RoboDragons 2015 Extended Team Description

Yusuke Adachi, Hiroyuki Kusakabe,Yuya Yamanaka,
Masahide Ito, Kazuhito Murakami, and Tadashi Naruse

Aichi Prefectural University, Nagakute, Aichi 480-1198, JAPAN
Email: is121004@cis.aichi-pu.ac.jp

Abstract. This paper describes a system configuration of RoboDragons 2015, Aichi Prefectural university's participating team for RoboCup small size league (SSL). On the robot hardware, we basically use the robots developed in 2012. We changed the radio system from 2.4 GHz wireless LAN to 5 GHz and adjusted the dribble device and chip kick bar to kick the strong ball and fly higher, respectively. On the soccer program, the base program is the one developed in CMRoboDragons joint team in 2004 and 2005. After the end of the joint team, we are continuously improving the program by introducing our research results. We describe two algorithms in this paper. One is a dynamic ball kicking algorithm which is often used in the attacking strategy. Another is a circle-and-pass motion algorithm which is used at throw-in. The effectiveness of these hardware and software improvements is experimentally shown.

1 Introduction

RoboDragons 2015 is the team of Aichi Prefectural university (APU) participating in the RoboCup small size league (SSL). It started in 1997 as a joint team between APU and Chubu university; team name was Owaribito. In 2002, as each university had been able to develop their robot system independently, we started a new team, RoboDragons. Since then, RoboDragons participate in SSL every year, including the joint team CMRoboDragons (2004, 2005) with Carnegie Mellon university. Our record is the 2nd place in 2009. Other than that, two 3rd places (2007, 2014) and three 4th places (2004, 2005, 2013).

In this paper, we describe a system configuration of RoboDragons 2015. On the robot hardware, we basically use the robots developed in 2012. We used 2.4 GHz wireless LAN for communication between robots and host computer in the early design and it worked very well in Japan open. However, in RoboCup 2013, the interference was harsh since several leagues shared a hall and the hall's radio system also used the same wireless LAN. Therefore, we changed the radio system from 2.4 GHz to 5 GHz in 2014. We had another problems, the dribbler and the chip kicker. It was not stable for the dribbler to keep the ball. With trial and error, we improved the mechanical design of the dribbler this year. For the chip kicker, the power to fly the ball higher and further was needed. For the limitation of the size of the solenoid coil, we changed that shape of the chip kick bar. In the section 2, we describe an overview of the robot hardware and the modifications for the dribbler and the chip kicker.

Our base soccer program is the one developed in the CMRoboDragons joint team in 2004 and 2005. After the end of the joint team, we are continuously improving the program by introducing our research results. In section 4, we describe a dynamic ball kicking algorithm. This is an algorithm that a robot runs after the ball as turning its face toward the target direction and kicks the ball. In section 5, we describe a circle-and-pass motion used at throw-in. This is an algorithm that may disturb the opponents, because the robot can quickly change its kick direction by going around the ball.

Finally, the experimental results of these hardware and software improvements are also shown in the appropriate sections.

2 Robot Hardware

The robots we use now in our laboratory are sixth generation robots. The major part of the robots have been developed in 2012. Their main features are,

- Cylinder with dimensions of 125 mm height and 178 mm diameter.
- Weight : 2.3 kg.
- Maximum percentage of the ball coverage : about 18%.
- Motor : 50 watt DC brushless motor for driving a wheel.
- Simple proximity sensor.
- Wireless LAN for communication.

The robot is shown in Figure 1. Each component of the robots is summarized in Table 1 and its photos in Figure 2. In the following subsection, we briefly explain each component.

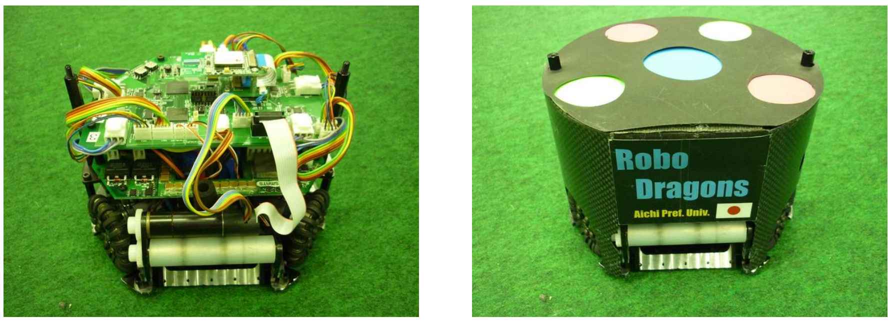
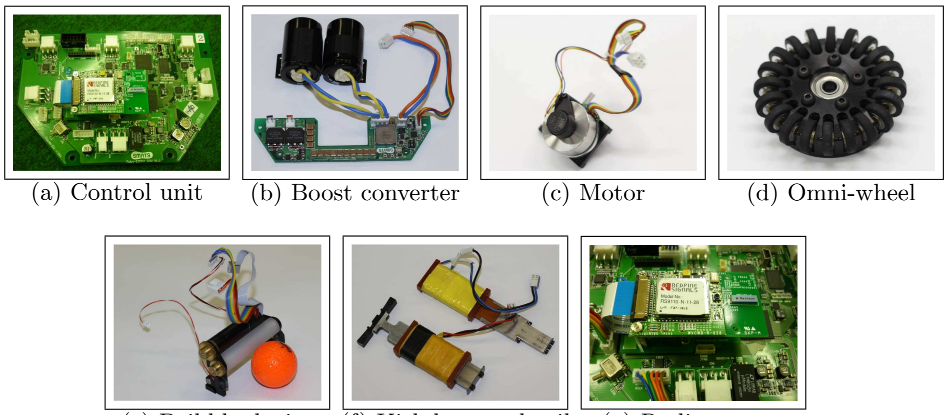


Fig. 1: Current robot developed in 2012 (modified in 2015)
(Left: without the cover, Right: with the cover)

Table 1: Summary of the robots

Device	Description
Control Unit	CPU: SH2A processor (Renesas Electronics Corporation) operated with 196 MHz clock. Peripheral circuits (except analog circuits) are almost in the Xilinx's Spartan-6 FPGA.
Boost Converter	Convert from 18.5 V DC to 150 V - 200 V DC. Condenser has a capacity of 4400 μF. Charging time is about 2 s (when output voltage is 200 V).
Motor	Maxon "EC 45 flat 50 W". Gear reduction ratio between motor and omni-wheel is 21:64.
Wheel	4 omni-wheels, each has 20 small tires in circumference. Diameter: omni-wheel 55 mm, small tire 12.4 mm.
Dribble Device	Dribble roller: 16 mm in diameter and 73 mm in length, made of aluminum shaft with silicon rubber. Motor is Maxon "EC 16 30 W".
Ball Sensor	Infra-red light emission diode and photo diode pair.
Kicker	Kick bar is made of 7075 aluminum alloy. Solenoid is a coil winding 0.6 mm$^{\phi}$ enameled wire. Straight kicker kicks a ball with over 8 m/s velocity at maximum. Chip-kicker kicks a ball as far as 4 m distance at maximum.
Communication	IEEE 802.11abgn 2.4/5 GHz wireless LAN. (New)


(a) Control unit (b) Boost converter (c) Motor (d) Omni-wheel

(e) Dribble device (f) Kick bars and coil (g) Radio system

Fig. 2: Components of the robot

2.1 Components of the robot

Control unit Fig. 2(a) shows the picture of the control board and the layout of the board is shown in figure 3. The control unit consists of CPU, FPGA, motion sensor, infrared sensor for detecting the ball and motor control circuits. The blank part of the board in Fig. 3 is mainly an area for connectors.

The operating system used is the TOPPERS[5] (Toyohashi OPen Platform for Embedded Real-time Systems), which is developed in Toyohashi University of Technology based on the ITRON[6] specifications and aimed to develop base software for use in embedded systems.

The robot control program is written in the language C.

Boost Converter Fig. 2(b) is a boost converter board (and condensers). The boost converter is redesigned and implemented in a flat board shown in Fig. 2(b). It makes the height of the robot lower than the 5th generation robots. It converts source voltage (14.8 volts) to about 200 volts and charges in the condensers within 1.3 second.

Motors and wheels Fig 2(c) and (d) show the motor and the omni-wheel. The motor is the Maxon's EC 45 flat 50 watt motor with encoder attached. The omni-wheel has 20 small tires around the large wheel, 5 more small tires than the 5th generation robot.

Dribble device Fig. 2(e) shows a dribble device. The dribble roller (white) is directly driven by the motor (black) through the gears. The photo diode and LED sensor pair is attached to the black frames (though not seen clearly). The silver (short) shafts in the frames are stoppers that stop the chip kicker not to move further up and let the ball go upward 45 degree direction.

Kicker RoboDragons 2015 has 2 kickers, straight and chip kickers, as other teams have. Fig. 2(f) shows solenoids. The right-upper is the solenoid for the chip kicker and the left-lower is the one for the straight kicker. The kick bars are made of 7075 aluminum alloy and the coils are made of winding 0.6mm^{ϕ} enameled wire.

Radio system We have used the radio modems of Futaba Co. in the 5th generation robots because of its stability of radio communication. However, its communication speed was low (19K bps). It is not enough for communication of the new robots, so we adopted the wireless LAN. Fig. 2(g) shows the radio device on the robot. The radio module is replaced by the 5 GHz wireless LAN in 2014. New module is a Redpine Signals RS9110-N-11-28.

2.2 Some Robot Hardware Improvements

One problem is an unstable dribbling. We moved the position of the dribble roller 1 mm higher. With this, the roller became keeping the ball stably. (Just adjustment.) Another is the chip kicker. The chip kick distance of the ball is less satisfied in the original robots. To overcome the problem, we made the surface of the bar coarse. As a result, the bar grips the ball better than the previous one and the chip kicker can fly the ball higher and further.

Figure 4 shows the experimental result of the flying distance. It is clear that the improved chip kicker can fly the ball further away.

2.3 Robot control program

The block diagram of the robot control program is shown in Figure 5. In the figure, each box named module is a thread program which runs independently and other boxes are hardware devices which are controlled by the modules.

Robot command is sent from the host computer to each robot by the command packet (see next section) every 1/60 seconds through wireless LAN. The command packet is broadcasted to all robots. The command packet has error correcting code (ECC) to make reliable communication possible. We use the Humming code as the ECC.

The communication module receives the command packet and extracts the robot command of its own. If error is detected, the command is abandoned. As far as the burst error is not occurred, abandoning the command is a good alternative. Extracted command is sent to the command module.

In the command module, the velocity of each wheel is calculated from the robot velocity, moving direction and angular velocity values (see Table 2) and the wheel velocities are sent to the motor control module. Moreover, the kick and dribble command is executed in this module.

In the motor control module, the PID control is performed. This is done by using the target velocity given by the robot command and the current velocity calculated from the encoder pulses. The PWM control is used for the motor driving.

2.4 Configuration of command packet

Thanks to the fast communication ability of the radio system, we can define a bit large command packet. The configuration of the command packet is as follows. The command packet consists of 20 byte header, 49 byte packet body and 2 byte footer. The packet body consists of 8 byte command for each robot and 1 byte common command for all robots.

The 8 byte command is shown in Table 2. Basic idea of the command is that we give the moving vector and the angular velocity of the robot. In the 6th and 7th bytes, we give the command. "hhh" field selects kicker (straight/chip) and kicking mean of kick (normal/forced). The normal means to kick when the ball sensor detects the ball while the forced means to kick just after the command is issued. The 1 byte command for all robots is mainly used for debug purpose.

3 Software System

3.1 Overview of the software system

In this section, we show how our software system in host computer is composed and relates to the information from real world. The overview of our software system is shown in Figure 6.

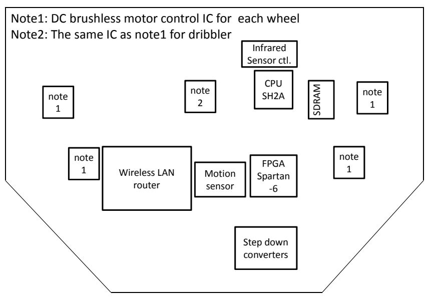


Fig. 3: Board layout

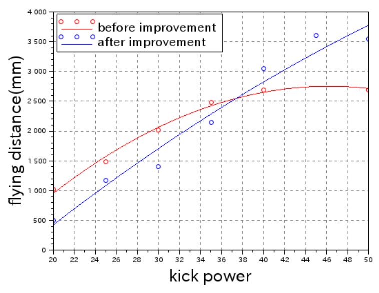


Fig. 4: The result of fitting a set of distance data points with a quadratic function

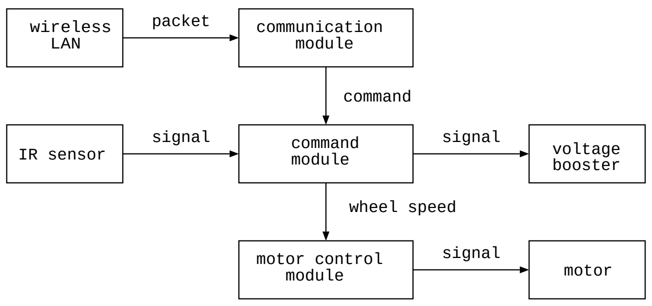


Fig. 5: Software configuration of the robot

Table 2: Command packet for controlling each robot

	Config.	Description
1st byte	`aaaabbbb`	`aaaa`: Robot ID, `bbbb`: Robot velocity(Upper 4bits)
2nd byte	`bbbbbbbb`	`bbbbbbbb`: Robot velocity(Lower 8bits), 0 - 4095 (mm/s)
3rd byte	`cccccccc`	`cccccccc`: Moving direction(Upper 8bits), Resolution is $2\pi/512$ radian
4th byte	`ddddefff`	`dddd`: Dribble velocity, `e`: Rotation direction, `fff`: Angular velocity(Upper 3bits)
5th byte	`ffffffff`	`ffffffff`: Angular velocity(Lower 8bits), 0 - 2047 (deg/s)
6th byte	`gggggggg`	`gggggggg`: Kick force, 256 levels
7th byte	`chhh0000`	`c`: Moving direction(Last bit), `hhh`:Normal/Forced kick
8th byte	`iiiiiiii`	`iiiiiiii`: CRC code

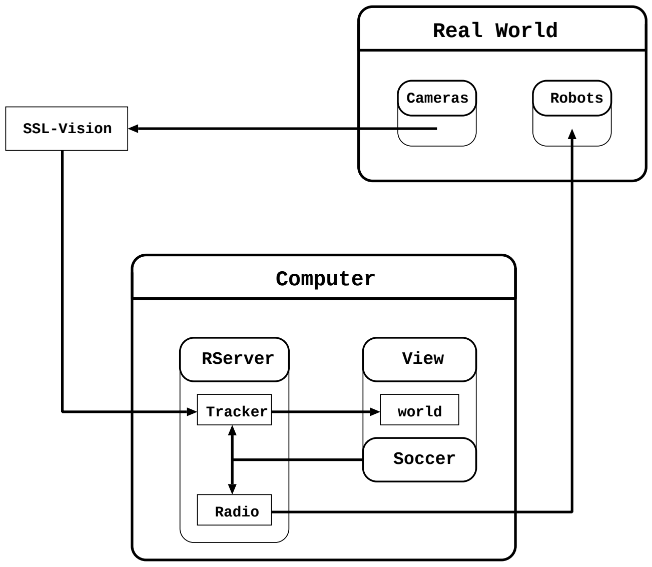


Fig. 6: Overview of the software system

The host computer is an off-the-shelf notebook computer. The CPU is Intel(R) Core(TM) i7 4700MQ and the main memory is 4 GB. The operating system is the Linux (lubuntu 14.04).

Our software system is divided into three main modules:

(1) The *Rserver* module receives the data sent from the SSL-Vision system and compensates the positions of the ball and robots by using the Kalman Filter in the tracker submodule. compensated positions are stored in the memory as the world data, which are shared with the view and soccer modules. The Rserver also sends the command packet to each robot through the radio submodule.

(2) The *View* module is a graphical user interface module. It displays the current soccer state which the team operator wants to know and take commands from the operator. The software system has a soccer simulator so that the simulation result is also displayed by the View module.

(3) The *Soccer* module makes an action command for each robot. By using the world data, the module chooses the best strategy for the current situation, gives each robot a role to take under the chosen strategy, and calculates a moving path to perform the role for each robot.

4 Dynamic Ball Kicking

One of the useful attacking skills is that a running robot kicks a moving ball. It will be often used when a robot runs after the ball as turning its face toward the target direction and kicks the ball. Figure 7 shows a typical situation. Robot R runs after the ball B and then comes to R' as turning its face toward target position G. (The ball comes to B' at the time.) Meanings of the symbols used in the figure are listed in Table 3. In the list, some other symbols are also listed, which will be used later.

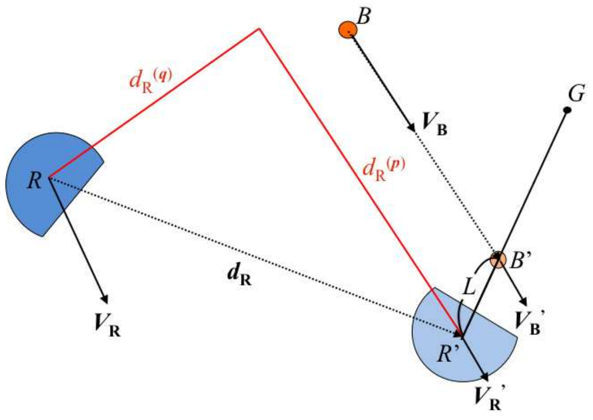

B
G
$d_R^{(q)}$
$\boldsymbol{V}_\mathbf{B}$
R
$d_R^{(p)}$
B'
L
$\boldsymbol{d}_\mathbf{R}$
$\boldsymbol{V}_\mathbf{B}'$
$\boldsymbol{V}_\mathbf{R}$
R'
$\boldsymbol{V}_\mathbf{R}'$

Fig. 7: Dynamic ball kicking

If the robot takes T seconds to move from R to R' and the ball does not get any force except for friction, the ball velocity $\boldsymbol{V}'_\mathbf{B}$ and the ball position B' is calculated as follows.

$$\boldsymbol{V}'_\mathbf{B} = \boldsymbol{V}_\mathbf{B} - D_\mathrm{B} T \frac{\boldsymbol{V}_\mathbf{B}}{|\boldsymbol{V}_\mathbf{B}|}, \quad B' = \frac{1}{2}(\boldsymbol{V}_\mathbf{B} + \boldsymbol{V}'_\mathbf{B})T + B \tag{1}$$

With this preparation, let's discuss the control algorithm to realize such robot motion. First, we decompose the robot velocity vector, $\boldsymbol{V}_\mathbf{R}$, into 2 components, $\boldsymbol{V}_\mathbf{R}^{(\boldsymbol{p})}$ and $\boldsymbol{V}_\mathbf{R}^{(\boldsymbol{q})}$, in the direction of $\boldsymbol{p}$ and $\boldsymbol{q}$. Then, at R', if we can control the velocity $\boldsymbol{V}'_\mathbf{R}$ to be $\boldsymbol{V}_\mathbf{R}^{'(\boldsymbol{p})} = \boldsymbol{V}'_\mathbf{B}$ and $\boldsymbol{V}_\mathbf{R}^{'(\boldsymbol{q})} = 0$, the robot can easily kick the ball by approaching the ball.

If T is given, we can calculate the ball position B' by Eq. (1). Then, we can make a motion profile of the robot for $\boldsymbol{V}_\mathbf{R}^{(\boldsymbol{p})}$ and $\boldsymbol{V}_\mathbf{R}^{(\boldsymbol{q})}$, respectively, because we can calculate R', $\boldsymbol{d}_\mathbf{R}$, $\boldsymbol{d}_\mathbf{R}^{(\boldsymbol{p})}$, and $\boldsymbol{d}_\mathbf{R}^{(\boldsymbol{q})}$. On the contrary, if the motion profile of the robot for a direction $\boldsymbol{p}$ is made, we can calculate T. By iterating this process, we can make the motion profiles realizing the run-kick skill shown in Fig. 7.

Table 3: Symbols list

B, B':	coordinate of the ball
R, R':	coordinate of the robot
$\boldsymbol{V_{\mathbf{B}}}, \boldsymbol{V'_{\mathbf{B}}}$:	velocity vector of the ball
$\boldsymbol{V_{\mathbf{R}}}, \boldsymbol{V'_{\mathbf{R}}}$:	velocity vector of the robot
G:	target position
L:	offset length
$\boldsymbol{p}, \boldsymbol{q}$:	$\boldsymbol{p}$ is the unit vector in the direction of $\boldsymbol{V_{\mathbf{B}}}$. $\boldsymbol{p} \perp \boldsymbol{q}$.
$\boldsymbol{d_{\mathbf{R}}}$:	vector from R to R'
$\boldsymbol{d_{\mathbf{R}}^{(p)}}$:	component of vector $\boldsymbol{d_{\mathbf{R}}}$ in the direction of $\boldsymbol{p}$
$\boldsymbol{d_{\mathbf{R}}^{(q)}}$:	$= \boldsymbol{d_{\mathbf{R}}} - \boldsymbol{d_{\mathbf{R}}^{(p)}}$
D_{B}:	ball deceleration due to the friction. (Assume it is constant.)
$V_{\max}$:	maximum velocity of the robot
A_{R}:	maximum acceleration of the robot
D_{R}:	maximum deceleration of the robot

4.1 Making a motion profile for the velocity $\boldsymbol{V_{\mathbf{R}}^{(p)}}$

First, we put following assumptions,

$$d_{\mathrm{R}}^{(\boldsymbol{p})} \geq d_{\mathrm{R}}^{(\boldsymbol{q})} \quad \text{and} \quad V_{\mathrm{R}}^{(\boldsymbol{p})} \geq |\boldsymbol{V_{\mathbf{B}}}|.$$

A motion profile shown in Figure 8 is made for given T, where, we assume $A_{\mathrm{R}}^{'} = \frac{1}{\sqrt{2}} A_{\mathrm{R}}$, $D_{\mathrm{R}}^{'} = \frac{1}{\sqrt{2}} D_{\mathrm{R}}$. In Fig. 8, variables ($|\boldsymbol{V_{\mathbf{B}}^{'}}|$, b, T_1, T_2, V_1, d) are calculated by the following equations.

$$\begin{aligned}
|\boldsymbol{V_B^{'}}| &= -D_B T + |\boldsymbol{V_B}| \\
b &= |\boldsymbol{V_B^{'}}| + D_R^{'} T \\
T_1 &= \frac{b - V_R^{(\boldsymbol{p})}}{D_R^{'} + A_R^{'}} \\
T_2 &= T - T_1 \\
V_1 &= A_R^{'} T_1 + V_R^{(\boldsymbol{p})} \\
d &= \frac{1}{2}\{(V_R^{(\boldsymbol{p})} + V_1) T_1 + (V_1 + |\boldsymbol{V_B^{'}}|) T_2\}
\end{aligned}$$

In motion profile, since the yellow area in Fig. 8 is equal to the distance the robot moved, i.e. d. Therefore, if $d = |\boldsymbol{d_{\mathbf{R}}^{(p)}}|$, we can get the desired motion profile with the final $\boldsymbol{V_{\mathbf{R}}^{'(p)}}$ being equal to $|\boldsymbol{V_{\mathbf{B}}^{'}}|$ at R'. Then, how can we calculate T? Bisection or Newton method[7]. The initial value of T is, for instance, the time when the ball is just stopped by the constant ball deceleration D_{B}.

In case of $V_1 > V_{\max}^{'}$ ($= \frac{1}{\sqrt{2}} V_{\max}$), we make the motion profile as shown in Figure 9.

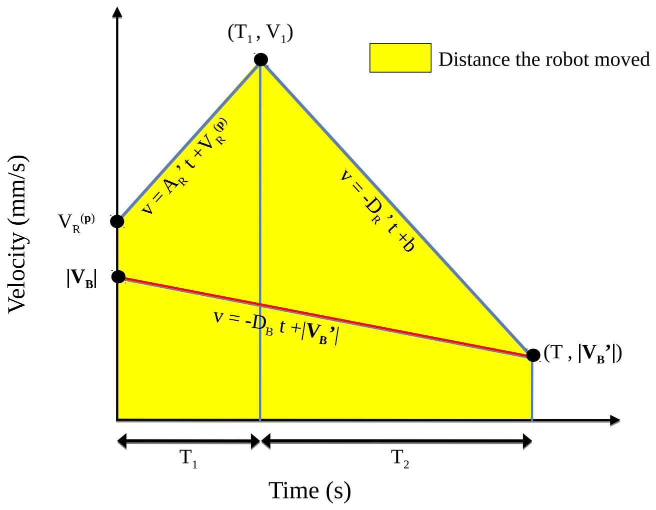


Fig. 8: Motion Profile

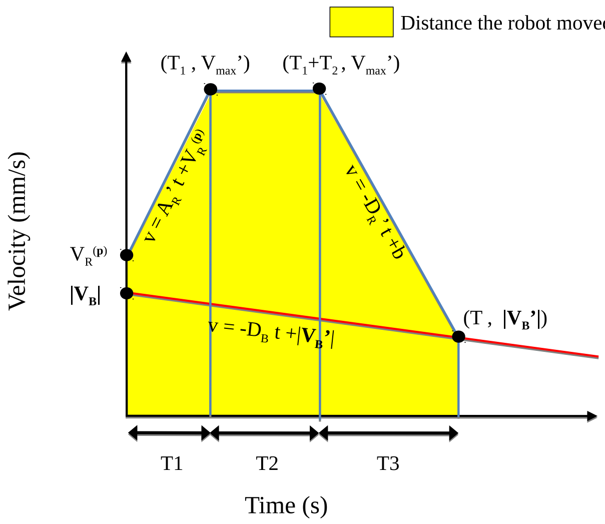


Fig. 9: Motion profile 2

In the figure, variables ($|\boldsymbol{V}_{\boldsymbol{B}}^{'}|$, b, T$_1$, T$_2$, T$_3$, d) are calculated by the following equations.

$$\begin{aligned}
|\boldsymbol{V}_{\boldsymbol{B}}^{'}| &= -D_B T + |\boldsymbol{V}_{\boldsymbol{B}}| \\
b &= |\boldsymbol{V}_{\boldsymbol{B}}^{'}| + D_R^{'} T \\
T_1 &= \frac{V_{max}^{'} - V_R^{(\boldsymbol{p})}}{A_R^{'}} \\
T_2 &= \frac{b - V_{max}^{'}}{D_R^{'}} - T_1 \\
T_3 &= T - (T_1 + T_2) \\
d &= V_{max}^{'} T - \frac{1}{2}\{\ (V_{max}^{'} - V_R^{(\boldsymbol{p})}) T_1 + (V_{max}^{'} - |\boldsymbol{V}_{\boldsymbol{B}}^{'}|) T_3\ \}
\end{aligned}$$

4.2 Making a motion profile for the velocity $\boldsymbol{V}_{\mathbf{R}}^{(\boldsymbol{q})}$

By the method discussed in section 4.1, we can get the value of T. With this T, we can calculate the position R' and hence $\boldsymbol{d}_{\mathbf{R}}^{(\boldsymbol{q})}$. Therefore, we can make a motion profile for $\boldsymbol{V}_{\mathbf{R}}^{(\boldsymbol{q})}$. In this motion profile, the velocity at R' is equal to 0. In the calculation of the motion profile, we use $A_{\mathrm{R}}^{'}$, $D_{\mathrm{R}}^{'}$ and $V_{\mathrm{max}}^{'}$ that are defined in section 4.1.

From the two motion profiles, a velocity vector $\boldsymbol{V}_{\mathbf{R}}$ is produced, i.e. $\boldsymbol{V}_{\mathbf{R}}(t) = V_{\mathrm{R}}^{(\boldsymbol{p})}(t)\boldsymbol{p} + V_{\mathrm{R}}^{(\boldsymbol{q})}(t)\boldsymbol{q}$.

Angular velocity is also calculated so that the front of the robot face rotates to face the target position G. In our system, constant angular velocity is adopted.

5 Circle-and-Pass Skill in Throw-in

So far, we have used the rapid random tree algorithm, RRT, for path planning[4]. In the throw-in, however, an easy and opponents disturbing path planning is welcomed in order to improve the success rate of passing between teammate robots. We have designed a new skill named circle-and-pass as shown in Figure 10. This skill starts when the robot gets in the circle with a radius of $r + \alpha$ and center at the ball. We assume that the front of the robot faces the ball when it gets in the circle. In this skill, the kicker first goes around the ball with its front facing the ball and then kicks the ball. As a result, the robot can easily kick the ball toward the desired direction and the opponent robots will be disturbed by the quick change of the kicking direction of the kick robot. In the next section, we give the algorithm.

5.1 Algorithm

We describe the circle-and-pass algorithm. First, we define symbols used in the algorithm.

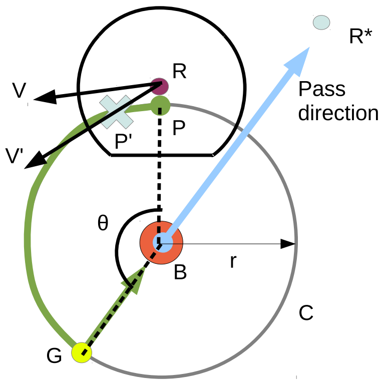


Fig. 10: Overview of the circle moving pass

Let B, R, and R^* be the positions of the ball, the kicker robot and the target robot, respectively. Consider a circular path C around B with radius r. Let P be the intersection of C and the line segment $\overline{BR}$ and let G be the intersection of C and the line segment $\overline{BR^*}$. Furthermore, let D and θ be the length and angle of the arc PG, respectively. In the following, a frame period (FP) means 1/60 seconds and "an angle is small enough" means the angle is less than an appropriate threshold (which will be determined by the experiment).

algorithm Circle-and-Pass skill

Step 1. Move the robot R along the arc PG with its front facing the ball B. If the robot is close enough to G and the angle between the pass direction vector and the forward direction vector of the robot is small enough for three consecutive frame periods, then the robot changes the moving direction and moves toward the ball and kicks the ball.

Step 2. For present velocity vector $\boldsymbol{V}$, calculate a tangential component vector to the circle. Let it be $\boldsymbol{V_0}$ and let v_0 be $|\boldsymbol{V_0}|$. If v_0 is less than $|\boldsymbol{V}| - d_m \cdot FP$, let v_0 be $|\boldsymbol{V}| - d_m \cdot FP$ in order to avoid uncontrolled situation.

Step 3. Make a velocity profile on the arc PG with the initial velocity v_0. From the velocity profile, calculate the move distance d on the arc PG after 1 FP. (See Figure 11 and 12.) Let it be P'.

Step 4. Let the length of the line segment $\overline{RP'}$ be D' and let the $\overline{RP'}$ direction component of the vector $\boldsymbol{V}$ be $\boldsymbol{V'}$ and its size be v'_0. If v'_0 is less than $|\boldsymbol{V}| - d_m \cdot FP$, let v'_0 be $|\boldsymbol{V}| - d_m \cdot FP$.

Step 5. Calculate an acceleration a to move D' mm in 1 FP second with an initial speed v'_0.

$$a = 2(\frac{D'}{FP^2} - \frac{v_0}{FP})$$

If a is greater than the maximum acceleration or maximum deceleration, a is truncated.

Step 6. Calculate the velocity v_1 at the time when the robot arrived at P' and let $\boldsymbol{V'_{new}} = v_1 \cdot \boldsymbol{V'}/||\boldsymbol{V'}||$.

Step 7. With the velocity $\boldsymbol{V'_{new}}$, the robot is moved. Then, go to step 1.

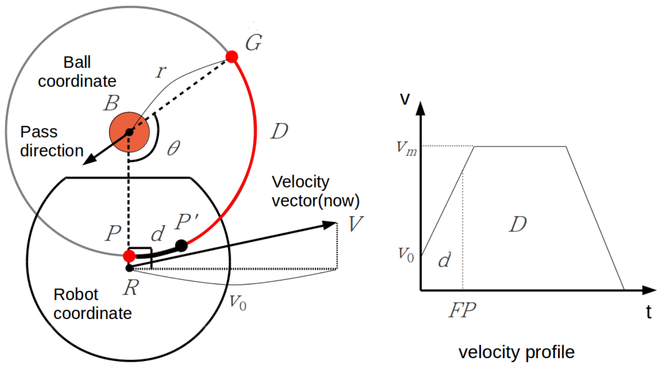


Fig. 11: Computation of P' based on the velocity profile (case 1)

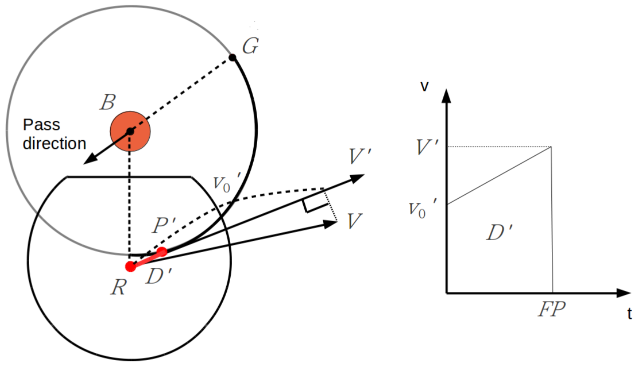


Fig. 12: Computation of P' based on the velocity profile (case 2)

5.2 Comparisons

We compared the circle-and-pass algorithm with the traditional RoboDragons' throw-in algorithm.

For the initial angle values of 130 degree and 160 degree, which are occurred often in throw-in, we carried out the experiments for each algorithm. The robot start at 119 mm away from the ball at first, then 129 mm away ... up to 189 mm away (every 10 mm). In the circle-and-pass motion, the radius r is equal to the distance from the ball to the start position. The values shown in the table are the best values in the experiments, for example, for rotation angle 130 degree, robot start at 119 mm away from the ball brought the best result for the circle-and-pass algorithm.

Table 4: Comparison of circle-and-pass and traditional RoboDragons' algorithms

rotation angle (θ)	130°		160°	
algorithm	circle	traditional	circle	traditional
start distance from ball (mm)	119	149	119	149
time necessary from start to pass (s)	1.12	1.19	1.24	1.63
moving distance (mm)	432	485	498	797
maximum distance from ball (mm)	126	194	129	277
error of pass (rad)	0.020	0.023	0.024	0.029

From the table, new algorithm shows the better performance than the traditional one.

6 Concluding Remarks

We have shown the RoboDragons system's hardware and software configuration, and useful algorithms in this paper. The robots are introduced in 2012 and improved every year. Major changes are the radio module in 2014 and the chip kick bar in 2015. In soccer program, we have developed a basic skill that a running robot kicks a moving ball in 2012. Using this skill, we developed many skills such as a run-pass, a run shoot, a star passing (for demo) and so on. In this paper, we show a revised description of the basic skill, which original version was shown in 2012 ETDP. A circle-and-pass motion algorithm has been using since 2013. However, we haven't shown it yet, we showed the detailed description of the algorithm this year.

References

1. Akeru Ishikawa, Takashi Sakai, Jousuke Nagai, Taro Inagaki, Hajime Sawaguchi, Yuji Nunome, Kazuhito Murakami and Tadashi Naruse "RoboDragons 2010 Team Description", RoboCup 2010 symposium CDROM, 2010
2. Kotaro Yasui, Taro Inagaki, Hajime Sawaguchi, Yuji Nunome, Hiroaki Sasai, Yuki Tsunoda, Shinya Matsuoka, Naoto Kawajiri, Togo Sato, Kazuhito Murakami and Tadashi Naruse "RoboDragons 2012 Team Description", RoboCup 2012 symposium CDROM, 2012
3. Kotaro Yasui, Yuji Nunome, Shinya Matsuoka, Yusuke Adachi, Kengo Atomi, Masahide Ito, Kunikazu Kobayashi, Kazuhito Murakami and Tadashi Naruse "RoboDragons 2013 Team Description", RoboCup 2013 symposium CDROM, 2013
4. James Bruce, Manuela Veloso, "Real-Time Randomized Path Planning for Robot Navigation", Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on Volume:3, pp.2383 - 2388, 2002
5. http://www.toppers.jp/en/index.html
6. http://en.wikipedia.org/wiki/TRON_project and
   http://en.wikipedia.org/wiki/ITRON
7. "Newton's Method"
   <http://www.math.montana.edu/frankw/ccp/calculus/numerical/newton/learn.htm>