Skuba 2010 Extended Team Description

Piyamate Wasuntapichaikul[1], Jirat Srisabye[1], Chanon Onman[1],
Supparat Damyot[2], Chinatun Areeprasert[2] and Kanjanapan Sukvichai[3]

[1] Department of Computer Engineering
[2] Department of Mechanical Engineering
[3] Department of Electrical Engineering
Faculty of Engineering, Kasetsart University
50 Phaholyothin Rd., Ladyao, Jatujak, Bangkok, 10900, Thailand
baugp@hotmail.com, jiratto@gmail.com, nobitaa@hotmail.com,
leksupp@hotmail.com, apeeranut@hotmail.com, fengkpsc@ku.ac.th

Abstract. This paper presents a detailed description of Skuba, a Small-Size League RoboCup robot team in addition to the team description paper. The robot system is designed under the RoboCup 2010 rules in order to participate in the RoboCup competition in Singapore. The system consists of several components which are explained in each section.

1 Introduction

Skuba is a small-size league soccer robot team from Kasetsart University, which has entered the RoboCup competition since 2006. We got the championship last year from the RoboCup 2009 in Graz, Austria and another championship in December from RoboCup China Open 2009 in Dalian, China.

The robot system consists of two main components: the robot hardware and the software. The software makes strategic decisions for the robot team by using information about the object positions from the vision system. The global vision system run by the shared vision software, SSL-Vision, uses two cameras mounted over field. The software executes plans by calculating the robot actions and then sends the commands to each robot.

Our team has ten identical robots, six of them were built in 2008 and another four were built in 2009 with some minor changes in material and mechanical design. We are not planning to make any major changes to the design. The robot hardware is the same as used in last year.

This year, the main focus of our development is the automatic calibration. Even though every robot is built by the same design and material, there are still some errors from the manufacturing and assembling process. Furthermore, some parameters can be changed according to the competition environments. These issues involve the need of calibration for the accuracy of the system. The kicker and the low level controller parameters used to be manually calibrated. These calibrations are time consuming and need manpower to do the experiments. The automatic calibration process simply uses the same procedure as the manual calibration does, but it’s done automatically by the software.

1.1 Team Members

Kanjanapan Sukvichai : Control Theory, Supervisor
Piyamate Wasuntapichaikul : Electronics, Firmware, Team Leader
Chanon Onman : AI Software
Phumin Phuangjaisri : Electronics
Chinatun Areeprasert : Mechanics
Tanusak Kathongthung : Mechanics
Nuttapol Runsewa : AI Software
Khakhana Thimachai : AI Software
Teeratath Ariyachartphadungkit : Apprentice
Krit Chaiso : AI Software

2 Robot Electronics

This section describes the robot electronics system including the designs and components. Details about operations and algorithms are in the firmware section.

The robot consists of two electronics boards: the main board and the kicker board. The main board handles all of the robot tasks except kicking. The kicker board controls the entire kicker system.

2.1 Main Electronics Board

The board consists of a Xilinx Spartan-3 XC3S400 FPGA, motor driver, user interface, some add-on modules and debugging port. The microprocessor core and interfacing logic for external peripherals are implemented using FPGA in order to handle the low-level control of the brushless motor such as velocity and position control. The main electronics board receives commands from the main software on a computer. The board integrates the processing components together with the power components to keep the board compact and minimize wiring. With limited space, almost components are in small SMD packages. However, these components still large enough for hand soldering with conventional tools. Figure 1 show the main electronics board of the robot.

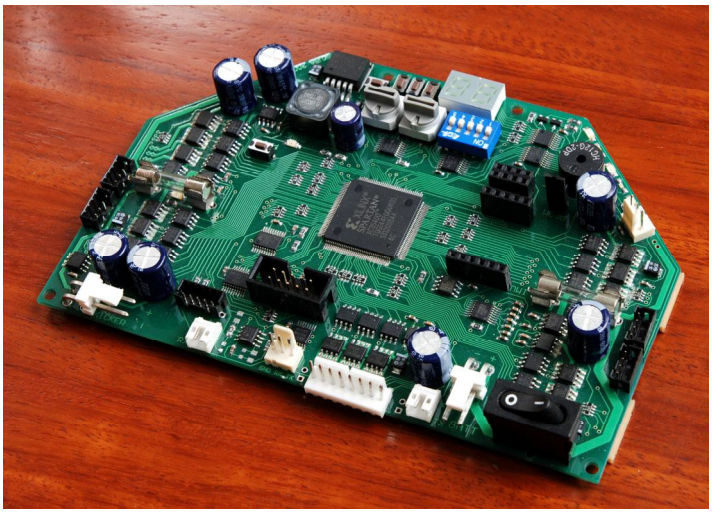

Fig. 1. The main electronics board.

2.2 Battery and Power Supply

Each robot uses 4-cell lithium polymer battery with capacity around 1700-2200mAh as a power source. The robot can run for several hours with this battery pack. There are three main power lines in the robot: kicker board, motor driver and processing components. These lines are protected with different current rating fuses. The power supply current is monitored by the current sensors which are attached to each motor driver and kicker board to limit the current when short-circuit occurs.

2.3 Motors

There are two types of motor in the robot, the driving motor and the dribbling motor, both are brushless motor. Each driving motor is a 30 watts Maxon EC45 flat motor with a custom back-extended shaft for attaching encoder wheel. The motor itself can produce a feedback signal from hall sensors for measuring wheel velocity. However, this multi-pole motor sends only roughly 48 pulses per revolution; therefore, this motor is equipped with an US Digital E4P encoder which have higher resolution of 1440 pulses per revolution. The dribbling motor is a high speed 15 watts Maxon EC16 motor. Despite a very low resolution of 6 pulses per revolution signal from hall sensors, the implementation of the PI controller is possible when running this motor at high speeds. The maximum speed of the dribbling bar is about 13000 rpm.

The motor driver is a three phase inverter circuit using complementary N and P channel power MOSFET in each phase. This configuration doesn't require bootstrap driver as in N-channel-only configuration. These MOSFETs are driven by MOSFET driver ICs to minimize switching loss. The motor commutation and PWM generation are described in the firmware section. Figure 2 shows the three-phase brushless motor driver circuit.

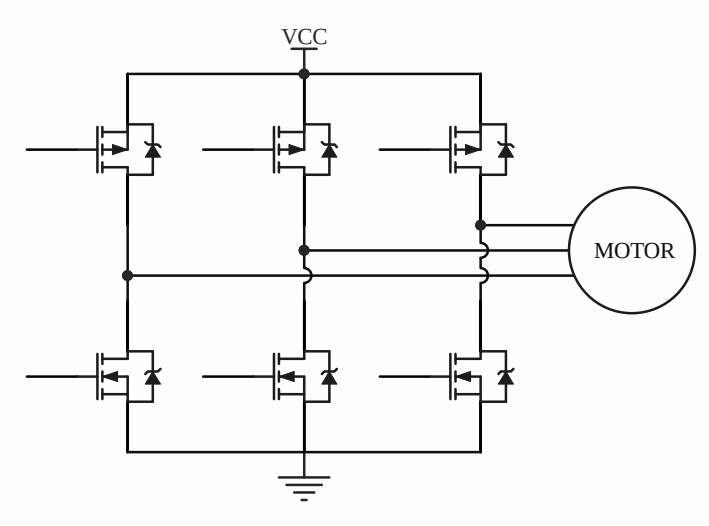

Fig. 2. The three phase inverter in complementary configuration.

2.4 Kicker

Both kicker and chip kicker in the robot are energized by solenoids. These solenoids are fed with a high current impulse to produce intense magnetic field in a short period of time. The impulse current is created from discharging the high voltage capacitor into solenoid wire. Charging and discharging the capacitor are controlled by the kicker board. This board consists of power switching devices: MOSFET and IGBT which are controlled from the main board.

The charger is a switching DC boost converter circuit. The RC snubber is added into the switching node in order to reduce ringing and EMI which are created by the parasitic inductances and capacitances [1]. With current design, the kicker board can charge two 2700 μF capacitors from 0V to 250V in about 5 seconds with 2A average current.

The kicker is a cylindrical shaped solenoid attached to a curved kicking plate and the chip-kicker is a flat shaped solenoid attached with a 45 degree hinged wedge. These solenoids are driven by IGBTs and the kicking force is controlled using PWM signal. The kicker board is depicted in Figure 3.

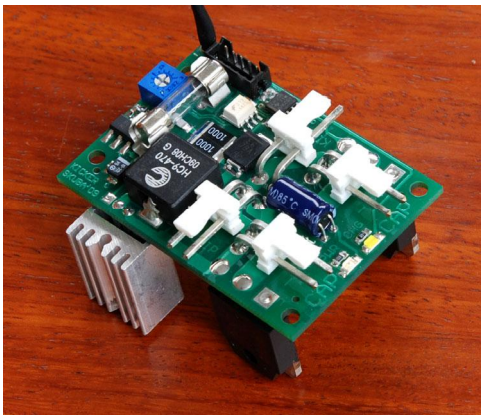

Fig. 3. The kicker board.

2.5 Ball Sensor

The infrared break beam in front of the kicker is used to detect the presence of the ball. This sensor is useful in the passing and one-touch shooting. These robot skills use this sensor as a trigger for the kicker, since the vision cannot detect the ball location accurately.

2.6 Communication

The communication between robots and the computer can be made by using a radio device. A bi-directional wireless module is operating at 2.4GHz frequency in ISM band which offers channel switching around 2.4GHz to 2.5GHz for the communication. This wireless module consists of a wireless transceiver IC from Nordic Semiconductor: nRF24L01+.

Each robot communicates with an external wireless board which is linked to the computer via a USB port. This board consists of two independent wireless modules which can provide a full-duplex communication.

2.7 Debugging

The main board has switches, LEDs and buzzer to provide debugging capabilities. These components are connected to the FPGA using shift registers to minimize FPGA input/output pin, only four signal lines are used. The robot can be manually controlled to do some tests for calibrating and setting parameters such as the kicker force and sensor compensation. These parameters are also saved inside an onboard non-volatile memory. This memory also used as a storage for capturing encoder, current or other information which can be downloaded via a serial port. This information is very useful for PI controller tuning and sensor calibration.

3 Robot Firmware

The main electronics board consists of a FPGA as a single chip central controller. The FPGA is embedded with a 32-bit processor, brushless motor controller, PWM generator, quadrature decoder, kicker board controller and onboard peripheral interfacing cores: SPI and UART. The processor runs at 30MIPS as same as oscillator clock speed. We use Altium Designer and Xilinx ISE software to generate, configure and debug these cores.

3.1 Brushless Motor Driver

The three phase inverter bridge is fed with signals from FPGA to provide commutation for each motor. These signals are ANDed with the PWM signal to vary the average voltage applied to the motor winding. The six steps commutation sequence is detected by three hall sensors in the motor. Both high and low side drivers are driven by PWM signals to control the torque applied to the motor.

3.2 Motion Control

The robot employs a PI controller as a motion controller, one controller for each motor. The control loop executes 600 times per second using velocity feedback from the encoder in driving motor and hall sensors in dribbling motor. The proportional and integral gains are manually hand-tuned. The computer sends a velocity for each DOF: x-y axis and rotation axis. Then, converted to each wheel velocity and sent to the PI controller. The output from the controller is sent directly to the PWM controller.

3.3 Over-current Protection

General problem when driving the inverter bridge is the shoot-through current. This current is caused by turning on one side of the driver immediately after the other side of the driver has been turned off, because the MOSFET turn-off time is usually higher than the turn-on time. This situation occurs when the motor is reversing direction, which can be prevented by adding a small delay time between each high and low side driver signal.

Many of robot skills use the dribbler. Some ball stealing skills can cause dribbling motor to stall when the dribbling bar is contacted with the opponent robot. The stalled motor consumes very high current and often burn the fuse out. This over-current situation can be detected by a current sensor and can be prevented by limiting a PWM duty cycle until the current drop below the safe motor operating current. Figure 4, depicts the motor stalling situation. When the motor stalled, the motor current increased and dropped in a short time due to limited duty cycle. The motor current is controlled around the threshold while the motor is stalling.

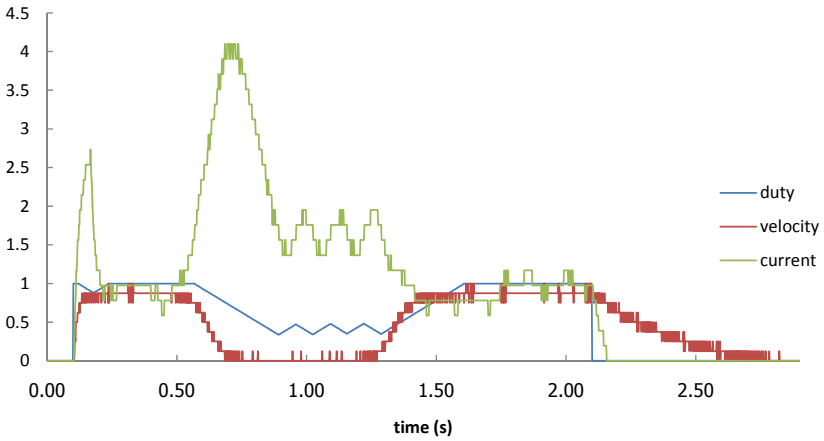

Fig. 4. PWM duty cycle, velocity and current. The motor is run from stop. Then, a very large load applied to the motor in between 0.5 sec to 1.5 sec. Current value is in ampere, duty cycle and velocity are normalized.

3.4 Kicker

Kicker board consists of power electronic components which are controlled directly from the FPGA in the main board. The board requires PWM signal for switching circuit and another PWM signal to impulse the kicker with the desired kicking force. The switching DC converter uses a soft-start method to reduce inrush current when the capacitor is empty. This method is done by starting with the low duty cycle and ramping up over the time until it reaches the limit. The ramping starts again when the kicker is activated. Details about operation and implementation of a similar circuit are in [2].

3.5 Communication

The robot commands are sent to each robot for each frame of software execution. Then, each robot sends back the status after received its commands immediately. The commands are in small packet containing velocities, dribbler and kicker command with some headers for testing and calibrating purpose. The robot status contains battery level and sensors information.

4 Robot Mechanics

This section describes the mechanical system of the robot which consists of the driving system, ball control system and kicking system.

The robot has a diameter of 176 mm and a height of 147mm.The dribbler covers up to 20% of the ball diameter. The 3D CAD model of the robot and the real robot are shown in Figure 5 and 6 respectively.

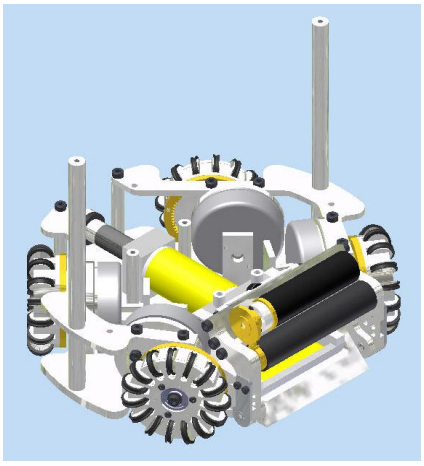

Fig. 5. 3D CAD model of the robot**.**

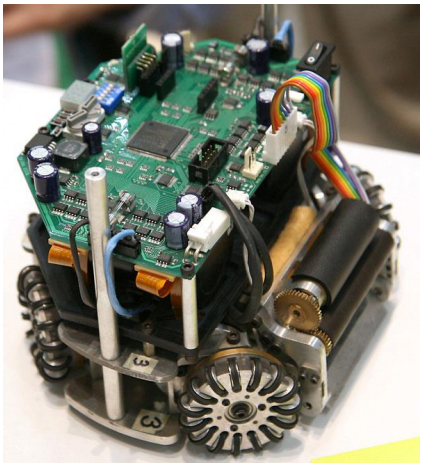

Fig. 6. Real robot.

4.1 Wheels

The robot has four omni-directional wheels. Each wheel has a diameter of 50.8 mm. The wheel cover is made from aluminum and its base part is made from polycarbonate, the light weight material. There are fifteen small rollers per wheel. Double seal o-ring is used for each roller in order to get more friction. This wheel provides enough friction to drive the robot with 3.5 m/s^2 of acceleration and 5 m/s of velocity. The robot wheel is shown in Figure 7 (CAD model) and 8 (real wheel).

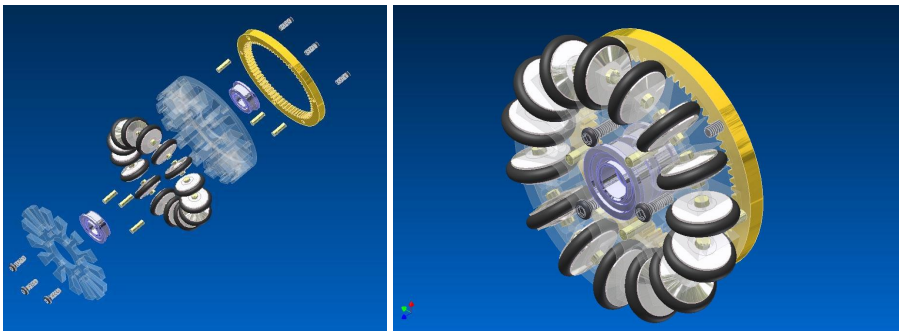

Fig. 7. 3D CAD model of the omni-directional wheel.

Fig. 8. The omni-directional wheel.

4.2 Driving System

The robot uses brushless motor, 30 watts Maxon EC45 flat, in the driving system. The driving system uses gear ratio of 3.6:1 (72:20). This gear ratio can provide the satisfying acceleration and velocity with the specified wheel diameter. The 5 mm-thick bottom plate connects all of the robot parts together. The robot's partly assembled chassis is shown in Figure 9.

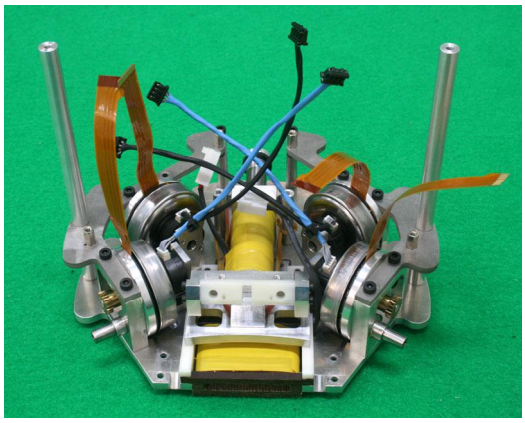

Fig. 9. The robot's chassis with four brushless driving motors.

4.3 Ball Control System

4.3.1 Suspension System

The entire dribbler assembly is hinged with the chassis plate which is attached with a sponge damper. We use adjustable screw as the stopper, allowing the suspension to swing about 6.5 degree. This suspension system makes the robot able to receive the fast moving ball in passing skill. Both outer sides of the suspension arms are equipped with the covers to protect the infrared sensors from damaging. Figure 10 and 11 shows the suspension system of the robot.

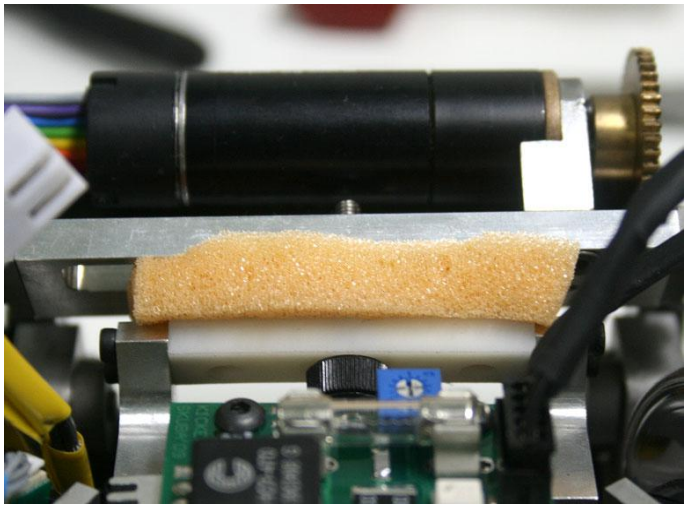

Fig. 10. The suspension with a sponge damper.

Fig. 11. The dribbler assembly.

4.3.2 Dribbling System

Suspension and dribbling system are both necessary for controlling the ball. The 15 watts Maxon EC16 is used as a dribbling motor. The 5.4:1 planetary gearhead is attached to this motor. The dribbling bar can spin at the maximum speed of 13000 rpm. The dribbling bar is made from aluminum rod with a diameter of 10 mm as shown in Figure 11. The dribbling bar is covered with a silicone tube which has a good property to spin the ball firmly.

4.4 Kicking System

4.4.1 Kicker

The kicker is energized by solenoid system. It is the cylindrical solenoid wound with seven layers of 23AWG enameled wire. The kicking plunger rod is separated into two parts. The first part is magnet part which is made from steel and the second is made from material with no magnetic property, aluminum. Both rods have the diameter of 11 mm. These rods are joined together and attached with the curved aluminum kicking plate. This plate has a contacting radius of 300 mm which results in more accurate shooting when the ball is not in the center of the kicker. The kicking system makes the robot able to kick the ball at maximum speed of 12-14 m/s.

4.4.2 Chip Kicker

The chip kicker uses the flat solenoid which is made from glass reinforced epoxy wound with five layers of 24AWG enameled wire. This solenoid is placed in the front part of the bottom plate. The flat plunger is steel with the thickness of 3.75 mm.

The chip-kicker is a hinged wedge which swings around the pivots. Pin is used as a pivot rather than the bearings because it has more endurance. The swinging degree is also limited by the other pins. The chip-kicker is made from 7075 aluminum alloy, which has more strength than the standard aluminum. It has a 45 degree slope at the front as the contact point. This chip kicker has an ability to chip the at a maximum distance of 7.5 m.

5 Software Architecture

The overall software architecture is illustrated in Figure 12. The software consists of several modules organized as a multilayer architecture. This software has been being continuously developed since RoboCup 2006 based on the strategy structure of Cornell Big Red 2002's software.

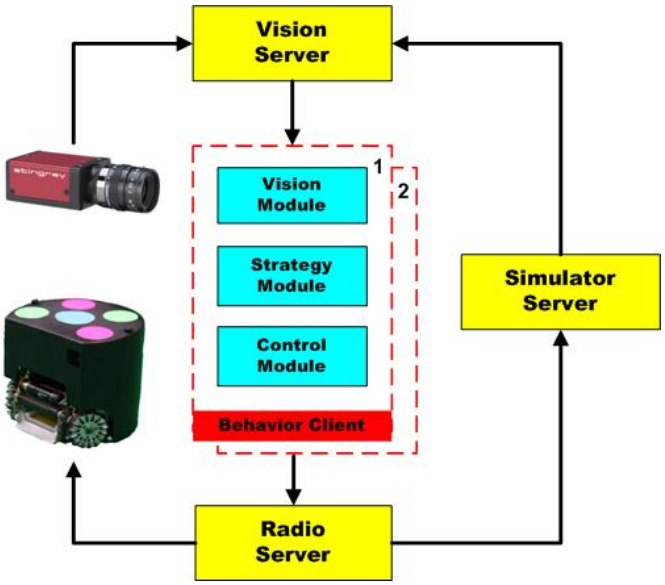

Fig. 12. The software architecture.

5.1 SSL-Vision

The use of shared vision system named SSL-Vision is required by the competition rule. This new vision software can be integrated into the system by simply replacing our Vision Server software. With some code changes in the vision protocol, the existing software works with the shared vision system successfully. The SSL-Vision also provides geometric parameters which are very useful for the chip-kicker calibration.

5.2 VisionServer

The integration of all cameras' information is done on the VisionServer process. The information from two cameras which comes from overlapped regions is integrated. Each detected object is tagged with the confident value based on confident rules. When the information from two cameras are inconsistent, for example, when there are more than one of the same type of object from different camera, the server decides on the higher confident value. Finally, the vision server sends the consistent information of the whole situation to the BehaviorClient.

5.3 BehaviorClient

The BehaviorClient has three main modules. At the First VisionModule receives vision information from VisionServer or SimulatorServer and predicts that information to account for latency. Then StrategyModule gets predicted vision information from the VisionModule and chooses destinations for all 5 robots. Finally ControlModule retrieves predicted vision from VisionModule and destinations from StrategyModule and makes robots go to those destinations.

5.3.1 VisionModule

This module was liable for taking the vision data, extracting velocity information from it, and predicting the location of the robots and the ball in the future frame.

Our total system latency, measuring from the period between command velocity and raw velocity, is approximately 133 ms (8 frames). When our robot move at the fastest speed, that is up to about 3.5 m/s, the distance between real robot position and the robot position from vision data will grow up about 47 cm. In order to correct this error we have to estimate the positions and orientations of the robots. The estimation architecture is shown in Figure 13.

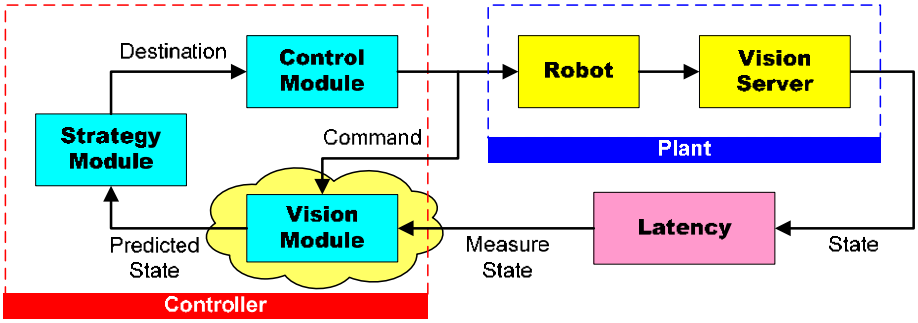

Fig. 13. The architecture of the estimator.

For the opponents and the ball, filtering and estimation were performed using Kalman filters. For teammate robots, the commanded robot velocities were used to gain more accurate estimation of their position.

5.3.2 StrategyModule

We have a hierarchical model in our StrategyModule design. The module is rebuilt from scratch by using strategy structure based on Cornell Big Red 2002.

The StrategyModule consists of multiple Plays. Whenever a Play is executed it calls the Roles for all Positions present. Then Roles run Skills for the related robots. All the plays are stored in the PlayBook in an array, while all the skills are stored in each SkillSet array.

The StrategyModule architecture is depicted in Figure 14. There are five layers, described in detail below.

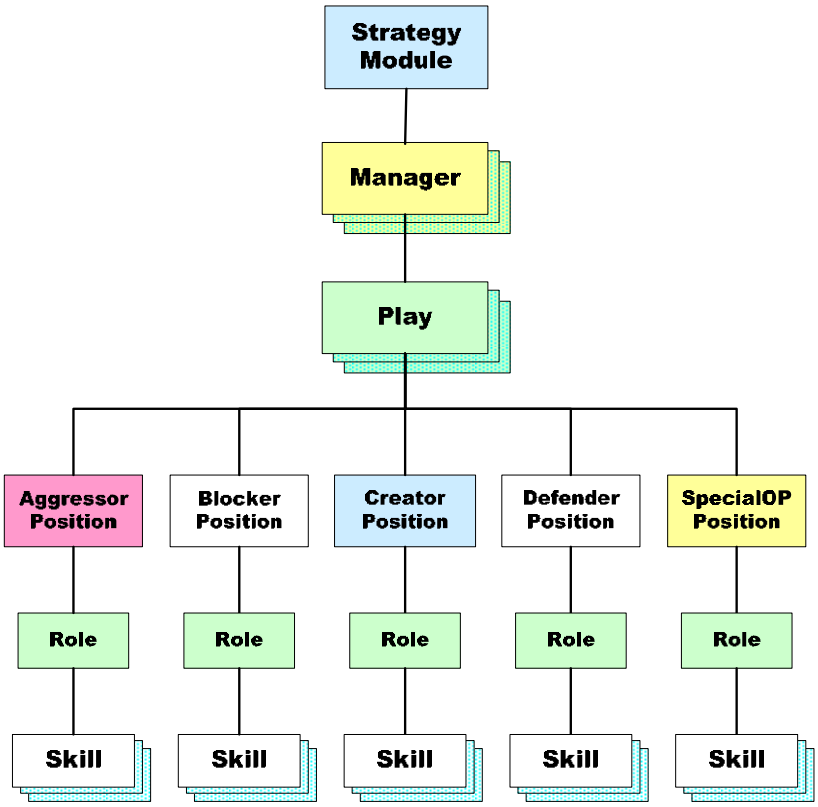

Fig. 14. The architecture of StrategyModule.

Play represents a particular global state of the AI and the general goal the positions are attempting to achieve at any given time. Examples are “OffensePlay”, “DefensePlay”, “FreekickUsPlay”. During any play, roles for positions that are present are executed. There are unique roles for each position for each play and plays do not call skill directly. The system will transition from one play to another when necessary by PlayTransition, but while in a particular state a particular play is being executed every frame.

After receiving data from referee box signal, Manager will select the group of play that suitable for that moment such as “GrazManager” or “SuzhouManager”. Each Play in Manager is configured by system parameter.

Skill is a basic action of robot, such as “ShootingSkill” or “GetBallSkill”. Each robot has a set of skills stored in a SkillSet object. Each skill is different and performs a different function. Skills allow state to be kept because skills are objects with private data, not just functions as used in the past. In addition, skills provide various methods for initialization, running, loading and reloading parameters, and much more.

Role is a combo set of skills that call by each position, such as “ForwordRole” or “GoalieRole”. By the way, both plays and positions can call skill directly but it will complicate if they have many states. Roles are object that inherit from skill, so they have same properties with skill.

There exist four robot positions on the field Blocker, Defender, Aggressor, and Creator. The fifth robot position is called SpecialOp that can take on one of three dutys: SpecialOpDefender, SpecialOpAggressor, or SpecialOpCreator.

The Blocker remains in the defense zone the majority of the time, only venturing slightly outside of it at times. The Blocker is the only position that will try to grab the ball when inside of the goalie box.

The Defender is a dedicated position to the defense. The defender always remains on our side of the field, almost entirely in the defense zone, and works with or supplements the actions of the blocker, stopping shots or closing holes whenever possible.

The Aggressor is the most active player on the field. See a robot who has the ball, he's undoubtedly the aggressor. See a robot go up to an opponent who has the ball, either to screen him from our goal or strip the ball away, that is the aggressor.

The Creator is our dedicated robot to creating opportunities. The creator spends the majority of his time far upfield, either in the kill zone, offensive zone, and sometimes as low but never lowers than the death zone.

The SpecialOpDefender acts as an auxiliary defender. When available, the SpecialOpDefender may screen auxiliary opponents who are coming down the field from getting near the ball. He may also help block passes or shots on goal. Usually he roams slightly in front of the Defender, or on the opposite side of the field, allowing him to move upfield and become a SpecialOpAggressor or SpecialOpCreator when the play changes.

The SpecialOpAggressor assists the aggressor. This means running screens to help the aggressor dribble up the field, setting up picks for quicker jukes by the aggressor, and also getting open for quick passes upfield when the aggressor gets bogged down.

The SpecialOpCreator helps the creator create opportunities by screening or various other blocking techniques. He also gets open for a pass under such scenarios.

5.3.3 ControlModule

ControlModule receives predicted vision from VisionModule and destinations from StrategyModule and makes robots go to those destinations. So, the essential component of ControlModule is a path planning algorithm.

Since the World RoboCup 2008 at Suzhou, we have made use of the “Real-Time Randomized Path Planning for Robot Navigation” [3] for default path planning algorithm. The path planning developed representation on Rapidly-Exploring Random Trees (RRTs) as shown in Figure 15.

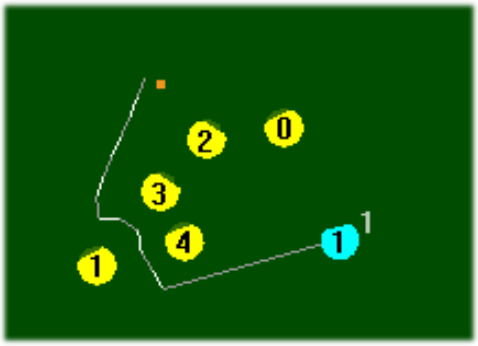

Fig. 15. The robot path generated from RRT path planner.

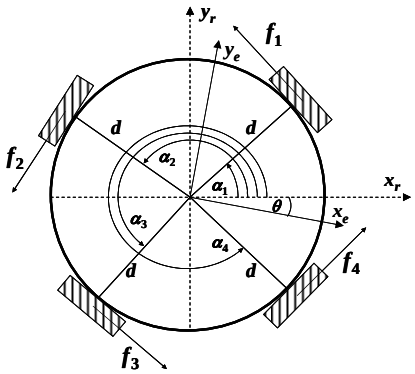

Fig. 16. Wheel configuration of the robot.

5.3.4 Modified Robot Kinematics

Normally, when the software sends the velocity command to the robot, it doesn't perform any velocity feedback control and it assumes that the robot's motion controller has already taken care of this. But due to the loss from friction, wheel slippage and other real world problems, the robot cannot move as fast as commanded. The regular robot kinematics describes an ideal situation where there's no system disturbance. In order to control the robot more accurately, the robot kinematics is modified with the some disturbance parameters. The friction force and traction torque vector are defined.

The normal kinematics can be written as:

$$\zeta_{earth} = (\psi^{\dagger}) \cdot \zeta_{command} \tag{1}$$

where,

$$\zeta_{earth} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{\theta} \end{bmatrix}^T$$

$$\zeta_{command} = \begin{bmatrix} \dot{\phi}_1 & \dot{\phi}_2 & \dot{\phi}_3 & \dot{\phi}_4 \end{bmatrix}^T$$

$$\psi = \begin{bmatrix} \cos\theta \cdot \sin\alpha_1 + \cos\alpha_1 \cdot \sin\theta & \sin\theta \cdot \sin\alpha_1 - \cos\alpha_1 \cdot \cos\theta & -d \\ \cos\theta \cdot \sin\alpha_2 + \cos\alpha_2 \cdot \sin\theta & \sin\theta \cdot \sin\alpha_2 - \cos\alpha_2 \cdot \cos\theta & -d \\ \cos\theta \cdot \sin\alpha_3 + \cos\alpha_3 \cdot \sin\theta & \sin\theta \cdot \sin\alpha_3 - \cos\alpha_3 \cdot \cos\theta & -d \\ \cos\theta \cdot \sin\alpha_4 + \cos\alpha_4 \cdot \sin\theta & \sin\theta \cdot \sin\alpha_4 - \cos\alpha_4 \cdot \cos\theta & -d \end{bmatrix}$$

$\psi^{\dagger}$ is the pseudo inverse of the kinematic equation

Equation (1) is lack of information about surface frictions. Therefore, if robot trajectories are generated from equation (1), those trajectories cannot guarantee the real robot velocity and position. The modified kinematic model is introduced in order to modify the robot kinematic with friction parameters. Let the modified kinematics of the mobile robot be described by

$$\zeta_{earth} = (\psi^{\dagger} + \varepsilon) \cdot \zeta_{command} + \Delta \tag{2}$$

where,

ε is the Viscous friction matrix (velocity friction)

Δ is the Coulomb friction vector (surface friction)

From the experiment, friction can be separated into two friction, coulomb friction and viscous friction. Coulomb friction vector (Δ) is a constant vector according to time while viscous friction matrix (ε) is a function of the specific angular velocity of the robot (independent of time). The coulomb friction vector is easily found by experiment since this friction is constant for specific surface but the viscous friction cannot be found by simple experiment. Although the viscous friction is linearly dependent on an angular velocity of the robot chassis but this friction is a non-linear function when it's transformed to time domain. The viscous friction can be approximated to the linear third-order polynomial. In order to find the viscous friction matrix, the captured robot velocity in the earth frame is used. Let consider

$$\zeta_{captured} = \zeta_{earth} + \eta \tag{3}$$

where,

$\zeta_{captured}$ is a captured robot velocity by bird eye view camera

η is a measurement noise

And

$$\zeta_{earth\_ideal} = \psi^{\dagger} \zeta_{command} \tag{3}$$

where,

ζ_{earth_ideal} is a ideal velocity of a robot when robot motors are motorized at command speed $\zeta_{command}$

Define the error function at particular time, by this definition ε is constant for the specific angular velocity at a time.

$$e = \zeta_{captured} - \zeta_{earth\_idea} \tag{4}$$

Evaluated equation (4) as following:

$$\begin{aligned} e &= \zeta_{captured} - \zeta_{earth\_idea} \\ &= \zeta_{earth} + \eta - \psi^{\dagger}\zeta_{command} \\ &= (\psi^{\dagger} + \varepsilon)\zeta_{command} + \Delta + \eta - \psi^{\dagger}\zeta_{command} \\ &= \psi^{\dagger}\zeta_{command} + \varepsilon \cdot \zeta_{command} + \Delta + \eta - \psi^{\dagger}\zeta_{command} \\ e &= \varepsilon \cdot \zeta_{command} + \Delta + \eta \end{aligned} \tag{5}$$

Idea is find the ε matrix by using Least Square Error (LSE) approximation. But if we develop equation (5) as a normal procedure, the inverse of term $(\zeta_{command} \cdot \zeta_{command}^{T})$ is not exist because $\det(\zeta_{command} \cdot \zeta_{command}^{T})$ is always zero. In order to solve this problem, the Kronecker product is used. The Kronecker product is a operator transform a regular matrix multiply to a block matrix as:

$$\bar{A} \otimes \bar{B} = block[a_{ij}\bar{B}] \tag{6}$$

If $\bar{A}$ is $m \times n$ and $\bar{B}$ is $s \times t$, then $\bar{A} \otimes \bar{B}$ is an $ms \times nt$ matrix. Kronecker production is used to rearrange an order of $\varepsilon \cdot \zeta_{command}$, and then the LSE is possible. After apply the index modification, the final form of the transformation can be defined as:

Let $A \in R^{m \times l}$ and $X \in R^{l \times m}$ then

$$AX = B \Leftrightarrow (I_m \otimes X^T) \cdot \vec{a} = \vec{b} \tag{7}$$

where,

$\vec{a}, \vec{b}$ are the vector version of the matrix A, B respectively

Equation (5) can be rearranged by Kronecker Product as the following process:

1. Let $y = e - (\Delta + \eta)$, $x = \zeta_{command}$, $A = \varepsilon$
   then $e - (\Delta + \eta) = \varepsilon \cdot \zeta_{command} \rightarrow y = Ax$
2. Solve $A_{mxn}x_{nx1} = y_{mx1} \Leftrightarrow (I_m \otimes x^T)\bar{A}_{mnx1} = y$

3. Viscous friction matrix (now in vector form) =

$$\bar{A}_{mnx1} = ((I_m \otimes x^T)^T \cdot (I_m \otimes x^T))^{-1} \cdot (I_m \otimes x^T)^T \cdot y$$

After viscous friction matrix is found, the experiment can be retest again in order to lower the measurement error.

Figure 17 and 18 show the captured data from vision system and the result after the viscous friction of the matrix is approximated respectively. Since there are measurement noises therefore these viscous matrix parameters are needed to average and their means are the representatives of their group. We use this average viscous friction matrix as our viscous matrix at particular time.

Although, the viscous friction matrix is found but it represents only at specific angular velocity of the robot at a particular time therefore the parameters of the matrix in other angular velocity are needed to be found by repeating the experiment, and then fitted into polynomial curve as mentioned before.

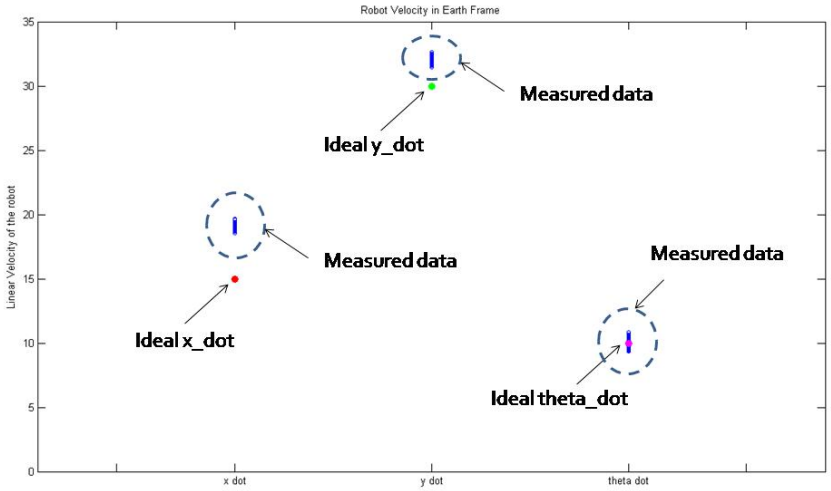

Fig. 17. Linear velocity of the robot in Earth frame $[\dot{x} \quad \dot{y} \quad \dot{\theta}]^T$

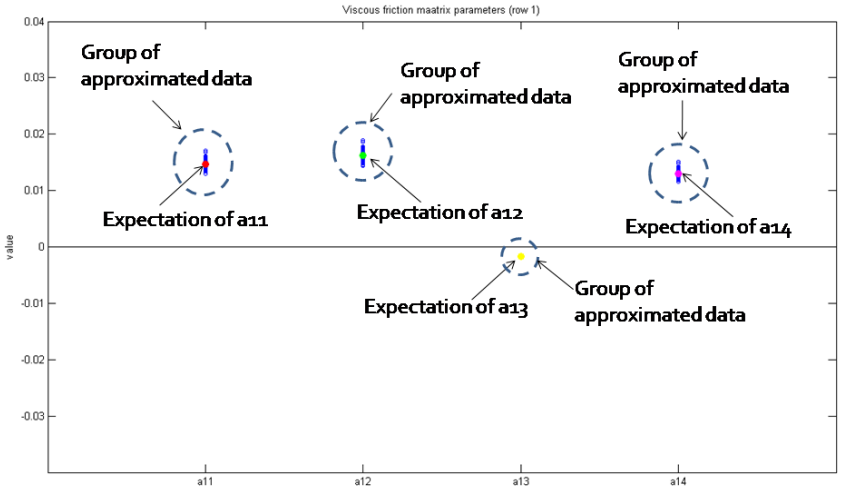

Fig. 18. Approximated viscous friction matrix parameters.

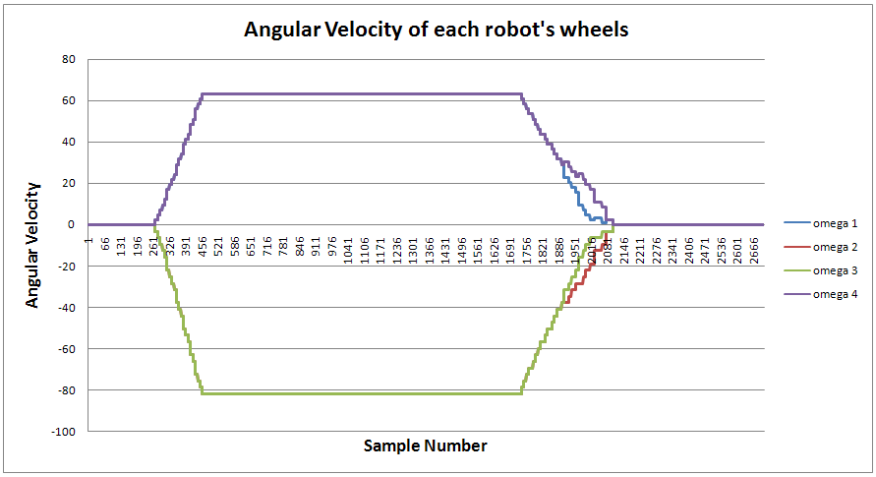

Fig. 19. Useable region of the velocity data.

By using modified kinematics to generate the control command, the robot can move more accurately. The comparison of the experimental result is shown in Figure 20.

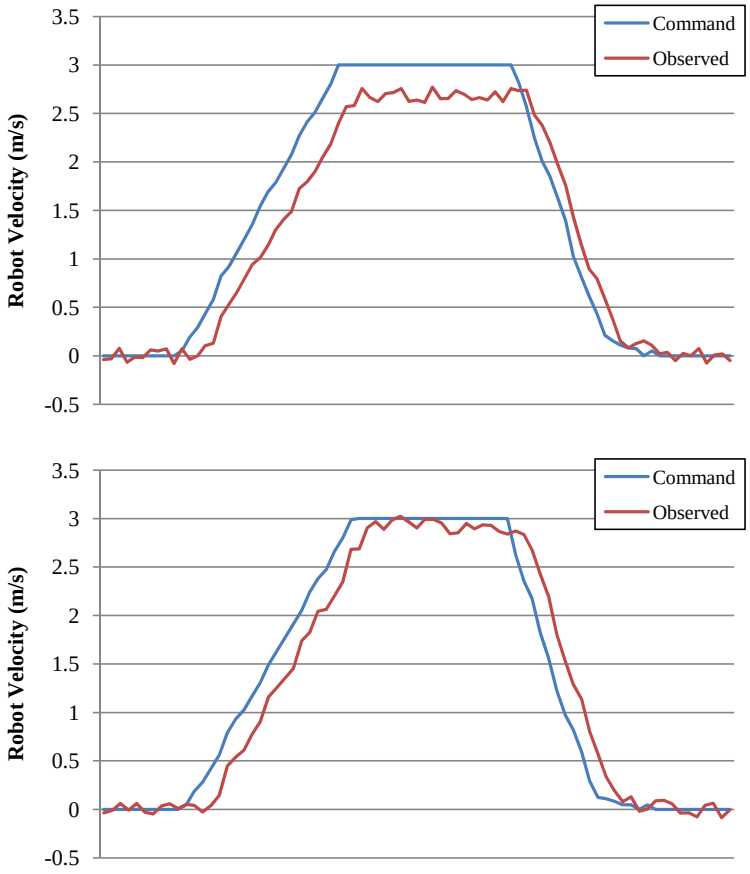

Fig. 20. The robot observed velocity profile using normal kinematics (top) and modified kinematics (bottom).

5.4 RadioServer

Commands which are generated from two individual StrategyModules are encapsulated into a single packet in the RadioServerModules. After commands are packed, the module will send this packet to the wireless board via a USB port. We use full duplex communication for our system in order to get some information form robots in real-time. Therefore, this RadioServerModule is also used to receive the send-back data and return it to each StrategyModule.

5.5 SimulatorServer

Our SimulatorServer is developed in order to simulate robot hardware behavior. The SimulatorServer receives a sequence of packets which is identical to packets that are sent to real robots then calculates some simple physics and returns the coordinate of objects in the field to the software as same as the VisionServer does. Our SimulatorServer is entirely independent from AI System which is capable of simulating all the field objects and latency of the system.

The field objects are simulated by physics engine library called Open Dynamics Engine (ODE) [12]. The SimulatorServer provides connection socket for some two AI systems which means that the simulator is capable to simulate a real competition situation.

5.6 Kicker Automatic Calibration

The automatic calibration system is added to the software in order to reduce manpower and time during the team setup process. The calibration software performs sets of experiment in order to obtain the relationship between input and output parameters. Similarly to the motion controller calibration method described in section 2.1, the kicker calibration software is used to estimate the relationship between both the chip-kicking distance or ball speed and the magnitude of the kick command sent to the robot. In the calibration procedure, the speed of the ball is easily obtained from the vision module while the chip-kicking distance requires trajectory estimation method based on the model given in [4]. This method uses the camera parameters from the SSL-Vision and several ball positions in consecutive frames to approximate the actual trajectory of the ball in the air. Given the parabolic trajectory of the ball, the falling point is calculated and the chip-kicking distance is obtained. The snapshots of the calibration process of the chip-kicker are shown in Figure 21 and 22. The relationship is then modeled using second order polynomial and can be estimated from the experimental results using least squares polynomial fitting. The example of the kicker calibration result is shown in Figure 23.

Fig. 21. Snapshots of the chip-kicker calibration process.

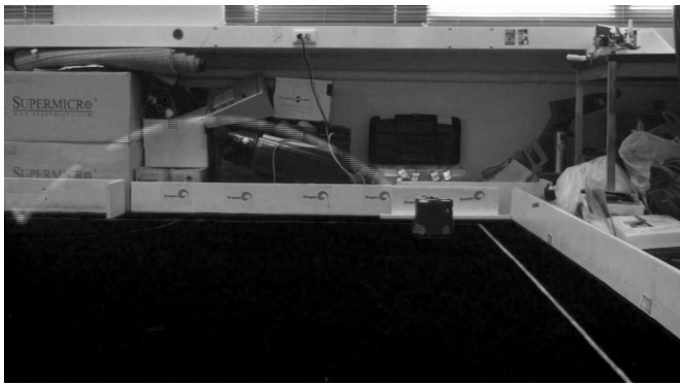

Fig. 22. Parabolic trajectory of the chip-kicked ball.

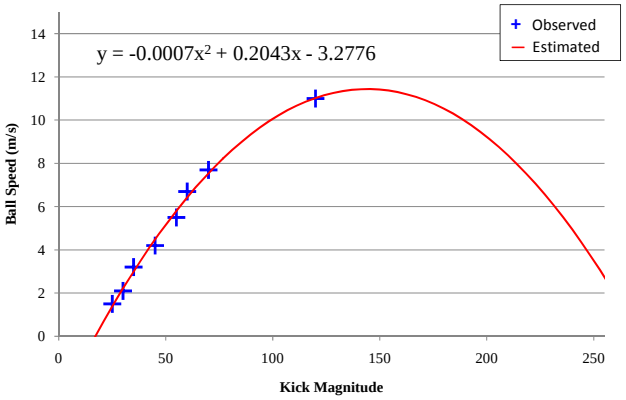

Fig. 23. Relationship between the ball speed and the magnitude of the kick command obtained using second order polynomial least squares fitting.

6 Conclusion

Table 1. Competition results for Skuba SSL RoboCup team.

Competition	Result
RoboCup Thailand Championship 2005	3rd Place
RoboCup Thailand Championship 2006	Quarter Final
RoboCup 2006	Round Robin
RoboCup Thailand Championship 2007	3rd Place
RoboCup Thailand Championship 2008	2nd Place
RoboCup 2008	3rd Place
RoboCup 2009	1st Place
RoboCup China Open 2009	1st Place

Our system has been continuously improving since the beginning. Last year, we introduced some improvements about the low level motion controller and the robot hardware. The new calibration software for this year was proven to be very useful. In RoboCup China Open 2009, the automatic calibration software greatly reduced the amount of team setup time which allowed us to focus more on the strategic planning. The software which runs the robot team was built in 2006 and improved each year. It has given us very successful competition results for the last several years, the results are summarized in table 1. We hope that our robot team will perform better in this year and we are looking forward to sharing experiences with other great teams around the world.

References

1. Falin, J.: Minimizing Ringing at the Switch Node of a Boost Converter. Application report, Texas Instruments Inc. (2006)
2. Fosler, R.: Generating High Voltage Using the PIC16C781/782. Application note, Microchip Technology Inc. (2005)
3. Bruce, J., Veloso, M.: Real-time randomized path planning for robot navigation. In: Proceedings of the IEEE Conference on Intelligent Robots and Systems. (2002)
4. Kim, T., Seo, Y., Hong, K.S.: Physics-based 3d position analysis of a soccer ball from monocular image sequences. Sixth International Conference on Computer Vision (1998) 721–726