ODENS 2010 Extended Team Description

Makoto Nakajima, Takaumi Kimura, Yasuhiro Masutani

Osaka Electro-Communication University
1130-70, Kiyotaki, Shijonawate, Osaka 575-0063, Japan

1 Introduction

Team ODENS consists of members of Masutani Laboratory in Department of Computer Science, Faculty of Information Science and Arts, Osaka Electro-Communication University, Japan. ODENS has participated in RoboCup competition since RoboCup Japan Open 2007. The results in the Japanese competitions were the 4th place in 2007, the 2nd place in 2008, and the 3rd place in 2009. Moreover, we won the 4th place in RoboCup 2009 Graz, which was the first experience of a world competition for ODENS[1].

In the Department of Computer Science, Exercise in Robot Programming is organized for the second grade students, which uses the actual robot system for RoboCup competition. Students learn basic programming of deciding action of soccer robots in the exercise. In the room for the exercise, full-size SSL field and two ceiling cameras are readied. Since ODENS develops robots and programs there, it can always do exercises and experiments on the assumption of regular game.

In this department, students belong to laboratories from the second semester of the second grade. Students who are interested in soccer robot through "Exercise in Robot Programming" wish to enter Masutani Laboratory. In Masutani Laboratory, projects for RoboCup are themes for pre-seminar before regular graduation thesis. Moreover some students study RoboCup as also theme of graduation thesis.

Since the department is in the field of computer science and technology, the second grade students focus on development of software for deciding action. The hardware of robot is designed and manufactured by a company. The software of controlling omnidirectional mechanism is improved by graduate students and higher grade students.

In the following sections, the hardware and software of robot is introduced first. After that the computer system outside of field, especially a method of deciding action, is described.

2 Overview of the system

The system of ODENS is a distributed and autonomous system, which consists of one server and some clients as shown in Fig.1. One client program corresponds to one robot. All client is independent of each other. The server receives coordinate

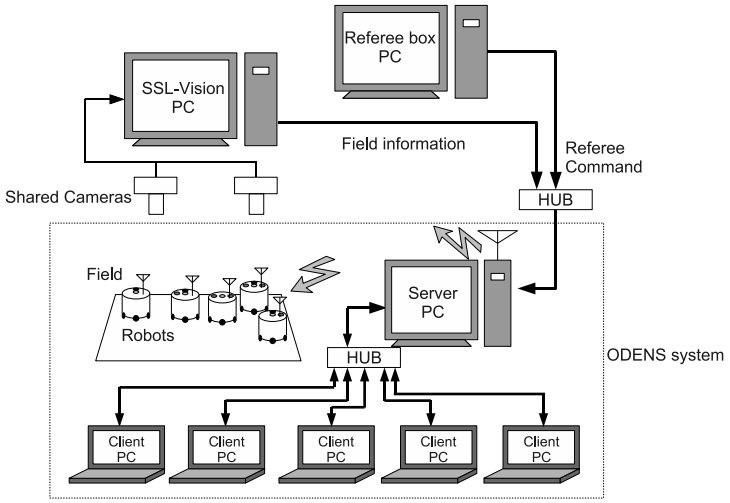


Fig. 1. Server and client system of Team ODENS

information of objects in the field provided by SSL-Vision. They are transformed in appropriate format and sent to the clients. Signals from the referee box are also sent to them. Each client program decides the next action for the corresponding robot based on the information received from the server and sends a command back to the server. The server collects the commands from all clients and sends it to the robots in the field by wireless. The above cycle is executed 60 times in one second.

3 Robots

The mechanical and electrical hardwares of robots is designed and manufactured by SMATS Corp., which is very similar to RoboDragons' robot[2]. However, the onboard software for control is original.

3.1 Hardware

An appearance of the robot is shown in Fig.2. The robot can be packed in the cylinder with dimensions of 146 mm height and 175 mm diameter. The main dimensions of the robot with a cover are shown in Fig.3. The main specifications are shown in Table 1. The system configuration of robot is shown in Fig.4.

3.2 Software for Robot Control

The program of CPU on the robot is developed with GNU C compiler. Three tasks are concurrently processed on the CPU. The task for feedback control is

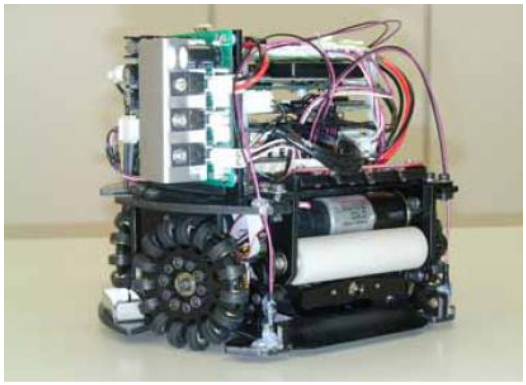
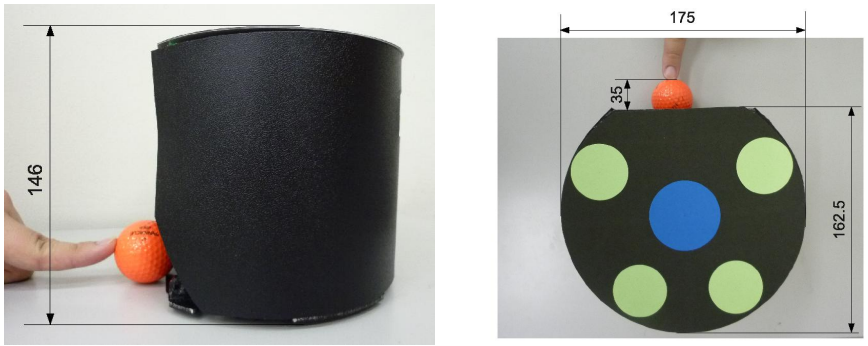


Fig. 2. Appearance of the robot



(a) Side view (b) Top view

Fig. 3. The main dimensions of the robot with a cover

Table 1. Specifications of the robot

Dimension	length 162.5[mm], width 175[mm], height 146[mm]
Mass	2.2[kg]
Wheel	number 4, radius 30.5[mm], sub wheels 15
Motor	maxon RE-max24 222050, 11[W], reduction ratio 7.916
Kicking device	3 solenoids, 240[V]
Dribbling device	radius 10[mm], length 72[mm]
Ball sensor	LED and Photo-transistor, number 4
Gyro sensor	ADXRS610
Wireless modem	Futaba FRH-SD07T, 2.4[GHz]]
CPU	Renesas SH7045F(SH-2), clock 28[MHz]
Memory	SRAM 8[Mbit], Flash ROM 4[Mbit]
Battery	Li-Polymer 14.8[V]×1, 7.4[V]×1

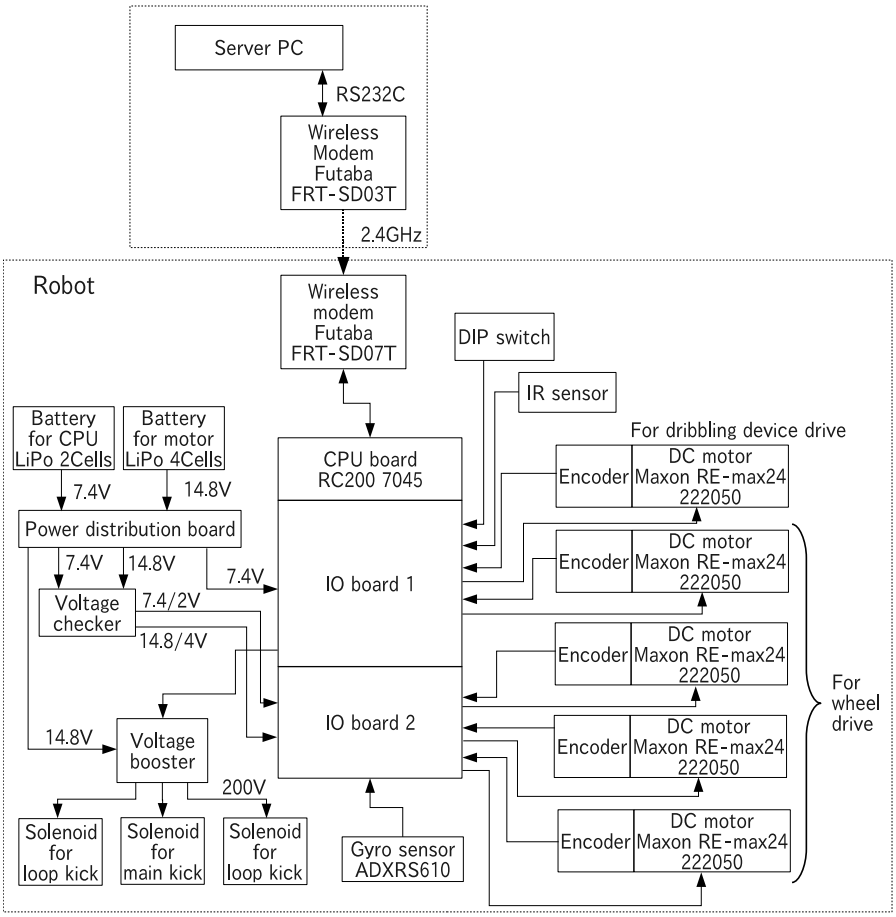


Fig. 4. Robot system configuration

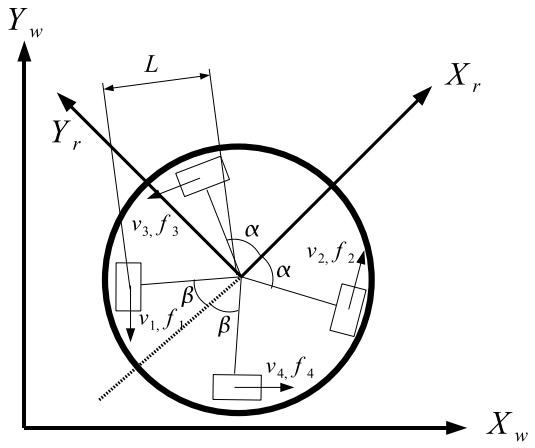


Fig. 5. Model of omnidirectional mobile robot with 4 wheels

executed every 2[ms]. The task for communication receives a command from the outside via wireless modem every 16.7[ms]. The command consists of magnitude and direction of linear velocity, angular velocity, and on/off of dribbling device and kicking device.

Modeling As shown in Fig.5, a coordinates system is attached on the robot. Then numbers are assigned for wheels. Let α be angle of axes of front wheels from the front direction $(+X_r)$, β be angle of axes of rear wheels from the rear direction $(-X_r)$, and, L be distance between the center and the wheel. We define the vector $\boldsymbol{v} = [v_1, v_2, v_3, v_4]^T$ as set of velocities of wheels at contact point and $\boldsymbol{V} = [V_x, V_y, \Omega]^T$ as set of linear and angular velocities of the body in the robot coordinates system. Relation between two velocity vectors is obtained based on geometry as follows,

$$\boldsymbol{v} = A\boldsymbol{V} \tag{1}$$

$$A = \begin{pmatrix} -\sin\beta & -\cos\beta & L \\ \sin\alpha & \cos\alpha & L \\ -\sin\alpha & \cos\alpha & L \\ \sin\beta & \cos\beta & L \end{pmatrix} \tag{2}$$

Furthermore, we define the vector $\boldsymbol{f} = [f_1, f_2, f_3, f_4]^T$ as set of forces acting on the wheels at contact point and the vector $\boldsymbol{F} = [F_x, F_y, N]^T$ as set of resultant forces and moment acting on the body. Relation between two force vectors is obtained from the principle of virtual work as follows,

$$\boldsymbol{F} = A^T\boldsymbol{f} \tag{3}$$

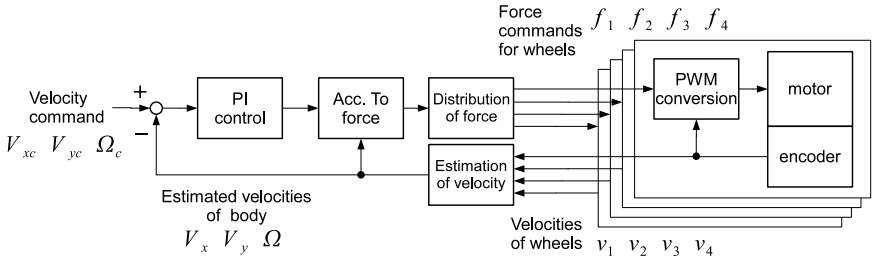


Fig. 6. Diagram of control law

Control law ODENS uses a control law based on dynamics model. Fig.6 shows its overview. For velocity command V_{xc}, V_{yc}, Ω_c given from the outside, method of computing the duty ratio of PWM motor drive is described in the following.

Letting ω_j be angular velocity of the j-th wheel, N be reduction ratio, and, r be radius of wheel, the velocity of wheel at contact point is represented as follows,

$$v_j = \frac{r}{N}\omega_j \tag{4}$$

The velocity of the body $\boldsymbol{V}$ is estimated from wheel velocities $\boldsymbol{v}$ based on an error minimum solution of Eq.(1).

$$\boldsymbol{V} = (A^T A)^{-1} A^T \boldsymbol{v} \tag{5}$$

The angular velocity Ω in $\boldsymbol{V}$ is replaced with measured value of gyro sensor to avoid error of estimation from wheel velocities.

Accelerations to be provided for the body are computed based on PI feedback.

$$a_x = k_{Px}(V_{xc} - V_x) + k_{Ix}\Delta t \sum (V_{xc} - V_x) \tag{6}$$

$$a_y = k_{Py}(V_{yc} - V_y) + k_{Iy}\Delta t \sum (V_{yc} - V_y) \tag{7}$$

$$\alpha = k_{P\theta}(\Omega_c - \Omega) + k_{I\theta}\Delta t \sum (\Omega_c - \Omega) \tag{8}$$

where k_{P*} is proportional gain and k_{I*} is integral gain. Forces acting on the body are computed based on equation of motion.

$$F_x = M(a_x - V_y\Omega) + \gamma_x \tag{9}$$

$$F_y = M(a_y + V_x\Omega) + \gamma_y \tag{10}$$

$$N = I\alpha + \gamma_\theta \tag{11}$$

where M and I are mass and inertia moment of the body respectively, γ_* is compensation of friction.

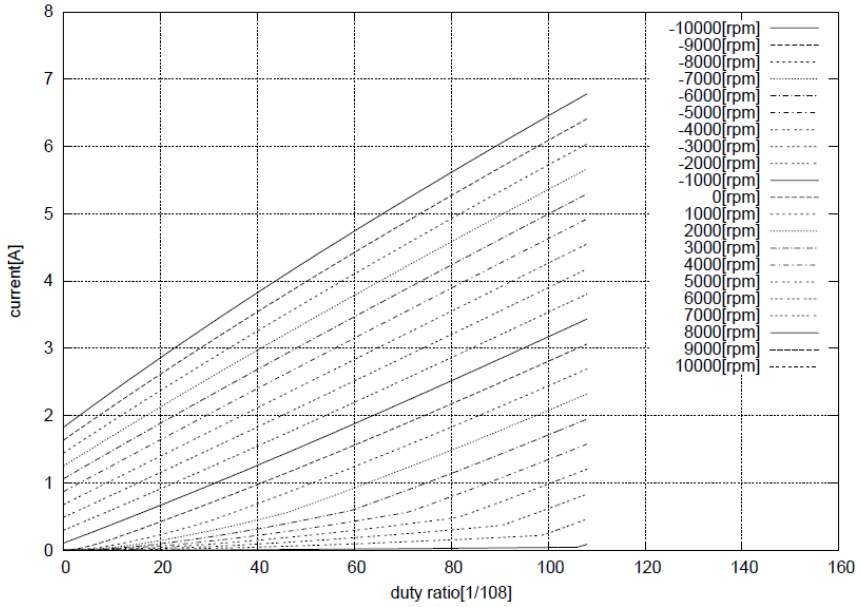


Fig. 7. Relation between the duty ratio and current

The wheel forces realizing this resultant force are computed as norm-minimizing solution of Eq.(3).

$$\boldsymbol{f} = A(A^T A)^{-1}\boldsymbol{F} \tag{12}$$

The current of the j-th motor i_j is given as follows,

$$i_j = \frac{r}{NK_t} f_j \tag{13}$$

where K_t is torque constant.

For PWM-driven motor, the current i can be represented as a function of the duty ratio p and the angular velocity ω, $i = g(p, \omega)$, based on model of electrical circuit. The program has a numerical table of inverse of this function. For given current and angular velocity, duty ratio is decided by the table looking up.

$$p_j = g^{-1}(i_j, \omega_j) \tag{14}$$

The relations the duty ratio and current is shown in Fig.7. The 3D shape of the $g^{-1}(i, \omega)$ is shown in Fig.8.

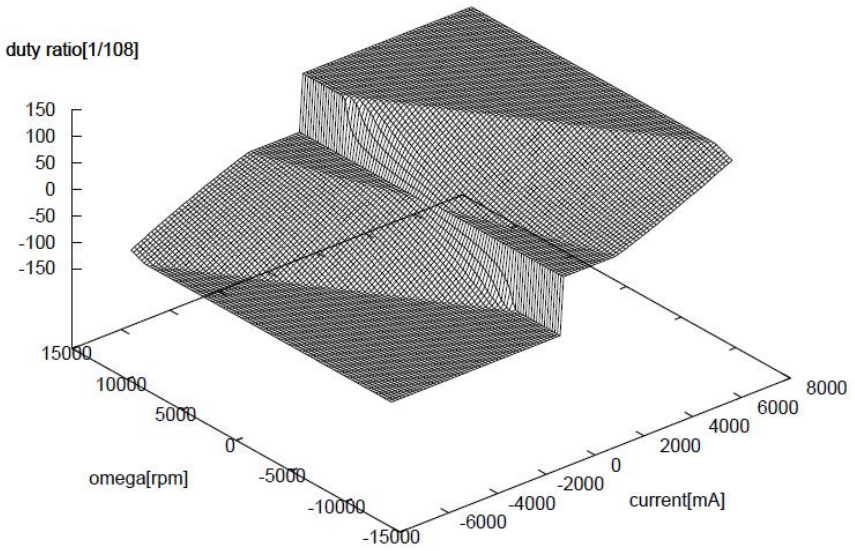


Fig. 8. $p = g^{-1}(i, \omega)$

4 Server

4.1 Hardware

PC The server PC is equipped with Intel Core2 Duo P8700 2.53[GHz] and 2[GB] RAM. It is connected to the SSL-Vision PCs, referee box PCs, and some client PCs through LAN. It also uses a USB-serial converter to connect the wireless modem.

Wireless communication Wireless modem Futaba Corp. FRH-SD3T is used to communicate with the robots in the field, whose communication rate is 38.4[kbps] and whose frequency is 2.4[GHz].

4.2 Software

OS for server PC is Microsoft Windows XP Professional SP3. Visual Studio 2008 C++ is used to develop the program. The server program has GUI and consists of multiple threads.

The construction of the threads and connecting relations with the outside of the server is shown in Fig.9. The oval represents outside of the server, and the rectangle represents the thread. The "signal" means synchronization between

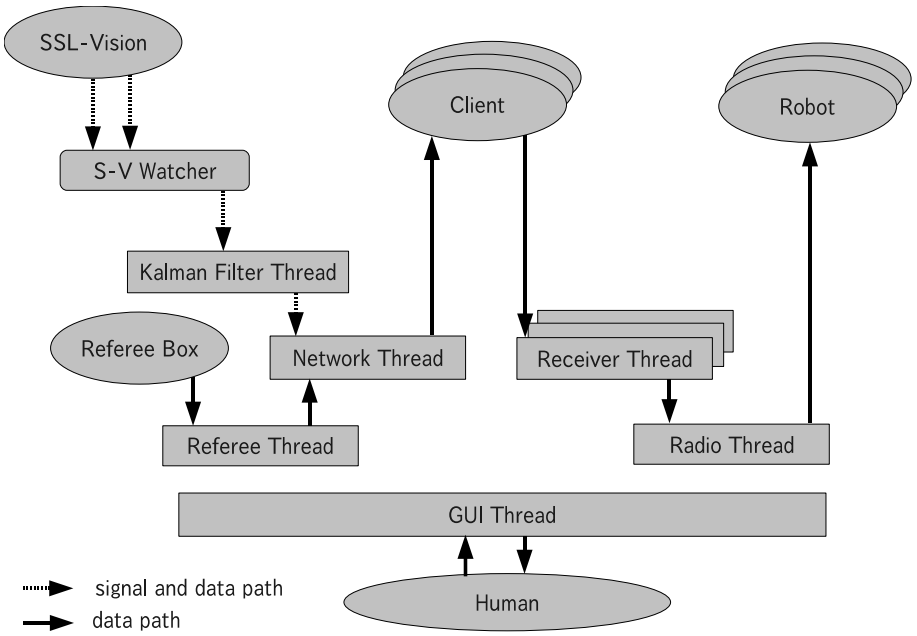


Fig. 9. The structure of multi-threads in the server program

the threads, and te "data" means the exchanges of data between the threads. "S-V watcher" is a program running on the server PC, but it is not contained in the server program. Functions of these threads are described as follows.

S-V watcher SSL-Vision watcher is a thread to merge and manage the field information divided into two that is given from SSL-Vision[3]. It gives the merged field information to the "Kalman filter" thread.

S-V watcher shared the memory area with the Kalman filter thread and exchanges information through this memory area. S-V watcher transmits the message to the Kalman filter thread to tell that new information is written on the memory.

Threads

Kalman filter thread The positions and velocities of objects are estimated by Kalman filter, whose observed values are the position of the objects given from S-V Watcher and whose model is a constant velocity motion. Furthermore, this function tackles exceptional situations by evaluating the difference between the observed value and the estimated value. It gives the estimated information such as robot number and coordinate to the network thread.

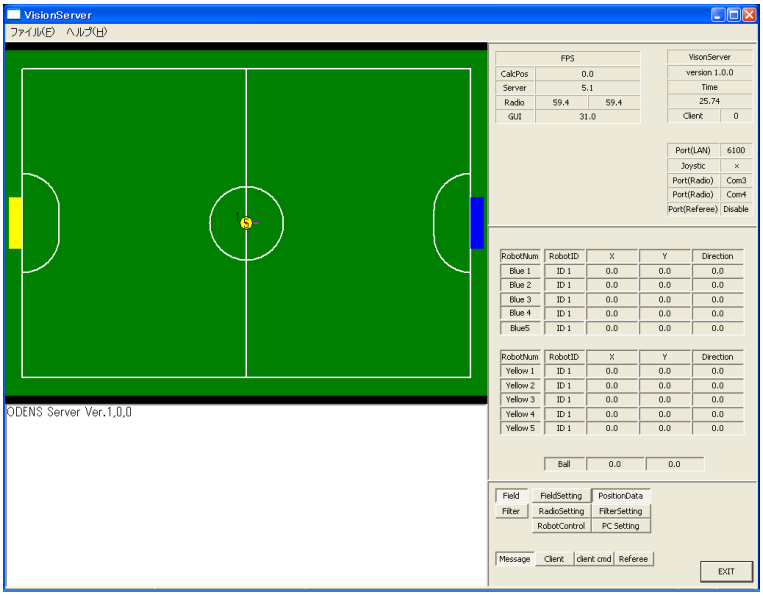


Fig. 10. GUI of the server program

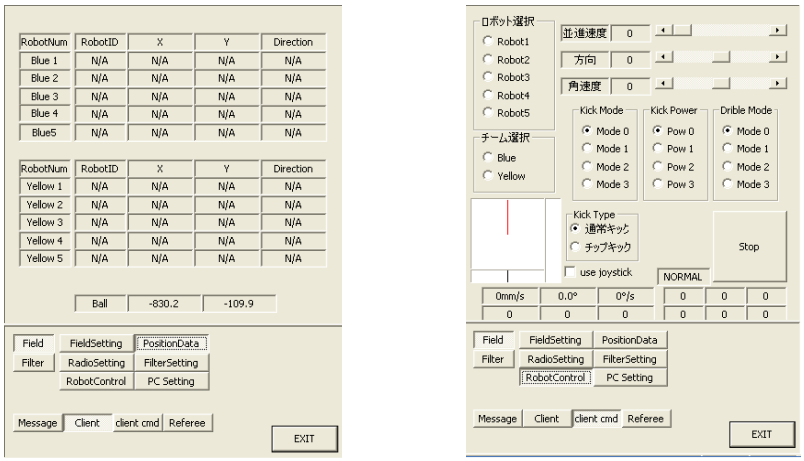
Network thread The network thread transmits the information received from the Kalman filter thread and the signals from the referee box to the client(s). Moreover this thread to accepts the connection request from the client(s).

Referee thread The referee thread transforms the signals received from the referee box to the network thread.

Receiver thread The receiver thread communicates with the client(s). However, this thread does not transmit the message to the client(s). It receives the command to the robot from the client(s). Since this thread corresponds to a client, the number of the threads increases and decreases depending on the number of the connected clients. It gives data which received form the client(s) to the radio thread.

Radio thread The radio thread transmits the command which received from the receive thread to the robots.

GUI thread The GUI thread accepts the input from the user and performs indication to the display. The processing priority of this thread is lower than other threads not to obstruct the more important threads. An example of the GUI screen is shown in Fig.10.



(a) Position data (b) Robot control

Fig. 11. The right side of GUI display

GUI The right side of GUI display are shown in Fig.11. Fig.11(a) shows the coordinates and front direction of the robots and the coordinate of the ball in the field. Fig.11(b) shows the robot control panel, which allows us to operate the robots manually. At first, one chooses the number and the color of the robot to operate in the left radio button. One can choose multiple buttons to can operate the multiple robots at the same time.

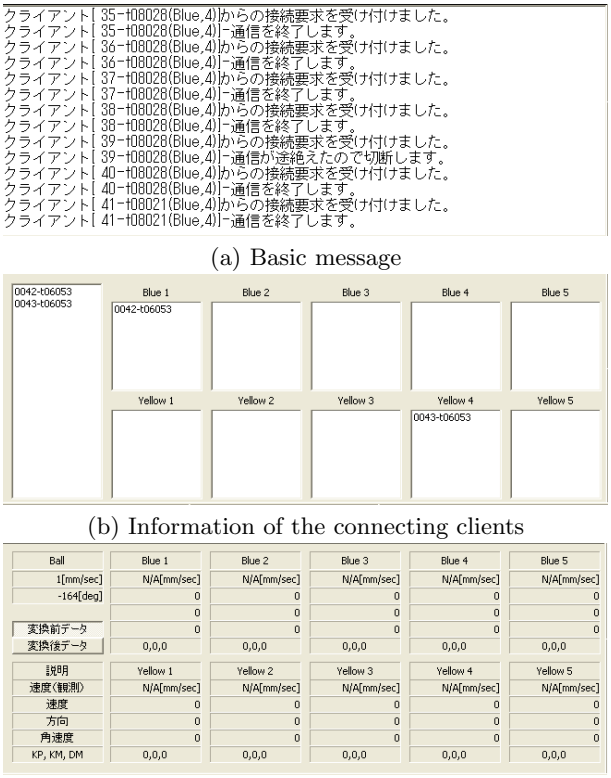
The lower left side of the display is shown in Fig.12. Fig.12(a) shows which the server accepts the connection and disconnection request from the clients. Fig.12(b) shows the name of connected client computer. Fig.12(c) shows the command to the robots received from the clients.

Communication protocol The system of ODENS adopts the server-client system. The client was developed aiming at running in any kind of environment if based on the protocol. The server and the client is connected by LAN and transmits and receives data by UDP.

In addition, loss and the delay of the message do not consider because the channels assumes only an extremely small network. Therefore, the client sends the necessary order one-sidedly without replying the reception completion message.

The information transmitted to the client(s) by the server is field information such as the coordinate of the robots and ball. The information transmitted to the server by the client is the command to the robot: velocity, movement direction, angular velocity, on/off of the kick device and the dribble device.

The protocol is represented in S-expression similar to that of the RoboCup Simulation League.



(a) Basic message

(b) Information of the connecting clients

(c) Commands to the robots

Fig. 12. Lower left side of the display

From the client to the server

- `(init (version` *VerNum*`)(number` *RoboNum*`)(`*Color*`)(global2)(Myname` *UserName*`))`
  VerNum::=the protocol version
  RoboNum::=(1–5)
  Color::=(`blue`|`yellow`)
  UserName::=the name of client comupter
- `(bye)`
- `(com (`*Vel Dir* ω *Options*`))`
  Vel::=the command of velocity
  Dir::=the command of movement direction
  ω::=the command of angular velocity
  Options::=the command of kick mode, kick power and dribbling mode
- `(say (`*Message*`) )`
  Message::=(`(pass ...`|`(role ...`)

From the server to the clients

- `(init l` *RoboNum* `global2 )`
  RoboNum::=(1–5)
- `(see` *Time* `(RS 1) ((b)` x_b y_b`) ((p` *Color RoboNum* `)`x y *dir*`) )`
  Time::=the running time of the server program
  x_b::=the x coordinate of the ball
  y_b::=the y coordinate of the ball
  Color::=(`b`|`y`)
  RoboNum::=(1–5)
  x::=the x coordinate of the robot
  y::=the y coordinate of the robot
  dir::=the front direction of the robot
- `(hear` *Time IDNum Message*
  Time::=the running time of the server program
  IDNum::=(1–5|99)
  Message::=(`(refereebox ...)`|`(role ...)`|`(pass ...)`
- `(refereebox` *Num*`)`
  Num::=Transformed char into int

From a client to the other clients

- `(pass 3` *RoboNum Time* `(` x_w `,` y_w `) (` x_t `,` y_t `)` *OwnNum*
  RoboNum::=(1–5)
  Time::=Wait time(default 3.0 sec)
  x_w::=the x coordinate to keep waiting
  y_w::=the y coordinate to keep waiting
  x_t::=the x coordinate that lets one kick the ball
  y_t::=the y coordinate that lets one kick the ball
  OwnNum::=(1–5)
- `(role` *OwnNum RoleID*`)`
  OwnNum::=(1–5)
  RoleID::=(0x$AABCDD$)

Management of clients The clients can connect to the server and specify the robot number. After that it can disconnect and reconnect at any time. The information of clients connecting is displayed in the GUI window. Since the server can manage clients of two teams at same time, two team can play a game by using one server if all clients follow the protocol.

Wireless communication The server can deal with two modems. The commands to five robots are packed in one packet and transmitted in broadcast. The packet is 32[byte], which consists of a header of 2[byte] and five commands of 6[byte]. The command consists of ECC (Error Check and Correct) data of 2[byte], magnitude of linear velocity of 1[byte], its direction of 1[byte], angular velocity of 1[byte], and switches of dribbling device and kicking device of 1[byte].

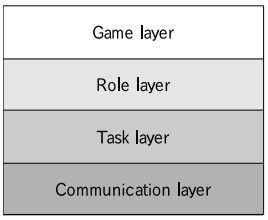


Fig. 13. Hierarchical structure of the client software

Referee box Commands received from the referee box is sent to all clients in the prescribed format.

Communication among the robots Communication among the robots is represented as communication among the clients through the server program in the prescribed format.

5 Simulator

A simulator is indispensable for development of the program of deciding action. The ODENS simulator uses ODE[4] for physical computation. Since it supports the same protocol as the server, the client program can connect to the simulator without changing its program. Moreover, it can deal with two or more clients and the referee box.

6 Client

PC executing the client program does not need special specifications and performance, whose OS is not specified(We use both Windows and Linux). Only function of network communication is needed to connect to the server.

One client program corresponds to the one robot. Therefore, all clients program are executed independently of each other.

6.1 Structure of action decision

The structure of client program is separated into four layers as shown in Fig.13.

Game layer This layer is processed based on a state transition, whose state is switched mainly by command received from the referee box. There are five states, “Out of play”, “Pre set play”, “Set play”, “In play”, and “Halt”. The state transition diagram is shown in Fig.14.

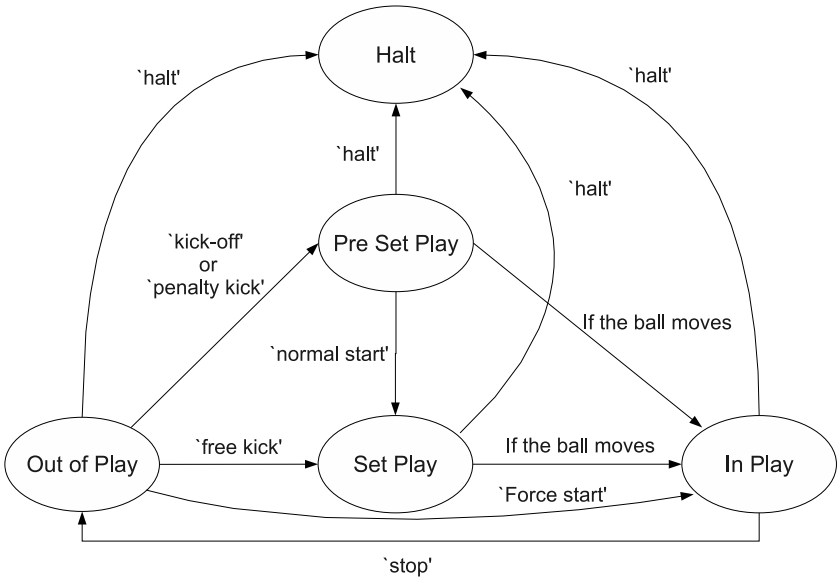


Fig. 14. The state transition in the game layer

Role layer This layer decides role of each robot at each state. The role is assigned based on a role table when the game state is "In play". Otherwise, The role is assigned based on only rank of distances between robots and ball in the team.

Task layer This layer execute concrete actions of the robot. For example, "Turn to the ball", "Move to specified coordinates while avoiding obstacles " and so on. It does not have state transition model internally. Therefore, when a destination to move and information of the field are given, the output action is uniquely decided. "Reflective kick" in the next sub-section is defined in the layer.

Communication layer This layer has functions for communication.

Role table The role table is used to decide the role from the following four conditions.

- Rank: Rank of distance from the ball in the team (0–4)
- Relation: which team is near the ball? (not both, own team, opp. team, or both)
- Distance: distance from the ball (possessed, near, middle, or, far)
- Position: in which area is the ball as shown in Fig.15? (0–5)

An example of the table is shown in Table 2. The symbols in the table mean roles as shown in Table 3.

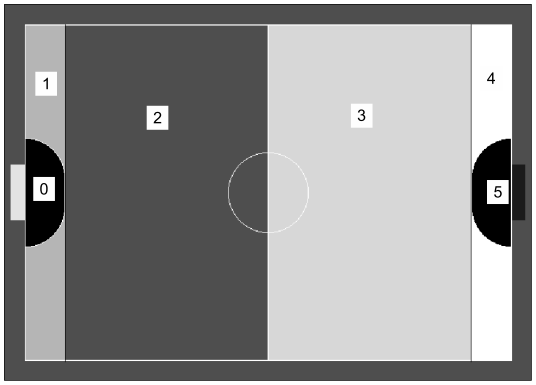


Fig. 15. Division of field

Table 2. Role table

Rank	Relation	both teams are far						my team is near						opponent team is near						both teams are near					
	Position / Distance	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
1	possessed	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1	AT1
	near	BG	AT1	AT1	AT1	AT1	AT1	BG	AT1	AT1	AT1	AT1	AT1	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG
	middle	BG	BG	BG	BG	BG	AT1	BG	BG	BG	BG	BG	AT1	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG
	far	BG	BG	BG	BG	BG	AT1	BG	BG	BG	BG	BG	AT1	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG	BG
2	possessed	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	GD	GD	AT2	AT2	AT2	AT2
	near	PC	PC	PC	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	GD	PC	AT2	AT2	AT2	AT2	GD	GD	AT2	AT2	AT2	AT2
	middle	GD	GD	PC	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	GD	GD	GD	AT2	AT2	AT2	GD	GD	GD	AT2	AT2	AT2
	far	GD	GD	GD	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	AT2	GD	GD	GD	AT2	AT2	AT2	GD	GD	GD	AT2	AT2	AT2
3	possessed	GD	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PC	PC	GD	GD	GD	GD	PW2	PC
	near	GD	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PC	PC	GD	GD	GD	GD	PW2	PC
	middle	GD	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PC	PC	GD	GD	GD	GD	PW2	PC
	far	GD	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PC	PC	GD	GD	GD	GD	PW2	PC
4	possessed	PW2	PW1	GD	GD	GD	GD	PW2	PW1	PW2	GD	GD	GD	PW2	GD	GD	GD	GD	GD	PW2	GD	GD	GD	GD	GD
	near	PW2	PW1	PW2	GD	GD	GD	PW2	PW1	PW2	GD	GD	GD	PW2	GD	GD	GD	GD	GD	PW2	GD	GD	GD	GD	GD
	middle	PW2	PW1	PW2	GD	GD	GD	PW2	PW1	PW2	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PW2	PW1	PW2	GD	GD	GD
	far	PW2	PW1	PW2	GD	GD	GD	PW2	PW1	PW2	GD	GD	GD	PW2	PC	GD	GD	GD	GD	PW2	PW1	PW2	GD	GD	GD
5	possessed	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD	PW2	GD	GD	GD	GD	GD
	near	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD	PW2	GD	GD	GD	GD	GD
	middle	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD
	far	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD	PW2	PW2	PW2	GD	GD	GD

AT1: Attacker1 PC: PassCutter PW1: PassWaiter1
AT2: Attacker2 GD: GoalDefender PW2: PassWaiter2
BG: BallGetter

Table 3. The Role symbols and meanings

Symbol	Role	Meaning
AT1	Attacker1	To attack the opponent team's goal positively.
AT2	Attacker2	To assist Attacker1.
BG	BallGetter	To take the ball which the opponent to have.
PC	PassCutter	To interfere with the pass between opponents.
PW1	PassWaiter1	To wait the pass from friends.
PW2	PassWaiter2	To wait the pass from friends.
GD	GoalDefender	To defend goal outside of penalty area.

Path planning and collision avoidance At first, as shown in Fig.16(a), the position the robot, the target, and the obstacles are given. The size of the obstacles enlarged so that the robot is considered to be point.

Next, temporary paths to the target are generated. The generating process is divided into step A as shown in Fig.16(b) and step B as shown in Fig.16(c). The process is carried out by calling them alternately.

The step A To check whether can move to the target linearly from the current position. When there are obstacles on the straight line, to find the point facing with the obstacle first.

The step B To follow the AABB(Axis Aligned Bounding Box) of the obstacles(the ball and robots).

After the path generation to the target is completed as shown in Fig.16(d), to perform "the visibility test" as shown in Fig.16(e). When all the path to the target are found, to choose the shortest path as shown in Fig.16(f).

When there are multiple obstacles, the paths to the target are shown in Fig.16(g). By visibility test, many paths are generated as shown in Fig.16(h). In such case the shortest path can be choosen by comparing the distances.

6.2 Reflective kick

Quick play is possible by the kick to the target without trapping the ball. We call it "Reflective kick", which is defined as a task in the task layer. Modeling of this task is as follows.

As shown in Fig.17, there is a robot on the predicted path of the ball that is approaching at the velocity. It is assumed that the direction of the ball after kick is given. Let be ϕ be the angle of the ball before kick, θ be the angle in direction of the front of robot, v be the velocity of the ball before kick, a be the kick velocity given in direction of the front, e_1 be the restitution coefficient in tangential direction, and e_2 be the restitution coefficient in normal direction . These parameter can be related in Eq.(15).

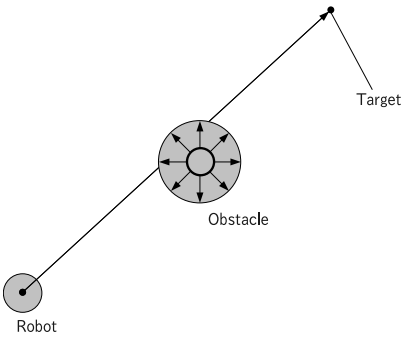
$$\tan(\phi - \theta) = \frac{e_1 v \sin\theta}{e_2 v \cos\theta + a} \tag{15}$$

To execute the reflective kick, the value of angle θ is needed, which is computed with Newton's method. In our case, $e_1 = 0.65$, $e_2 = 0.34$, and $a = 2000$[mm/s] are used.

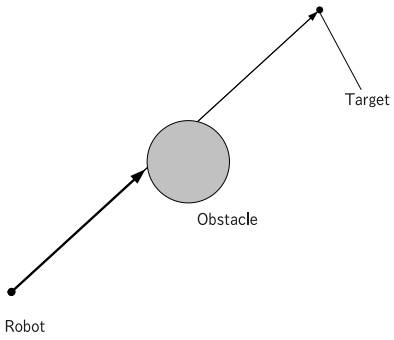
6.3 Value of safety

To decide the action of the robot, "value of safety" is defined, which evaluates the quality of the course of the ball. It is used in the task layer.

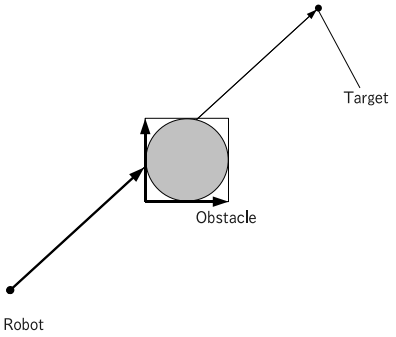
It is assumed that the course of the ball to be evaluated is given as a line segment. When the perpendicular is lowered from the nth obstacle (robot and goal post of the opponent team) in this segment, the distance from the starting



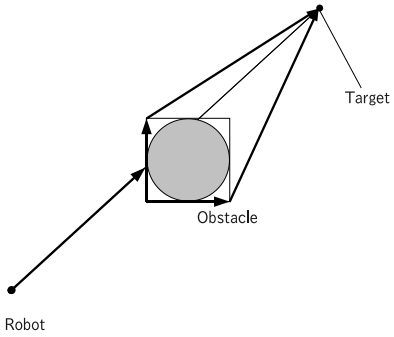
(a) In case of one obstacle



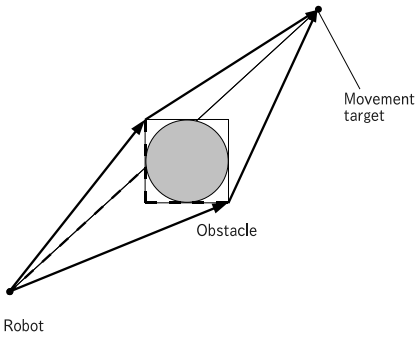
(b) The step A



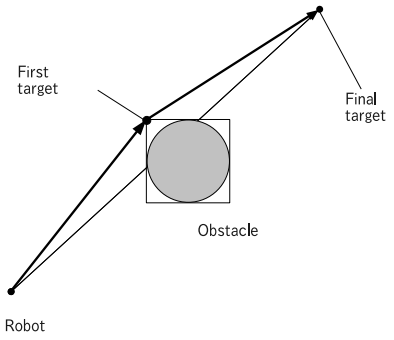
(c) The step B



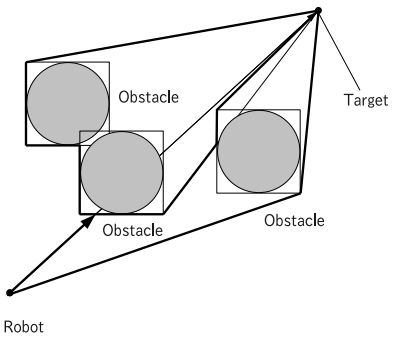
(d) A discover the target



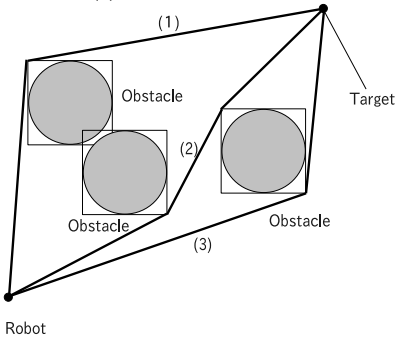
(e) Visibility test



(f) Choice of the target



(e) In case of three obstacles(The step A and B)



(h) In case of theree obstacles : After visibility test

Fig. 16. The path planing to avoid obstacles

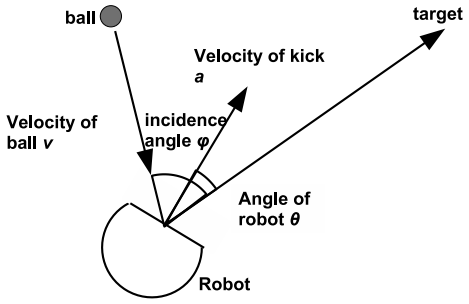


Fig. 17. Reflective kick

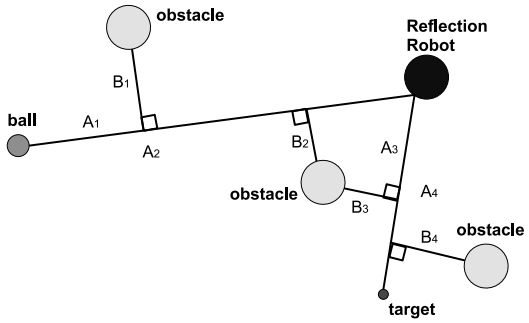


Fig. 18. Value of safety

point to the orthocenter is defined as A_n and the perpendicular length is defined as B_n. After these values are computed for all obstacles, the value of safety is defined in Eq. (16).

$$\alpha = \min_n \frac{B_n}{A_n} \tag{16}$$

When the course of reflective kick is evaluated as shown in Fig.18, value of safety is computed respectively for two segments before and after the kick, then the value of safety of the entire course is a sum of two values.

A Method of Placing Defenders The defenders (including the goalkeeper) of ODENS were poor at positioning. The fact is proven by some called games in the competitions that we participated in 2009. In order to overcome the problem, a new algorithm of placing defender is introduced. Moreover the placing goalkeeper is improved.

Table 4. Parameters for the value of danger

a	t_{delay}	$v_{b_{max}}$	$v_{b_{pass}}$
acceleration	time delay	shoot velocity	pass velocity
2627[mm/s^2]	0.149[s]	10[m/s]	5[m/s]

Value of danger To decide positions were defenders should defend, we defin "value of danger". It assesses of possibility to be aimed by opponents for a point on the own goal line. We consider two possibilities in case that a ball owner shoot directly (named direct shot) and in case that a friend who get a pass from a ball owner shoots (named indirect shot).

The method of computing the value of danger is as follows. First, the own goal mouth is equally divided into eight parts. Let $i(i = 0, 1, 2, \ldots, 8)$ be the i-th dividing point. Next, Shoot lines for i are defined. For the direct shot, the shoot line is defined as the line which links the i-th point to the ball. For the indirect shot, the shoot line is defined as the lines which links the i-th to each opponents (except the ball owner). If the line intersects to a range of the goalkeeper, the value of danger is defined as zero. The range that the goalkeeper can prevent is the range that the goalkeeper can move during before the ball arriving at the goalkeeper. Let (x_b, y_b) be coordinates of the ball, (x, y) be coordinates of the goalkeeper, $(x_n, y_n)(n = 1, 2, 3, 4, 5)$ be coordinates of the n-th opponents. The range is computed by Eq.(17) for the direct shot, or by Eq.(18) for the indirect shot.

$$t_D = \frac{\sqrt{(x - x_b)^2 + (y - y_b)^2}}{v_{b_{max}}} - t_{delay}$$
$$r_D = \frac{a{t_D}^2}{2} \tag{17}$$
$$t_{I_n} = \frac{\sqrt{(x_b - x_n)^2 + (y_b - y_n)^2}}{v_{b_{pass}}} + \frac{\sqrt{(x_n - x)^2 + (y_n - y)^2}}{v_{b_{max}}} - t_{delay}$$
$$r_{I_n} = \frac{a{t_{I_n}}^2}{2} \tag{18}$$

The parameters in Eqs.(17) and (18) we use are shown in Table 4. If the line cannot be prevented by the goalkeeper, the value of danger is computed by Eq.(19) for the direct shot, or by Eq.(20) for the indirect shot.

$$\alpha_i = p\frac{1}{\sqrt{(x_b - x_{g_i})^2 + (y_b - y_{g_i})^2}}, (i = 0, 1, 2, \ldots, 8) \tag{19}$$
$$\beta_{i,n} = q\frac{1}{\sqrt{(x_n - x_{g_i})^2 + (y_n - y_{g_i})^2}}, (i = 0, 1, 2, \ldots, 8) \tag{20}$$

where p and q are appropriate weight coefficients.

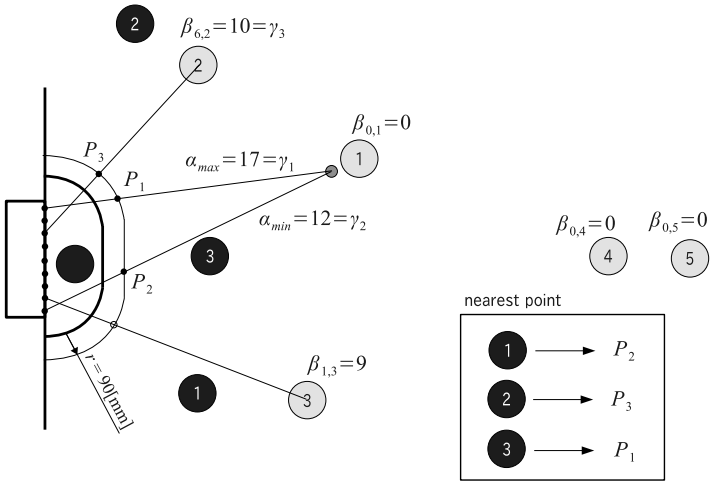


Fig. 19. An example of the method of placing defenders

Placement based on value of danger Among the values of danger computed above, the following seven values are selected; the maximum and minimum values of α_i for the direct shot and the median values of $\beta_{i,n}$ for the indirect shot from every n-th opponent. Let γ_k be the k-th values of the selected values. Moreover m higher values are selected from γ_k , where m is the number of defenders decided by the role layer.

The defence place corresponding to the value γ_k is the point of intersection with the line where 90[mm] just extended an own penalty area line and the shoot lines of the k. At the point with the highest value, the defender nearest to it is placed. At the other points, the nearest defender is placed among the defenders whose places are not decided. An example of the method of placing defenders based on the value of dangers is shown in Fig.19.

7 Conclusion

As mentioned above, the part of deciding action is perfectly separated from image processing and robot control in the system. Owing to this feature, this system became useful platforms for both education and research. For the purpose of education, many students learned robot programing by using it. On the other hand, ODENS won the 4th in the RoboCup 2009 Graz by using the same system.

Although under the SSL regulation it is possible to make centralized system, in the system of ODENS, the program for each robot is independent of others. In the future, following this policy we will study robot system and participate in competitions.

References

[1] K. Kanaya et al., "ODENS 2009 Team Description", RoboCup2009 (2009).
[2] J. Maeno et al., "RoboDragons 2009 Team Description", RoboCup2009 (2009).
[3] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Velosso: SSL vision: The Shared Vision System for the RoboCup Small Size League, RoboCup 2009: Robot Soccer World Cup XIII, Springer-Verlag (2010).
[4] R. Smith et al., "Open Dynamics Engine", `http://www.ode.org/`