Plasma-Z
Extended Team Description Paper

Chitchanok Chuengsatiansup[1], Thiraphat Charoensripongsa[1],
Kanit Wongsuphasawat[1], Komsit Rattana[1], Pawawat Doungsodsri[1],
Aphilux Buathong[2], Kittipat Wejwittayaklung[3],
Manop Wongsaisuwan[4], Wittaya Wannasuphoprasit[3]

Department of Computer Engineering[1]
Department of Industrial Engineering[2]
Department of Mechanical Engineering[3]
Department of Electrical Engineering[4]
Chulalongkorn University
Phayathai Road, Bangkok 10330, THAILAND
http://www.eic.eng.chula.ac.th
elizaemol@gmail.com, tjump_c@hotmail.com

Abstract. This document describes more detailed design in both hardware and software aspects of the currently developed Plasma-Z soccer robots. The hardware part includes vision equipments and the mechanical system of the robot. The software part contains AI layers and vision system. We also describes how hardware and software modules work together to control the robots. Additionally, we provide an example of a situation and explain the response of our system.

Keywords: RoboCup, Small-Size, Robot, Soccer, Vision, Artificial Intelligence, Control

1) Introduction

Plasma-Z is a robot soccer team from Engineering Innovator Club, Faculty of Engineering, Chulalongkorn University, Thailand. Plasma-Z has joined the RoboCup Small-Sized League since 2003 and has won the RoboCup Thailand Championship for 6 consecutive years.

Moreover, in the international level, Plasma-Z has joined World RoboCup Championship since 2003. Plasma-Z has showed significant improvement since we almost reached the quarter final at Osaka RoboCup in 2005. Later, we won the third place and technical challenge at Bremen RoboCup in 2006, and won the second place at 2007 Atlanta World RoboCup. Currently, Plasma-Z was the champion of World RoboCup Small-Sized Robot League at Suzhou, China, 2008.

In Section 2, we give the details of the hardware parts of both the vision system and the robots. The basic mechanical components of the robots are described in details. We also discuss about the background idea of how and why all components are chosen. The electronics design of the robots is also elaborated. The software parts of the system are explained in Section 3. These include the software for the vision system and the artificial intelligence system. For examples, the topics of vision calibration, networking, motion control, simulator, and playing strategies are discussed in this section. Conclusions are given in Section 4.

2) Hardware

2.1 Vision Hardware Component

Our vision consists of two Stingray F-046B/C video cameras, space-saving IEEE 1394 SVGA C-mount cameras equipped with a highly-sensitive SONY 1/2 progressive CCD sensor. At full image resolution, the AVT STINGRAY F-046B/C offers up to 61 fps.

Fig.1 Stingray F-046B/C (1)

The Stingray F-046B/C's high resolution is exactly suitable for the new RoboCup pitch size. Higher resolution image data are interpreted into more accurate robot and ball positions since we have more pixels for the localization process. Additionally, high frame-rate speeds up our input rate and increase output data's precision. Higher sensitivity also offers more system elasticity for gloomy environment.

Another advantage of Stingray is that the Stingray camera series include FirePackage API and SDK providing full camera access and control. With FirePackage, we can implement image property controller from our software. Hence, we can adjust brightness, UB and VR value (of YUV Color Model), shutter speed, gain, hue, saturation and gamma from our application. However, we can adjust neither the focal length nor the focusing ring from our software. Accordingly, during our camera setup, we need to finish our camera placement and focal length/focus ring adjustment. Then, we can directly alter other properties from our vision software.

Because of high resolution and high frame rate images, these two cameras need to be connected to our vision computer through high bandwidth FireWire 1394b Adapter, IOI FWB-PCIE1X20 which is equipped with 2x1394b PCI Express. Both FireWire and PCI Express provide sufficient bandwidth for the transmission. Moreover, this adapter also includes screw locking which prevents the problem of loosen wires.

Table 1 AVT STINGRAY F-046B/C Technical Specifications (1)

AVT STINGRAY F-046B/C Technical Specifications	
Image Device: Type 1/2 (diag 8mm) progressive scan SONY IT CCD, ICX 415	**Resolution Depth:** 8 bit / 14 bit (16 bit in High SNR mode)
Effective Picture Elements 780 (H) x 580 (V)	**Picture Size** 780 (H) x 580 (V)
Digital Interface: IEEE 1394b (S 800 daisy chain)	**Transfer Rate:** 100, 200, 400, 800 Mbit/s
Frame Rates **up to 61 fps (full frames)**	**Gain Control:** Manual: 0-24 db; Auto Gain
Shutter Speed: 30 µs - 67 s; Auto Shutter	

More specification can be found at Allied Vision's website:
http://www.alliedvisiontec.com/avt-products/cameras/stingray/f-046-b-bs-c-fiber.html

2.2 Robot

Our robot's hardware design can be categorized into two main systems: mechanical system and electrical system. These two systems have to be designed so that they are compatible with each other for the best performance.

2.2.1 Components

The mechanical system is divided into five subsystems: a driving system, a flat-kick system, a chip-kick system, a dribbling system and a robot's structure system.

2.2.1.1 Driving System

The driving system consists of omni-directional wheels, motors, gear heads, motor mounts, encoders and etc as shown in **Fig.2**.

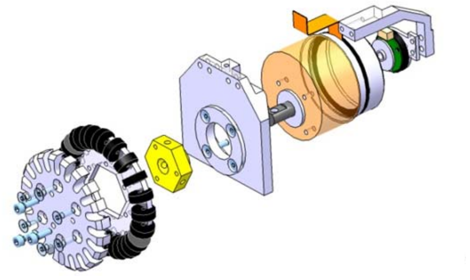


Fig.2 Exploded view of the driving system

Omni-directional wheels have become a mainstream in the small size league because robots often have to spin or rapidly change their moving direction. Using this kind of wheel, the robot easily obtains 3 degrees of freedom in movement: translating in X and Y direction and rotating in Z direction. The robot can move to any direction freely because of several small rings which are attached to the edge of the large wheel.

Motor is an actuator of the wheel. Connecting it with appropriate gear head ratio can make the robot move with desired velocity and acceleration. A motor mount is used for attaching the motor with the robot. An encoder is a device used to help calculating the angular velocity of the wheel or the motor.

2.2.1.2 Flat-kick System

The flat-kick system composes of two important parts which are solenoid and plunger as shown in **Fig.3**.

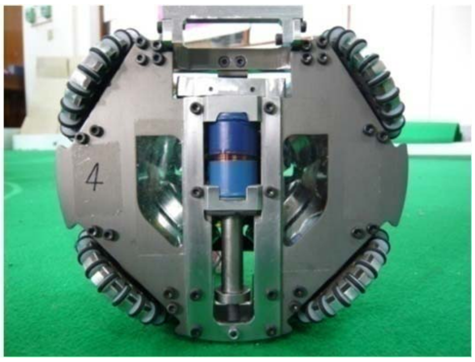


Fig.3 Flat-kick system

The solenoid is wound around the plastic core with copper wire. When the current flows through it, a magnetic field is built. The plunger made from soft magnetic material is pulled by magnetic force generated by the solenoid to kick the ball. To design the flat-kick system, there are many parameters to be determined, but we will discuss later in mechanical design section.

2.2.1.3 Chip-kick System

This system is similar to flat-kick system. Instead of exerting force to the plunger, we have to attach some parts to the plunger to convert linear motion into circular motion in order to make the ball bounce up to the air.

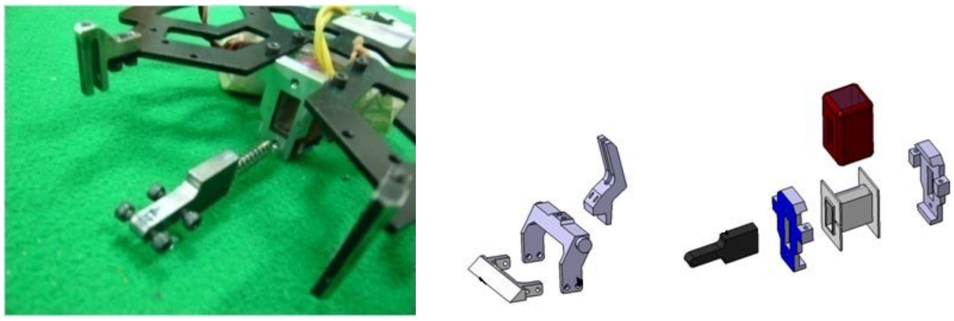


Fig.4 Chip-kick solenoid rod

2.2.1.4 Dribbling system

The main responsibility for dribbling system is to hold the ball with the back spinning motion of the dribbler shaft. For RoboCup soccer 2008, our team use EC-max 22, 25 watt, as a dribbler motor. Generally, the factor involved in the efficiency of the dribbler includes shape of dribbler shaft, an elasticity of dribbler material, diameter of dribbler shaft, the power of the motor, speed and friction force of the shaft that exert on the golf ball, the height of the shaft from the ground to the center of the shaft. Considering these factors lead to the trial-and-error to find the most proper solution.

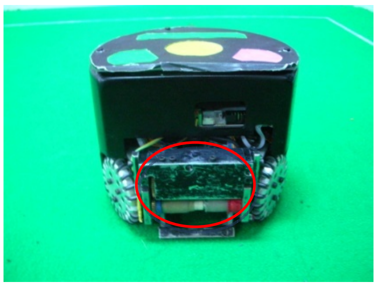


Fig.5 The front view of the robot and overview of the dribbler

After the experiment, we have the following observations:

1. The speed of the dribbler shaft affects this system as it improves the ability to stop the ball as soon as it touches the dribbler. After testing at many different speeds, we found that the most appropriate rotating speed is 13,000 rpm. Furthermore, we select the shaft with 10mm outer diameter and

6.5mm inner diameter. As a result of many experiments, these diameter sizes of dribbler give the most efficient ball dribbling.

2. The material wrapped around the dribbler shaft has a significant impact on controlling the ball. Too sticky or too slippery materials will make the golf ball bounces out of control.
3. We design the shaft with shape and groove as shown in **Fig.6**.

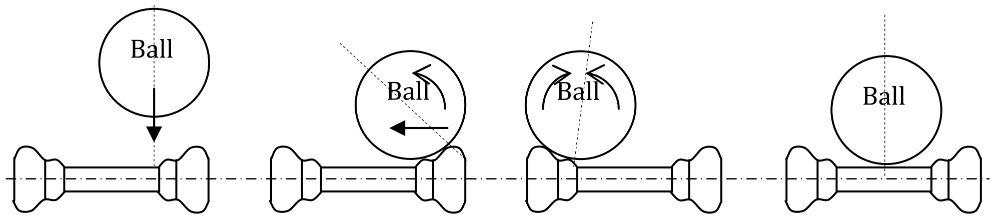


Fig.6 The grooved shaft makes the ball roll toward the center. Dashed line represents the ball rotating axis which is normal to the contacting point.

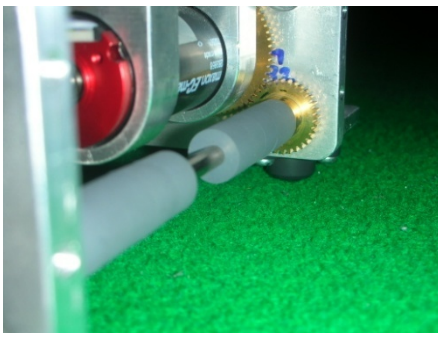
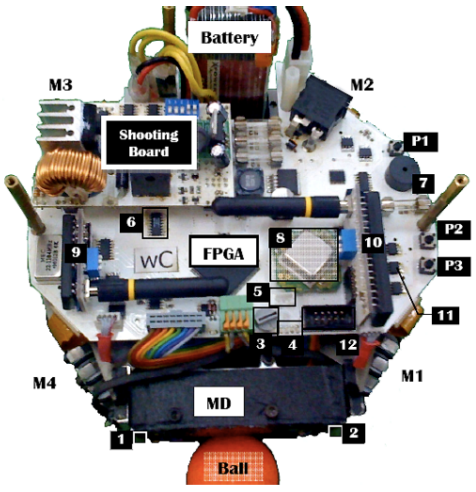


Fig.7 The dribbler experimental tool

4. In general, the dribbler shaft has to withstand a great amount of force within a short period of time. A suspension system provides a good solution that allows us to increase the capability of holding the ball right in front of the robot as it extends the ball and dribbler shaft impact time.

The electrical system consists of circuit board and other electrical parts. The details of our electrical and mechanical systems will be discussed in the following sections.

2.2.2 Electronic Design & Fabrication



1. Infrared LED
2. Photo Transistor
3. Code Switch
4. 4 LEDs Display
5. RS232 Connector
6. Analog to Digital Converter
7. Buzzer
8. Gyro meter
9. RF Transmitter Module
10. RF Receiver Module
11. Motor drive circuit
12. JTAG Connector

P1 Discharging Button
P2 Reset Button
P3 Mode Selection Button
M1, M2, M3, and M4 Motors for each wheel

Fig.8 Circuit overview of Plasma-Z's robot

The electronic part of Plasma-Z team has been designed and developed continuously for more than 6 years. The PCB layout is changed and modified every year. The good parts are inherited, and the bad parts are improved year by year. The major changes during the past 3 years are the processor circuit, the motor driving circuit, and wireless communication module. In the latest version, the main processor has been integrated into FPGA contrasted to the former version which was separated. This design not only reduces the processing cycle time but also increases the space on the PCB. The motor driving circuit which was changed from H-bridge circuit to 3-phase inverter circuit for driving 3-phase Brushless DC (BLDC) motors. This makes the robot move faster while the robot speed can be controlled more accurately. However, its cost is higher, and it is more complicated to make a good control. The wireless communication has not been changed much in the circuit layout, but we change the selection of RF Module (or chip).

The circuit was designed to satisfy all required functions of the soccer robot. The space on the PCB is very limited due to the space of mechanical part. Furthermore, we try harder to make the PCBs be easy to assemble, debug, and run in the test mode. The latest version of electronic design contains 4 boards which are main circuit board, RF receiver board, RF transmitter board, and Shooting board. All circuits connected with FPGA are categorized into 7 following parts:

1) Human Interface part: This part contains input and output/display circuit. There are three inputs from 3 push buttons for resetting the system, mode selection (for test mode), and discharging the capacitor. Furthermore, a 4-bit code switch is used to get an input data of robot number and test mode selection. For the output, 4 LEDs array and buzzer are also equipped.

2) Computer Interface part: This includes RS-232 and JTAG which are frequently used to debug the programming problem and send out a large amount of data which cannot be displayed by the Human Interface part.

3) Wireless Communication Part: This circuit part is the path for data package transmitted and received serially to frequency modulating modules. RF ICs module used to receive and transmit data are LINX©, RXM-900-HP3-PPS and TXM-900-HP3-PPS. These are FM types which operate over the frequency range of 900MHz.

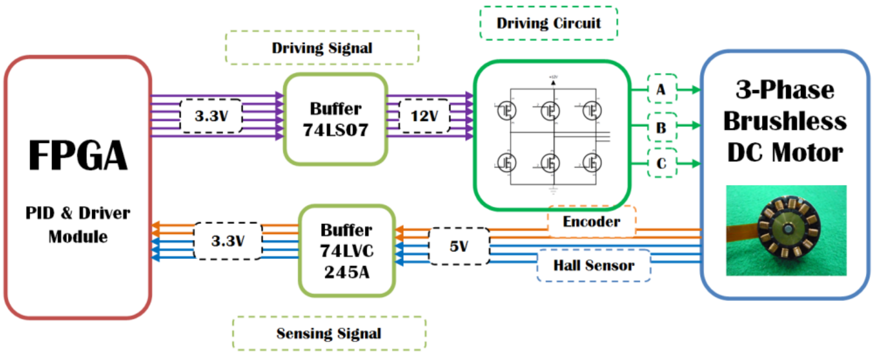


Fig.9 Brushless DC Motor Diagram

4) Brushless DC Motor Driving part: There are 5 BLDC motors in each robot. Four of them are used as the robot wheels for omni-directional movement and one for dribbling the ball. The input signals from the motor are two of digital quadrature encoder feedback signals and three of built-in hall sensors. The signals from miniature encoder and hall sensors are isolated from FPGA by 74LVC245A IC buffer. To drive each motor, the output signals from 6 pins of FPGA control 6 MOSFETs to switch the current flowing through the stator coil of each BLDC motor. The signal path between FPGA and driving circuit is connected through 74LS07 IC Buffer. The direction of motor rotation can be controlled by re-order the phase drive signals (Phase-A, B, C) e.g. A-B-C-A-B- to C-B-A-C-B.

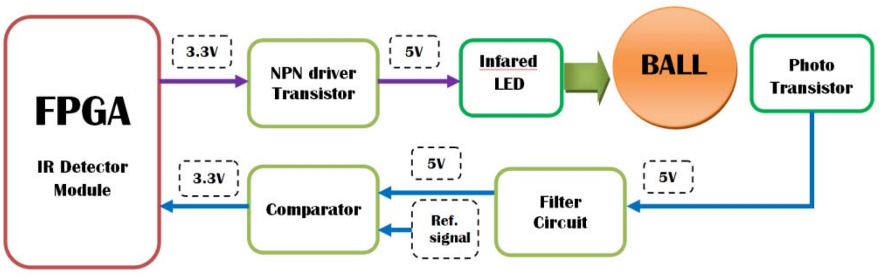


Fig.10 Ball detection diagram.

5) Ball Detection Part: As shown in **Fig.10**. FPGA generates rectangular wave with frequency 2.4 kHz to an infrared LED. It sends out the infrared rectangular wave without the low-frequency noise problem. The photo-transistor is used for detecting this infrared signal. If the ball lies close to the robots dribbler, it will block an infrared signal, and the photo-transistor would not be turned on. On the contrary, if the ball is not there, the photo-transistor will turn on and let the signal pass through a high-pass filter circuit to a comparator circuit. It will be compared with an offset reference voltage to show the state whether the ball is there or not.

6) Shooting Part: This system consists of controller system, charging system, and the set of capacitors and solenoids. It is separated from the main board so that it can operate on its own (**Fig.11**). There are 5 signals for controlling this system; charge, enable, flat-shooting, chip-shooting and voltage reference. Ground of the main board and shooting board is separated by optocouples. For charging system, pulse width generated by IC 555 is used to switch the MOSFET for boosting circuit. Then two capacitors are charged to 250 Volts. Capacitor voltage is controlled by a comparator logic using an operational amplifier (OPAMP). There are two types of shooting which are flat-shooting and chip-shooting. The main board generates a pulse for opening IGBT gate which enables current flows from capacitors through the solenoid. The velocity of the ball is also controlled by the period of this pulse.

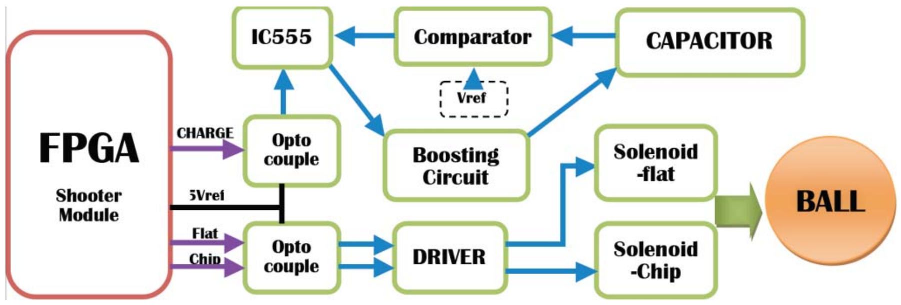


Fig.11 Shooting diagram.

7) *Power Supply Part*: From the +14.8V battery, there are 4 fuses for protecting 4 electronic parts; left-side motors, right-side motors, shooting board, and main board. The lower voltage levels on the main board are: +5V (LM2596)[1] for digital part; +5V (78L05BP) for analog part; and +3.3V (LM1117), +2.5V (LM317), +1.2V (LM317) for FPGA. To connect an analog ground to a digital ground, R0Ω is used for interposition. The ground of +16.6V for motor is separated by a ferrite bead to prevent a high transient voltage change contributed from a high current when start a motor.

[1] The name in the parenthesis specify the name of regulator IC

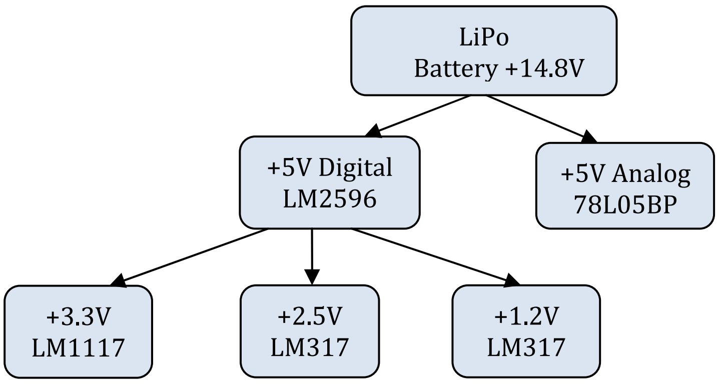


Fig.12 Diagram of power supply part.

2.2.3 FPGA Design

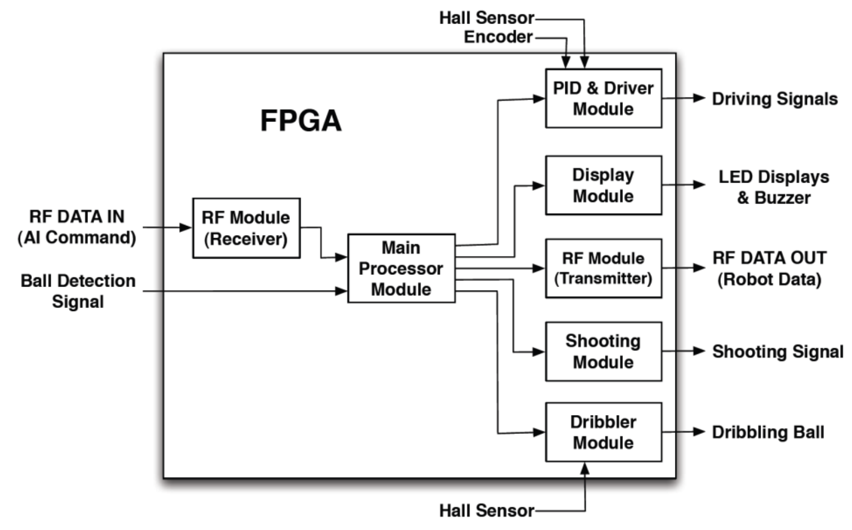


Fig.13 An FPGA Module overview

FPGA (Xilinx: SPARTAN3-XC3S400) system is composed of 8 modules as shown in **Fig.13**. There is one main processor module. A significant input in robot soccer is RF command sent from AI system to control the robot. The output are all the actions that robot performs which are the movement, shooting the ball, dribbling the ball, LED display and Buzzer. The modules in an FPGA are designed and debugged by using software Altium Designer version 6.5. The total number of occupied slices for synthesizing all modules is 3582 or 99%. The details of each module are described as follow:

1) Microprocessor Module: The previous version of robot soccer system used 4 microcontrollers to process PID control and one main microcontroller. This uses a lot of space on PCB. By integrating MCU Nexar TSK51 (2) in FPGA, it uses 2379 of occupied slices for synthesizing the MCU or 66% of total slices. The space can be reduced and the speed of the overall system is also increased. The MCU provide 2 modes in which the robot works, test mode and run mode. Test mode is used for manually checking the robot in order to make sure that all components will work normally. The need for test mode is crucial since the robot is very complicated and has high probability of malfunctioning in every part. Run mode is used in the normal competition.

2) PID and Driver Module: This module is used in Motion Control System to control the movement of the robot. It makes closed-loop control system to control the angular velocity of each motor. The detail is described in Motor control section.

3) IR Module: Its function is the ball detector. The FPGA generates rectangular wave to an infrared LED. The phototransistor is used to detect the ball in front of the robot. It also used in wait kick mode where the detail is described in Kick Control section.

4) RF Module: This module can be divided into two parts: RF Receiver and RF Transmitter. RF Receiver is used to receive the command sent from AI System, and RF Transmitter is used for sending out the robot's data to AI system. However in this version, the transmitter is not yet employed. The detail of the operation of the receiver is described in Wireless Communication section.

5) Shooting Module: This module controls the operation in Shooting System according to the AI command. These operations are charging, flat shooting, and chip shooting. It also cooperates with the IR Module to emerge *wait kick* mode. More information is described in Kick Control section.

6) Dribble Module: This module control the dribble motor to keep the ball in front of the robot. Dribble Control section describes this module in more detail.

7) Display Module: This module controls the human interface for testing and displaying. The modules and systems that can be tested are IR module, shooting system, dribbler system and driving system. For displaying, 4 LEDs equipped to show the state of the robot such as the identification number of robot, testing mode and etc.

2.2.3 Mechanical Design & Fabrication

In this section, we discuss more details about our mechanical systems.

2.2.3.1 Driving system

There are several parameters which have to be considered when design a driving system.

Number of wheel on the robot

In general, the robot may have three or four omni-directional wheels because three omni-directional wheels at three different angles are sufficient for omni-directional driving. Additional wheels provide redundancy to the system (3). However, we had chosen the four-wheel system because it is easier to allocate a flat-kick system and a chip-kick system inside the body of the robot.

An angle of the wheel and size of the wheel

An angle of the wheel is measured relative to the x axis in the robot coordinate frame reference shown in **Fig.**14.

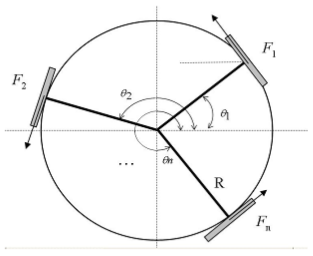


Fig.14 Arrangement of n wheel and their angles

An angle affects maximum velocity and acceleration of the robot in any direction as can be seen from **Fig.15**. Using our present wheel's angle, our robot has more maximum velocity in Y-axis than X-axis (**Fig.15**) while opposite in acceleration. Moreover, an angle of the wheel can also affect to the remaining space of the robot to allocate other components. The angle of our front wheel is 33º, and the back one is 45º. This design allows our robot to have enough dribbler length (about 7 cm) to dribble the ball effectively. However, unequal of front wheel angle and back wheel angle can cause some controlling problem.

The wheel size or the radius of the wheel is a parameter which has a strong relationship to the final velocity of the robot as given by:

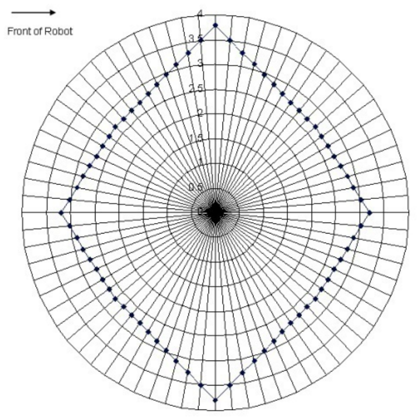
$$V = \omega R . \qquad \textbf{(1)}$$

Where,

V	is linear velocity of the robot (m/s),
ω	is the angular velocity of the robot (rad/s), and
R	is a radius of the wheel.

Because of the field-size increasing in 2008, the maximum final velocity is important. Therefore, we designed a big wheel instead of reducing the gearbox ratio substantially because maximum torque of the wheel is also important.

The diameter of our wheel is about 61 mm. With practical motor speed of about 4200 rpm and 5:1 gear ratio, the theoretical robot's forward average speed is 2.91 m/s.



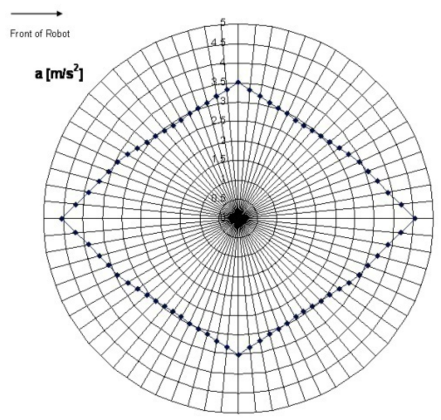


Fig.15 Velocity profile of the robot (left) and acceleration profile of the robot (right)

Type of motor and gear head ratio

There are two important criterions in selecting the driving motor which are current and power. We can calculate the power of motor from mass, velocity and acceleration of the robot. After calculating the power, we choose the motor which requires low current when operating at that power. We decided to use EC flat 45, brushless, 30 watt as the driving motor because of its efficiency and suitable dimension. This motor has high power, but it is small and thin. Moreover, it comes with hall sensors. These help us control the speed of the motor much easily.

A standard gear head Maxon GS45 with 5:1 gear ratio is introduced in order to guarantee the smoothness of the motion and reduce the effect of different friction of each gearbox, which causes us some difficulty in 2007 version. Its 8 mm shaft is neatly drilled by 3 mm diameter hole, with 3 mm depth, to prevent the wheel from slipping off and rotate them together. With all these, our robots have average acceleration of 3.31 m/sec^2.

Encoder and Others

An external encoder, US Digital E4P 300 teeth, is attached at the end of each motor. All aluminum parts of the robot are also coated by hard anodizing process in order to prevent an accidentally electrical short-circuit. The cover of our robot is made of composite material, 2 mm-thick fiberglass, in order to protect inner mechanical and electrical components. The material selection is totally redone: for example, front rubber bands are replaced by springs, to avoid the non-uniformity in material quality. The total robot structure is designed with the easy-to-assemble concept in our mind.

2.2.3.2 Shooting system

In shooting system, there are many parameters to be optimized such as plunger properties, solenoid properties, etc.

Plunger properties

The plunger should be made from a high magnetic permeability material which is always soft magnetic steel in order to absorb energy from solenoid as much as possible. Since the higher magnetic permeability results in higher absorbed energy. High permeability steel must have low carbon percentage (medium or low carbon steel), and a large and oriented gain. The plunger should be long because an extended length can also absorb the energy. However, the area far from the solenoid, the magnetic field is nearly zero. If the plunger is too long, very little energy can be absorbed by the extended part. Moreover, the plunger's diameter should be big as much as possible in order to absorb energy from electrical energy.

Coil properties

Higher number of turns and layers of coil gives higher force of plunger, but it also increases delay of shooting system.

In 2008 version, our chip-kick system is distinctively changed because of an extension of new spur gear head. The best solution is by reshaping the solenoid from circular to rectangular and by placing it into the available space precisely. The plungers of both flat and chip kick are made of S45C iron which has high magnetic permeability and eventually leads to powerful shooting force. The 2008 version robot is able to kick straightly up to 10.7 m/s and chip the golf ball more than 3.7 m long. However, in a game, the robot can adjust its ball shooting speed via modifying the turning-on duration of the IGBT gate.

The flat-kick is attached at the bottom chassis, which acts like a huge heat sink for heat reduction, and supported by 2-mm U-shape stainless steel plate, while the chip-kick is held by the top chassis. These chassis plates play a role as a backbone of the robot as all 4 driving motors are mounted on them. The top chassis also serves as a base platform for placing electrical PCB and robot's cover.

The new figure of chip-kick and flat-kick are both based on same fundamental concept: high durability and effectiveness. Hence, we designed them in the simplest way by reducing the number of flat shooting parts from 5 pieces, in 2007 model, to 3 pieces, while in the chip-kick system, from 6 pieces to pieces. Besides, we also widen a face of shooting rod by 52 mm and 68 mm in flat-kick and chip-kick respectively.

The concave aluminum face of the flat-kick rod results in straighter shooting direction even though the initial position of the ball is not at the center. A tension spring is hooked at the back of flat-kick solenoid rod while a compression spring is squeezed inside chip kick solenoid cavity. These springs are very trustworthy since they outlive former rubber bands.

2.2.4 Batteries

Lithium polymer battery (LiPo) is chosen to be the main power supply. The model XP18004GT from the Dualsky holds the electrical specifications with 1800 mAh, 14.8 V (discharged) to 16.8 V (fully-charged), and 4-cell batteries with extra-low internal impedance in one pack. The dimension is about 100 X 34 X 24 mm for length, width and depth respectively. The weight is 198 g.

The voltage of each cell varies from about 2.7 V (discharged) to about 4.2 V (fully-charged). During charging, each cell is protected from overcharge by limiting the applied voltage to not to be more than 4.2 V per cell (used in a series combination). Overcharging will likely results in an explosion and/or fire of batteries. During supplying to load, the load has to be removed as soon as the voltage drops below approximately 3.0 V per cell (used in a series combination), or else the battery will subsequently no longer accept a full charge and may experience problem about holding voltage under load.

Fig.16 Lithium Polymer battery (LiPo)

3) Software

Our software consists of two software subsystems running on two separate computers, vision system and artificial intelligence system.

Basically, the vision system is responsible for handling each input image data frame from the camera and converting the image data into robot's and ball's positions. Then, it sends the information of each frame to the AI system through a socket opened via Local Area Network (LAN).

Meanwhile, the AI system consists of two main parts: vision filtering system and strategy system. Since the image processing in the vision system consumes a significant amount of time and causes delay, we create the filtering system in software procedure to convert delayed raw positions from the vision system into predicted present positions. The strategy system is considered to be the most important part of our system. Its duties are to plan and control our robots. In other words, it is our system's brain.

In order to perform an implementation and test each robot function independently, we use layer-architecture design approach. By means of layer-architecture, our strategy system is divided into 7 layers including: manager, play, group, role, skill, trajectory, and control layers. Next seven paragraphs will outline basic description of these layers.

Manager: Manager is analogous to a human soccer team manager. By receiving data such as game score, ball possession, opponent pattern, and referee box signal, manager applies the most suitable play for that moment.

Play: Play provides a real strategy of AI which consists of many game plans. Play will determine the pattern of the game and notify to all robots by assigning robots into 'Group.' Play also selects zone for robots and may issue some commands directly to robot role such as force role to pass the ball to another robot or to shoot the ball.

Group: Group is to implement a group of collaborative robot group to perform a mission such as offensive group and defensive group. One group implementation may be used in many different plays. For instance, there are plays that use the same defensive group but use different offensive group.

Role: Role is implemented as robot behavior which is assigned by 'Play' or 'Group' to control robot to perform a specific action such as manipulating ball inside zone, running to zone, scrambling ball from opponent, getting ball. The 'Role' mechanism first assigns a particular skill to the robot, and then generates the best point for robot action and set another parameter to 'Skill.'

Skill: Skill is a set of basic knowledge for every robot, such as how to move to a point, how to get the ball and shoot. Skill module generates path (a set of points), dribbling and kicking commands that will be processed by varying of trajectory module as selected by 'Skill' module. Each skill has different main idea of generating path for robot. For example, 'get-ball skill' differs from 'move-to-point skill' in many ways. We can study and test each skill independently for the best performance.

Trajectory: Trajectory is a set of methods that generates velocities to control robots. Since each skill focuses on different points, for example, some skills require fast motion while others require accurate positions; different trajectories are created to serve these various needs.

Control: Control is the lowest layer of AI. Its duty is to control the robots' movement. Control receives control data such as velocity, angular velocity from 'Trajectory.' Then Control converts these data into a specified format and sends to each robot via RF device.

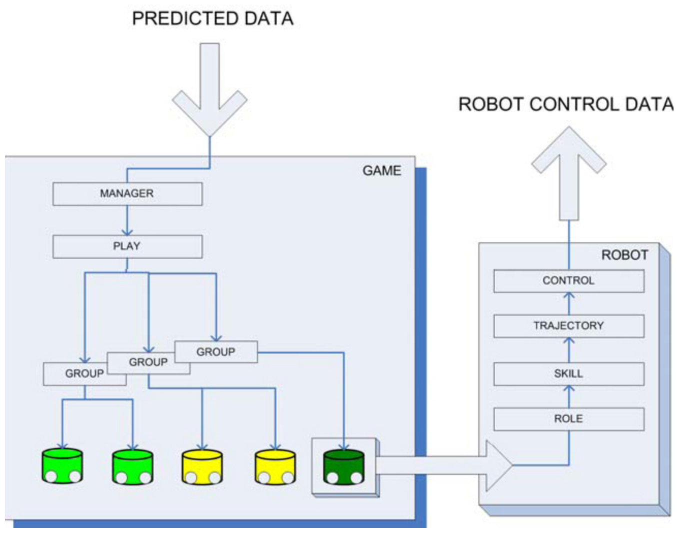


Fig.17 AI's layer architecture

The following sections will give more specific details about computer vision, robot control, game simulator and robot's behavior.

3.1 Environment

Our Systems are developed with Microsoft Visual Studio 2005. The vision system is based on C++ due to more running speed requirement. Conversely, since the AI system does not require running speed as much as the vision system, it is based on C# for faster development process.

There are many open source library used in our systems. The vision system utilizes OpenCV, an open source Computer Vision library. The simulator system also uses ODE (Open Dynamic Engine) to simulate a game. Moreover, OpenGL (Open Graphic Library) is used by our systems to produce graphics for user interface.

3.2 Vision

In order to provide all positions and directions of robots and ball in the entire field, our vision system combines two image data from two distinct cameras. Since the raw images contain lens distortion, our system transforms the image to remove distortion and then extracts useful data from images using pattern recognition. Finally it sends the data to client AI computers via the local area network (LAN)

Our vision software's functions can be grouped into 5 subjects: color model, calibration, segmentation, pattern recognition, and networking.

3.2.1 Color Model

There are 3 color models related to our vision system. First, our camera received image data from the camera in *YUV Color Model*. Next, we have to convert it to *RGB Model* in order to display it on the monitor. As *HSV Model* can describe perceptual color relationships more accurately than *RGB*, we convert the *RGB* image to *HSV* one for segmentation process.

3.2.2 Calibration

Our vision software requires some camera parameters in order to process the data received from the camera. These parameters are grouped into intrinsic and extrinsic parameters.

First, the intrinsic or internal parameters are the data depending on the camera and lens. There are five intrinsic parameters:

- f	Effective focal length of the pin-hole camera,
- k	1st order radial lens distortion coefficient,
- C_x, C_y	Coordinates of center of radial lens distortion and the piercing point of the camera coordinate frame's Z axis with the camera's sensor plane,
- S_x	Scaling factor to account for any uncertainty in the frame grabber's re-sampling of the horizontal scan line.

Second, the extrinsic or external parameters are based on the position of the camera. These parameters are used to convert the world coordinate to the image coordinate. There are six parameters:

- T_x, T_y, T_z	Translational components for the transform between the world and camera coordinate frames,
- R_x, R_y, R_z	Rotation angles for the transform between the world and camera coordinate frames.

The calibrate process can be separated into 2 steps.

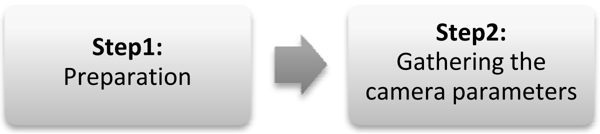


Fig.18 Camera Calibration Steps

In the first step, preparation, the software in not involved. After finishing setting the cameras, we attach 4 calibration plates on the field coinciding with four corners as shown in **Fig.19**. The calibration plate is a rectangular white sheet of paper with 54 circle black dots lined up as a 6 rows × 9 columns table. This step is very important. We must place them carefully; otherwise, the extracted parameters in the proceeding steps will be inaccurate.

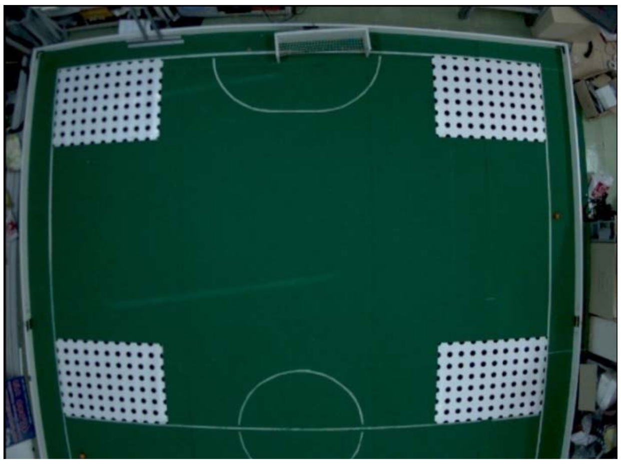


Fig.19 Illustration of preparation step.

In the second step, we obtain the camera parameters using the well-known camera calibration method called *Tsai's algorithm* (4) (5) (6) and a tool for camera calibration, *tclcalib* (7), to extract the parameters. To initialize, the software finds centers of black dots on the calibration plates. Then, it sends these data to *tclcalib* for calculating the camera parameters using Tsai's method and returning them to the vision software. When the software receives the returned data, it calculates them for addition parameters called the rotation matrix *R*, which are use for transforming the real world to the image position by:

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = R \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + T$$

where

$$R = \begin{bmatrix} r1 & r2 & r3 \\ r4 & r5 & r6 \\ r7 & r8 & r9 \end{bmatrix} \quad T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}$$

with

$$r1 = \cos R_y \cos R_z$$
$$r2 = \cos R_z \sin R_x \sin R_y - \cos R_x \sin R_z$$
$$r3 = \sin R_x \sin R_z + \cos R_x \cos R_z \sin R_y$$
$$r4 = \cos R_y \sin R_z$$
$$r5 = \sin R_x \sin R_y \sin R_z + \cos R_x \cos R_z$$
$$r6 = \cos R_x \sin R_y \sin R_z - \cos R_z \sin R_x$$
$$r7 = -\sin R_y$$
$$r8 = \cos R_y \sin R_x$$
$$r9 = \cos R_x \cos R_y$$

We perform this method only once after finishing the camera setup to obtain the camera parameters. The system uses the intrinsic parameters in order to solve the distortion of the lens as shown in **Fig.20**.

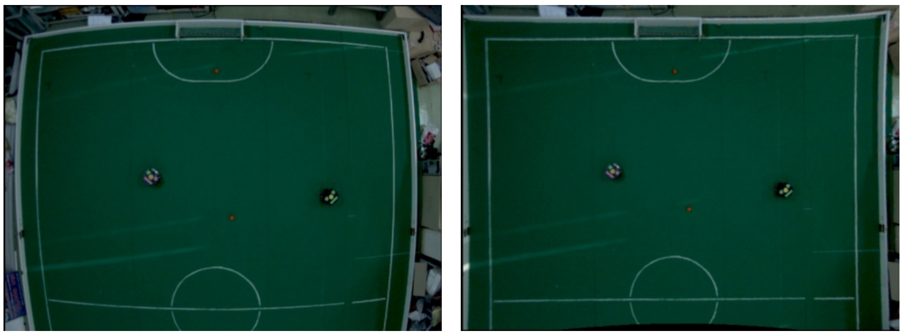


Fig.20 Distorted image (upper) and undistorted image (lower)

3.2.3 Segmentation

Segmentation is a process of partitioning an image into multiple segments. The objective is to simplify or change the original image into another representation which is more meaningful or easier to analyze.

The vision software achieves its goal by starting with scanning the entire original image to collect a position that its color is in the software's color lookup table. These positions tend to be either robots or balls. Then, the software will use the collected position to decide that which section in the image should be undistorted. This makes our system process the image faster than un-distorting the whole image. Finally the software will intensively scan each segment again in order to extract the exact position of the objects.

To initialize the process, we need to set the color range in the color lookup table and the criterion of the objects. First for the color lookup table, we use HSV color model to specify the range of each color. Second, there are three considered object criteria: color, blob size, and eigenvector. Blue and yellow are used for identifying the robots' team while orange is used for the ball and other colors are used for identifying the robots' number. The blob size is the range of pixels indicating the blob we interested. For each extracted data, we have a *confidence value* to indicate how confident the data is. The closer the blob size gets to the criteria, the better confidence value it gets. If the software finds a blob whose size is out of range, it will conclude that the blob is a noise and can be discarded. The last part, eigenvector (8) is used for determining the expansion of blob. As we are interested only in the circle and long rectangle blobs, a blob with longest expansion of eigenvector is a long rectangle; otherwise, it is a circle.

After finishing the initialization, the program extracts a blob that matches the criteria using *the flood fill algorithm* (9) for both rough and thorough scans. This process is critical and time-consuming. The extracted blobs are further used to recognize the objects.

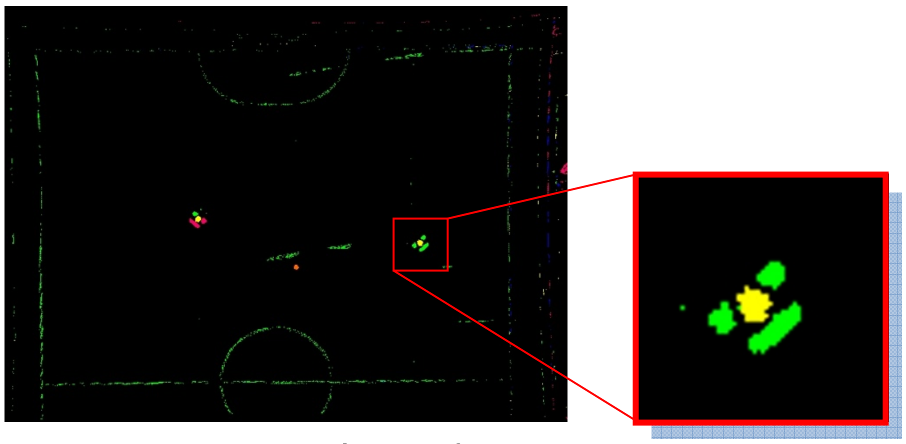


Fig.21 The image after segmentation

3.2.4 Pattern Recognition

The vision software recognizes only our robot's head. It uses each blob for identifying the number of our robots. They are represented in binary format. One blob represents one bit. We use a green blob for representing bit 0, and pink for bit 1. For example, in **Fig.22**, the number of this robot is 001_2 in binary format or 1_{10} in decimal. For opponent robots, the software just finds a center circle blob and assumes that there is no noise blob of which color is the same as the opponent's.

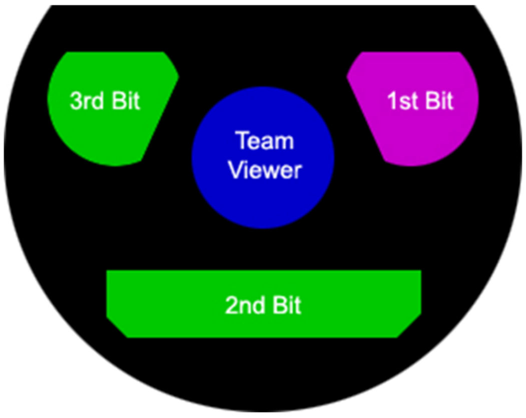


Fig.22 Illustration of colors representing robot number

Although the software can get the robot's and ball's position from the image, it assumes their heights are zero and cannot obtain their accurate positions as shown in **Fig.23**. Thus, the program needs to use the real heights of all objects as the extrinsic parameters in order to calculate their correct positions.

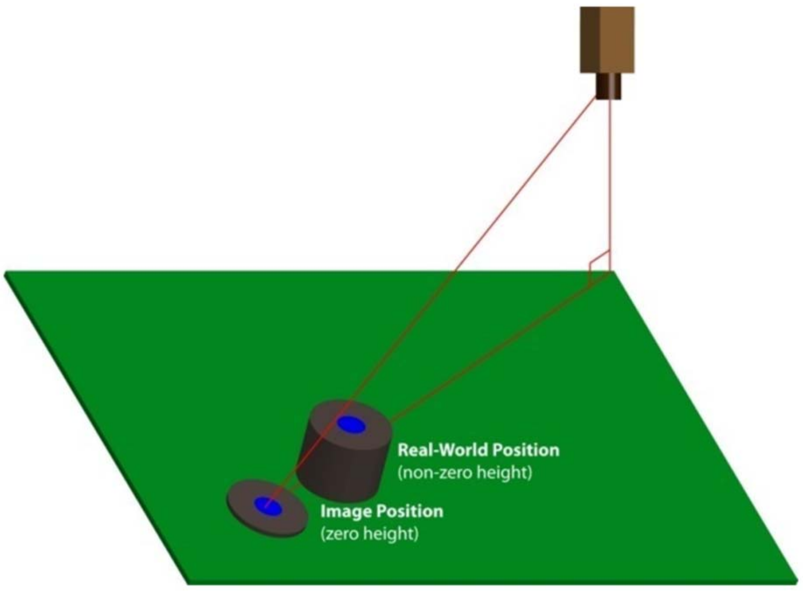


Fig.23 Comparison between zero and non-zero robot's height

3.2.5 Networking

The vision system, working as a server for AI system clients, will open the socket for receiving connection requests from clients. Moreover, this socket also sends the data packets via the local area network (LAN) using *TCP* protocol which guarantees reliability and packet ordering.

The software sends the robots' and balls' data (including position, direction, and confidence level) to the client AI computer every frame after finishing processing the captured image. Hence, the transfer rate is equal to the camera frame rate, approximately 60 frames per second.

3.3 Motion Control

The velocity and the direction of the robot depend on the velocities of all driving motors (V_1, V_2, V_3, V_4). It is significant to control the velocities of all driving motors to meet the velocity and direction requested from AI system. The AI system command contains velocities in x-y dimension, V_{xB}, V_{yB}, and an angular velocity of the robot, ω_{zB}, on a body-fixed frame as shown in **Fig.24**.

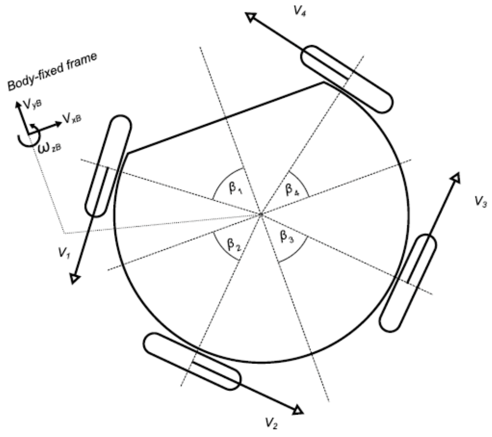


Fig.24 Motion Control

To control the velocity of each driving motor, the main processor must translate the x-y velocities and angular velocity commands into the velocity command of each wheel ($v_{1R}, v_{2R}, v_{3R}, v_{4R}$) by the following linear transformation (10)

$$\begin{bmatrix} v_{1R} \\ v_{2R} \\ v_{3R} \\ v_{4R} \end{bmatrix} = \begin{bmatrix} -\cos(\beta_1) & -\sin(\beta_1) & R_R \\ \sin(\beta_2) & -\cos(\beta_2) & R_R \\ \cos(\beta_3) & \sin(\beta_3) & R_R \\ -\sin(\beta_4) & \cos(\beta_4) & R_R \end{bmatrix} \begin{bmatrix} V_{xB} \\ V_{yB} \\ \omega_{zB} \end{bmatrix} \qquad \textbf{(1)}$$

where R_R is the radius of the robot and β_i, $i = 1-4$, are angles of each wheel given by the mechanical design of the robot. Then, the main processor sends four velocity commands to PID and Driver module in order to make closed-loop control system for the velocity of each driving motor.

3.3.1 Closed-loop Motion Control system

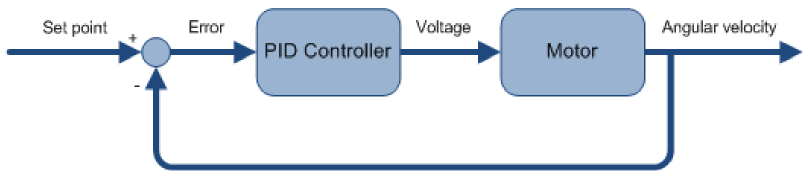


Fig.25 Closed-loop Control system diagram

Closed-loop Control System is an important part of a robot and is included in our design as the *PID & Driver Module*. Its task is to control the velocity of each motor. A velocity command pre-calculated by the main processor is the set point value of the PID & Driver Module.

The velocity of each wheel is measured by a quadrature encoder. This real velocity will be fed back to compare with a set point value. Then, the error will be an input of the PID controller for adjusting an appropriate duty cycle of the pulse (Pulse Width Modulation) sent to a motor driving circuit. This cycle is iterated until the system is shut down.

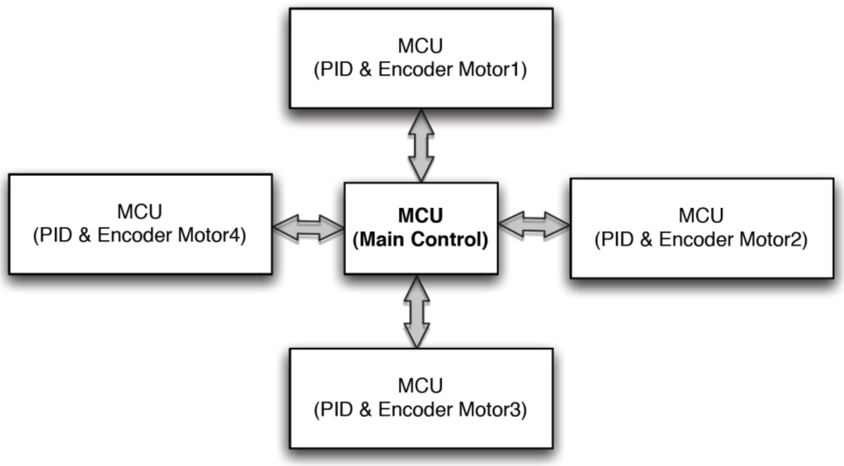


Fig.26 The previous version of system used multiple processors.

In the previous works, some hardware designers utilized off-the-shelf controller, such as LM629, to setup a PID control loop for each individual motor. The drawback of such approach is the unbalanced synchronization of the controllers because the programmer has to send a series of commands to each controller individually. Thus, there will be a time lag in the system depending on the number of controllers. Therefore, PID & Driver Module is constructed so that it can perform all tasks in parallel for PID control and encoder module for each wheel of the robot (11) (12) (13)

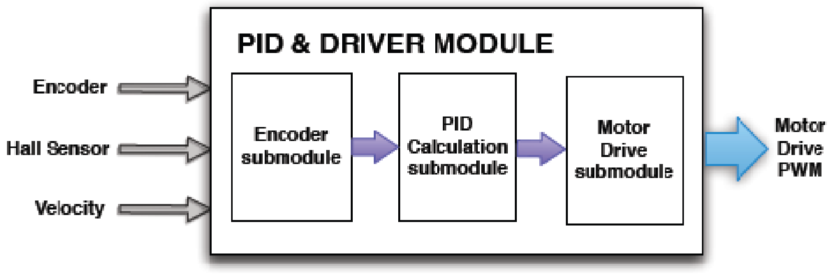


Fig.27 PID & Driver Module

PID & Driver Module is composed of three submodules as shown in **Fig.27**. The first submodule is the Encoder Submodule, which is the module that counts pulse signals from an encoder using a counting register. The second one is the PID Calculation Submodule which implements the PID controller. It calculates the velocity of a wheel by using the number of pulse signals presented in the counting register of the Encoder Submodule. Then, this module uses the result of the calculation and set points from the main processor in order to calculate an appropriate duty cycle of the PWM for a driving motor. The last one is the Motor Drive Submodule which drives the driving motor. Since the motor is a 3-phase BLDC motor, it is important to know the next phase to apply voltage in order to create a maximum torque. This can be determined by hall sensor signals so that the Motor Drive Submodule can apply voltage to the driving circuit with the correct phase of BLDC motor.

3.4 Kick Control

There are two modes for kick control command: flat kick and chip kick. A kick power level for each mode ranges from 0-255. Once a mode for shooting is selected and a kick power level is calculated, the AI system sends the kicking command to the robot. After getting IGBT gate opening time, kicking mode and charging command from AI system, the main processor in the robot sends these parameters to the *Shooting module*. IGBT gate opening time indicates pulse generating time to IGBT gate to enable current flow from the capacitor to the solenoid. Kicking mode determines which solenoid, either flat kick solenoid or chip kick solenoid, is the destination of the generated pulse signal. Charging command switches the MOSFET to boost circuit in order to start charging the capacitor.

AI system can choose between 2 kick modes: *wait kick* mode and *forced kick* mode. In the wait kick mode, kicking occurs only when the ball is detected by ball IR detector. On the other hand, in the force kick mode, kicking takes place immediately.

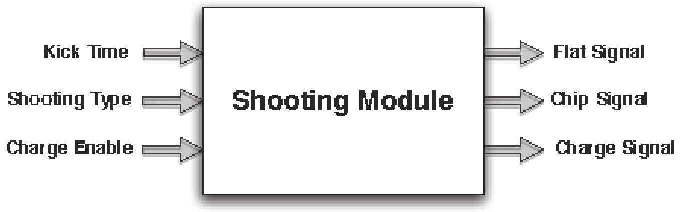


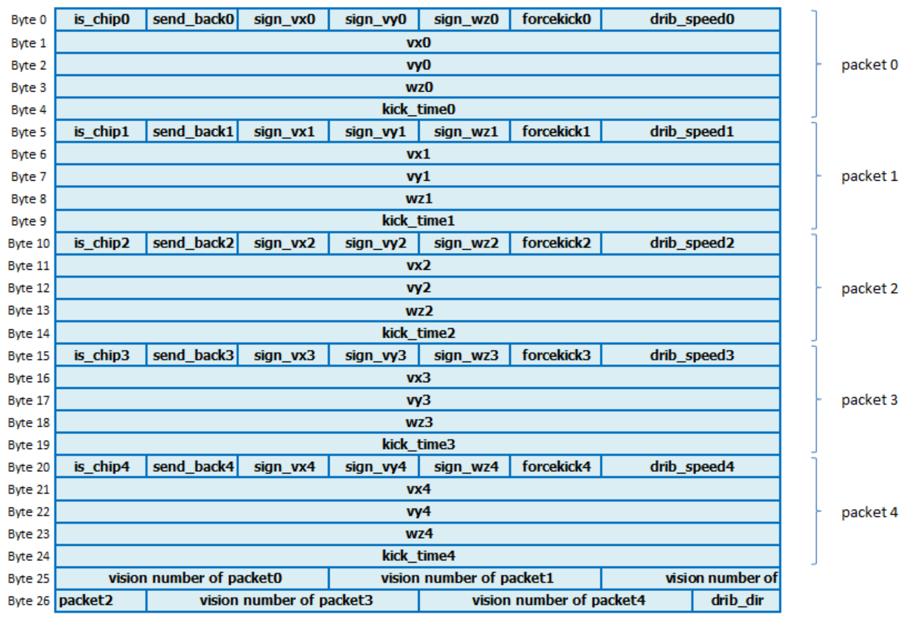
Fig.28 Shooting module

3.5 Dribbler Control

Dribble Module is the open loop control system for driving the dribble motor. Because the dribble motor is a BLDC motor, it is important to apply PWM signal to the driving circuit using the correct phase determined by the hall sensor signals. Closed-loop Control system is not necessary for this application because the aim is only to hold the ball, not controlling the angular velocity of the motor. A fixed speed for rotation is enough to hold the ball near the robot. Dribble Module provides four levels of angular velocity to AI system which, in turn, chooses the level of angular velocity and the direction of rotating, and then sends that command to the robot.

3.6 Wireless Communication Control

AI System uses radio frequency to broadcast commands to the robots. The commands of all robots are packaged in one serial data chunk, as shown in **Fig.29**. *RF Receiver Module* in the FPGA accesses the buffer in the RF device to get the package sent from AI and, then, sends them to Main processor module. Because the package contains all commands for not just one but all robots, each robot must choose the packet in the whole package to obtain its own command. The Vision Number of Packet field in the 25th and 26th bytes of the package is used to select the robot for that packet. For example, the vision number of packet0 is 2. It means that the packet0 belongs to robot number 2. Before sending the package to the main processor module, the CRC Checker submodule in RF Receive Module also detects communication errors of receiving package.



Byte 0	is_chip0	send_back0	sign_vx0	sign_vy0	sign_wz0	forcekick0	drib_speed0		packet 0
Byte 1	vx0								
Byte 2	vy0								
Byte 3	wz0								
Byte 4	kick_time0								
Byte 5	is_chip1	send_back1	sign_vx1	sign_vy1	sign_wz1	forcekick1	drib_speed1		packet 1
Byte 6	vx1								
Byte 7	vy1								
Byte 8	wz1								
Byte 9	kick_time1								
Byte 10	is_chip2	send_back2	sign_vx2	sign_vy2	sign_wz2	forcekick2	drib_speed2		packet 2
Byte 11	vx2								
Byte 12	vy2								
Byte 13	wz2								
Byte 14	kick_time2								
Byte 15	is_chip3	send_back3	sign_vx3	sign_vy3	sign_wz3	forcekick3	drib_speed3		packet 3
Byte 16	vx3								
Byte 17	vy3								
Byte 18	wz3								
Byte 19	kick_time3								
Byte 20	is_chip4	send_back4	sign_vx4	sign_vy4	sign_wz4	forcekick4	drib_speed4		packet 4
Byte 21	vx4								
Byte 22	vy4								
Byte 23	wz4								
Byte 24	kick_time4								
Byte 25	vision number of packet0			vision number of packet1			vision number of		
Byte 26	packet2	vision number of packet3			vision number of packet4			drib_dir	

Fig.29 The format of package sending in wireless communication system

The meaning of each field in **Fig.29** is described in the **Table 2**.

Table 2 Format description of communication data

field	Size (bit)	description
is_chip	1	indicate which kick to perform: flat kick or chip kick
send_back	1	request the data from the robot but still unused yet
sign_vx	1	indicate the sign of vx0
sign_vy	1	indicate the sign of vy0
sign_wz	1	indicate the sign of wz0
forcekick	1	indicate kick mode: wait kick or force kick
drib_speed	2	control the angular velocity of the dribbler motor between four levels
vx	8	value of the velocity command in x axis.
vy	8	value of the velocity command in y axis.
wz	8	value of the angular velocity command in z direction.
kick_time	8	indicates pulse generating time to IGBT gate
vision number of packet	3	match the robot to that packet.
drib_dir	1	indicate the direction of dribble motor rotation

3.7 Simulator

In the early phase of play and strategy testing, sometimes it takes too excessive cost and overhead time to setup the testing environment. Therefore, testing plays on a simulator will let the developer focus on the test rather than setting up testing environment. After the play performing well on the simulation, we proceed to test it in the real environment.

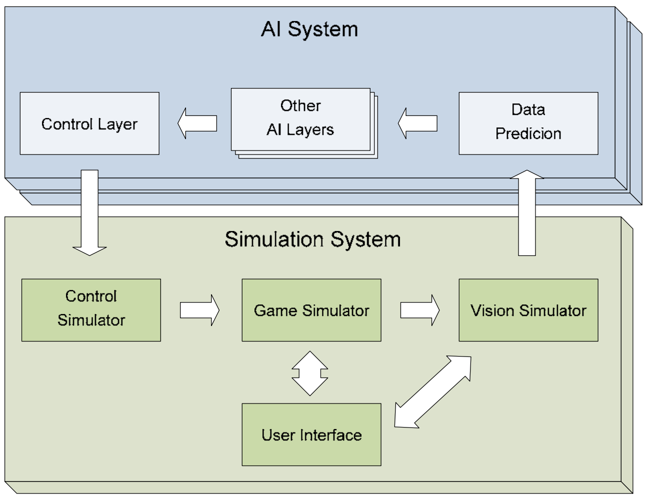


Fig.30 Overview of Simulation System – AI System Relationship

Our simulator is developed with Microsoft .Net Framework, mainly C#. With .Net Framework, our system can be developed using OOP approach, using C#. Meanwhile, we can also make utilization of open-source library such as Open Dynamic Engine (ODE), an open source high performance library for simulating rigid body dynamics.

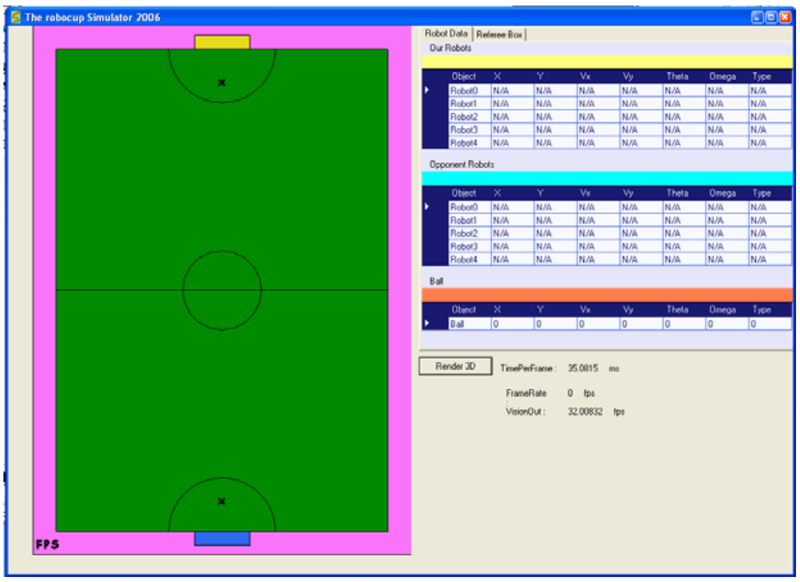
Our simulator system consists of Control Simulator, Game Simulator, Vision Simulator and User Interface.

First, the Control Simulator receives control data packages from AI System and translates them into input data to the Game Simulator.

Next, the Game Simulator is developed using the ODE library. Initially, we define a new world by defining related object, e.g. robots, ball, soccer field, goals and field boundaries. Each defined object is defined as objects from the ODE library which include its characteristics including mass, shape. Afterward, we also define how force is applied to each object in each action. For example, we have to define how force is applied to the ball when the robot shoots with each specific power level. With these definitions, the ODE can simulate the game.

Subsequently, the AI Systems needs Vision data so as to plan, calculate and control the robot. Hence, we have a vision simulator whose duty is to convert the robot and ball positions from simulated world into vision data packages. We also provide optional delay to provide more realistic vision data for the test.

Finally, our simulation system allows human to view output vision data. Moreover, there is also a Referee Box Simulator for sending referee signal to the AI system.



Our Robots

Object	X	Y	Vx	Vy	Theta	Omega	Type
Robot0	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Robot1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Robot2	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Robot3	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Robot4	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Opponent Robots

Object	X	Y	Vx	Vy	Theta	Omega	Type
Robot0	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Robot1	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Robot2	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Robot3	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Robot4	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Ball

Object	X	Y	Vx	Vy	Theta	Omega	Type
Ball	0	0	0	0	0	0	0

```
TimePerFrame :   35.0815   ms
FrameRate        0         fps
VisionOut :      32.00832  fps
```

Fig.31 Screenshot from Simulator UI

3.8 Behavior

3.8.1 Path Planning and Collision Avoidance

Our path planning system is based on the Force Field approach. The goal of the system is to find a collision-free path from the start position to the target position. In order to avoid collision, the obstacles such as robots and walls are taken into the system as repulsive forces. A ball is sometimes considered as an obstacle when the robot performs ball avoidance. In addition, the attractive force pointed toward the goal position is summed up to attract the robot. The path planning system obtains the desired path by calculating a resultant force along the path. However, there is a problem with this approach. It is a local minimum which is the point that the resultant force approaches zero and traps the robot before reaching its goal position. In order to avoid a local minimum, we use a pattern of force field around the robot preventing the goal point as depicted in **Fig.32**. Additionally, when there are several obstructive robots, they are grouped as obstacle and a similar pattern of force field is then applied as shown in **Fig.33**.

Currently, there are many parameters that are tuned in a brute-force manner. However, our system provides a collision-free path that the robot can follow properly without a local minimum problem.

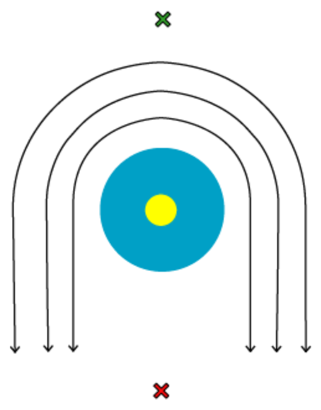


Fig.32 A force field pattern around an obstacle

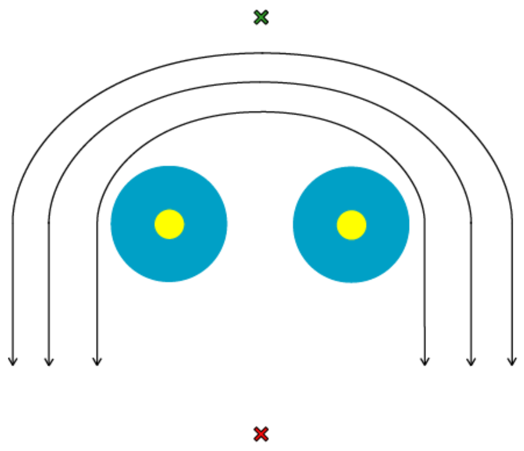


Fig.33 A force field pattern for a group of obstacles

3.8.2 Passing

When a robot passes the ball to another robot, there are many factors to be considered. For examples, are there any opponent's robots around, or is our teammate's robot ready to receive the ball? If the certain conditions are met, robot will perform the passing. Otherwise, our AI system will consider alternative options.

One of the easy but effective ways to calculate the position to pass the ball is to use the position of the receiver's robot plus the offset of the radius of the robot in the direction of the receiving robot as shown in **Fig.34** where a red cross indicates the receiving position.

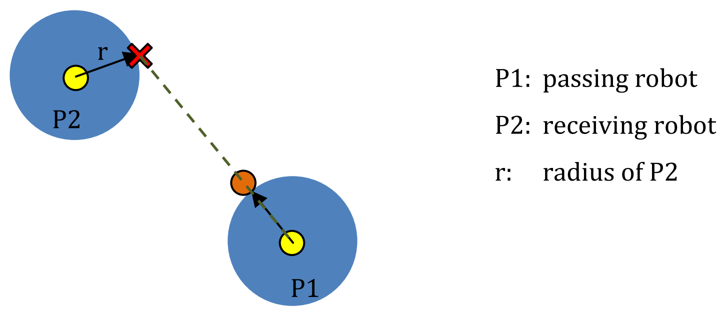


Fig.34 Calculating the receiving ball's position

After the receiving position is calculated, the distance between the ball and its desired position can be calculated. This distance is used for selecting the kick power. Then the system can perform the passing by turning the passing robot to the correct direction (using receiving ball's position) and kicking the ball with the calculated kick power.

3.8.3 Kicking

Since our robot has two types of kicking: flat-kick and chip-kick, we have to consider which type to be performed. Basically, we use the flat-kick when there is no blocking opponent's robot or if we want to kick the ball with a very high speed, especially for shooting. Conversely, chip-kick is used when the shooting direction is blocked by one of opponent's robots or when the speed of the travelling ball is not required.

Another kicking technique is to turn on the dribbler and the kicking mechanism simultaneously in order to center the ball. As a result, the direction of the ball is more accurate to the desired position than without the dribbling.

The main parameter that can be changed for each kick is the kick power. For flat-kick, kick power determines how fast the ball will travel while for chip-kick, kick power determines how far or how high the ball will be after being kicked.

3.8.4 Dribbling

In our program, there is no specific skill for just dribbling, but we will do it along with other skills. To dribble is just to turn on the dribbler and move to the desired direction. One thing to be kept in mind is that the allowable robot speed depends on the quality of the dribbler. If the dribbler works very well, robot can move faster. Otherwise, robot might have to slow down its speed in order to avoid losing the possession of the ball. Using the dribble near team's defense area is also risky. Since the dribble spins the ball backward, if the robot loses the ball, it is highly probable that the ball will move toward into our own team's goal.

Even though the main use of dribbler is to control the ball, its usage can be various. These include taking the ball from opponents, controlling the ball while dribbling it for a short distance, or even guiding the direction of the ball before being kicked.

3.8.5 Roles & Positioning

Adapted from traditional football plays, which have four major players' roles: Goalkeepers, Defenders, Midfielders and Attackers; our robots have three major roles: Goalkeeper, Defenders and Attackers. Due to less number of players in the pitch, including Midfielder Robots would add unnecessary complexity in our system since attacker, defender and goalkeeper are sufficient for strategy.

Each robot's role will be assigned manually at the beginning of the game. One reason that we do not allow robots to switch its role during the game is that the switching condition is very subtle. As a sample situation, the attacker is about to get the ball from the opponent, however, at the same time, there is a small gap for the opponent to shoot the ball to our goal. That attacker robot might move back to narrow down the gap. This could mean that we have missed a good chance to steal the ball from the opponent.

The main job for attackers is trying to possess the ball and shoot the ball into the opponent's goal. One of the attackers will try to get the ball while the other two will move around and be ready to receive the ball if passing technique is being used.

The main job for defender and goalkeeper is trying to prevent the ball from entering into our goal. The defender will closely cooperate with the goalkeeper in order to prevent the ball from entering our goal.

3.8.6 Strategies

In the world of Traditional Football, there is a well-known strategy: "attacking is the best defense." Our team strongly agrees to that theory. Accordingly, we prefer to use 1-1-3 formation. In other words, there are one goalkeeper, one defender and three attackers. The goalkeeper works cooperatively with the other defender to narrow down the shooting angle, clear the ball away from risky area or do any other things that will ensure our goal's safety By the way, the attackers mainly works cooperatively to score a goal. However, if the opponent possesses the ball, our attackers will help the goalkeeper and defender by tackling, stealing the ball, or marking enemy robots.

We have many different plays that match various situations. For instance, we have plays for offensive and defensive corner kicks, direct free kicks, indirect free kicks, normal play, etc. There are several plays that can be used in the same situation. The decision whether to perform which playing pattern is highly related to its effectiveness. There is a parameter to indicate the effectiveness for each play. Our play selector has more chance to select higher valued play. These effectiveness values are manually edited from previously recorded statistics. As a future development, we plan to record the game stats so that these parameters can be automatically adjusted after each game or even during the game

4) Conclusions

After many years of research and development, we have faced various kinds of problems and also found some good solutions for those. The experience we gained results in the improvement of our team in the competitions at both national and international levels. Nevertheless, we will keep developing our system for even better performance.

Reference

1. AVT Stingray F046B/C Fiber DataSheet. [Online] AlliedVisionTec. http://www.g4.com.tw/web/file/product/demo/Stingray_DataSheet_F046BC_fiber_V1.0.0_en.pdf.
2. Altium Nexar TSK51. [Online] http://www.keil.com/dd/chip/3730.htm.
3. [Online] http://robocup.mi.fu-berlin.de/buch/omnidrive.pdf.
4. *A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses.* **Roger, Tsai Y.**, IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, pp. 323-344, August 1987.
5. *An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision.* **Roger, Tsai Y.**, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. pp. 364-374, 1986.
6. **Dias, Paulo.** Tsai Camera Calibration. [Online] 11 5, 2003. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/DIAS1/.
7. **Davis, James.** tclcalib - A tool for camera calibration. [Online] http://www.soe.ucsc.edu/~davis/projects/tclcalib/.
8. Eigenvalue, eigenvector and eigenspace. *Wikipedia, the free encyclopedia.* [Online] http://en.wikipedia.org/wiki/Eigenvector.
9. Flood fill. *Wikipedia, the free encyclopedia.* [Online] http://en.wikipedia.org/wiki/Flood_fill.
10. *Accuracy Improvement of Omni-Directional Mobile Robot using Gyroscope and Acelerometers.* **Sirichai Pornsarayouth.** M.Eng. Dissertation, Chulalongkorn University, Bangkok, Thailand, 2008.
11. *FPGA Implementation of Closed-Loop Control System for Small- Scale Robot.* **Wei Zhao, Byung Hwa Kim, Amy C. Larson and Richard M. Voyles.** *Proceedings of the International Conference on Advanced Robotics*, pp. 70-77, 2005.