robot Operating System). Sub controller, circuit with MCU based on STM32F446RE, processes gyroscope acceleration data, load cell data and controls motors (DOF). Gyroscope acceleration is filtered and used to determine fall of robot and for balance control, and such data is transferred to main controller via RS-232. Load cell used to determine humanoid COP (Center Of Pressure), and such data is transferred to main controller via RS-232. The main controller and the sub controller are connected by serial communication.

Based on the data obtained from the image processing and sub controller, the main controller program the whole robot driving algorithm and transfers the necessary motor motions to the sub controller, and the sub controller executes the kinematics and motion based on the data.

4.5 UDP Communication

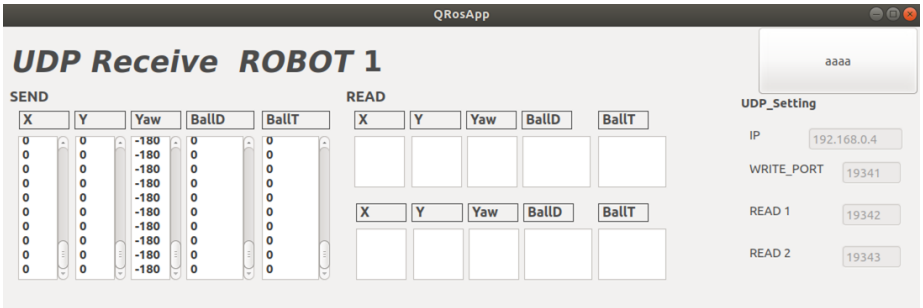

Fig. 17. UDP communication program

We are using UDP communication to share the position of the robot and the position of the ball for our team play. First, we find IP address, and connect UDP socket and communication port. Through this, collisions between robots were prevented or movement was prioritized to enable high-quality team games. In addition, this allows us to operate the game strategically and carry out cooperative missions such as technical challenges

4.6 Overview of Soccer Algorithm

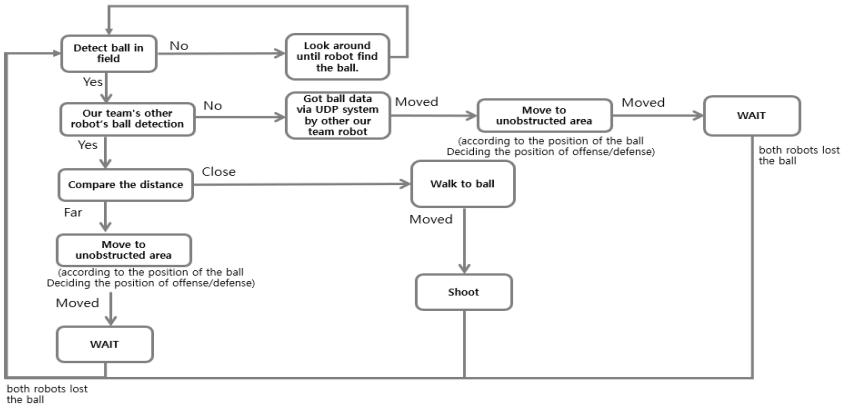

Fig. 18. Overview of Soccer Algorithm

Our team has been participating in the RoboCup for several years, and we've developed a lot about how to walk efficiently towards the ball, how robots can exchange data on the ball, and team play. First, when there are two or more robots in the stadium, we each move the robot's pan-tilt motor to find the ball and then we walk around their set areas and quickly search it in a wide field if we can't find it for a several times. If our team's robot finds the ball, the robot works in the same Wi-Fi environment, so we can share the data we get from each other using UDP method. And we

share the robot's position on the field. So, to save time and efficiently play soccer, the robot that is close to the ball approaches and kicks the ball, and the other robot moves forward or sideways depending on the position of the ball so that it doesn't get in the way of the robot kicking the ball. Also, it creates a curved path from the robot's current position to the ball so that the kicker aligns with the opponent's goal post and quickly maintains kickable distance.

Our team has an algorithm for goalkeepers. In each section, when the ball is outside the penalty box, it gives information about the ball to the player robot and does not move. Then when the ball is in the penalty box, it aligns the x-coordinates of the ball and the keeper robot itself. And when the ball was int the penalty box, we created an algorithm to kick the ball and kick the ball and get it back to its seat.

Reference

1. Joseph Redmon∗, Santosh Divvala, Ross Girshick, Ali Farhadi, “You Only Look Once: Unified, Real-Time Object Detection”, IEEE Conference on Computer Vision and Pattern Recognition, 2016 IEEE Conference on. :779-788 Jun, 2016

2. YOLO: Real-Time Object Detection. https://pjreddie.com/darknet/yolo/

3. L. Deng, G. Lu, Y. Shao, M. Fei and H, Hu, “A Novel Camera Calibration Technique based on Differen tial Evolution Particle Swarm Optimization Algorithm”, *Neurocomputing*, vol. 174, pp. 456-465, Jan 2016.

4. A. C. Almeida, A. H. R. Costa and R. A. C. Bianchi, "Vision-based monte-carlo localization for humanoid soccer robots," 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), Curitiba, pp. 1-6, 2017

5. I. Nagi, W. Adiprawita and K. Mutijarsa, "Vision-based Monte Carlo localization for Robocup Humanoid Kid-Size League," 2014 13th International Conference on Control Automation Robotics & Vision (ICARCV), Singapore, pp. 1433-1438, 2014

6. S Sebastian Thrun, Dieter Fox, Wolfram Burgard, “PROBABILISTIC ROBOTICS”, pp. 139-200, 1999-2000.

7. M. A. Ali, H. Andy Park and C. S. G. Lee, "Closed-form inverse kinematic joint solution for humanoid robots", *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, pp. 704-709, 2010.

8. Youngjin Choi, Doik Kim, Yonghwan Oh, and Bum-Jae You, “Posture/Walking Control of Humanoid Robot based on the Kinematic Resolution of COM Jacobian with Embedded Motion”, *IEEE Transactions on Robotics*, vol. 23, No. 6, pp. 1285-1293, Dec 2007.

9. Napoleon, Shigekhi NaKaura, and Mitsuji Sampei, “Balance Control Analysis of Humanoid Robot based on ZMP Feedback Control”, *IEEE International Conference on Robotics and Automation*, vol. 3, pp. 2437-2442, Oct 2002.

10. Yisoo Lee, Jaeheung Park "Real-Time Force Control of Biped Robot to Generate High-Speed Horizontal Motion of Center of Mass", *Journal of Korea Robotics Society*, Korea, pp. 1-10,2016