RoboDragons 2014 Team Description

Yuji Nunome, Tomonori Hibino, Akifumi Hosino, Shohei Yokota,
Yusuke Adachi, Masahide Ito, Kunikazu Kobayashi,
Kazuhito Murakami and Tadashi Naruse

Aichi Prefectural University, Nagakute city, Aichi, 480-1198 JAPAN

Abstract. This paper describes a system configuration of RoboDragons, a team of Aichi Prefectural University, Japan. The robots were newly developed last year. The features of the robot are to use a 50 watt DC blushless motor for driving an omni-wheel, an improved chip-kicker, a simple proximity sensor, and a wireless LAN for communication. Software on the RoboDragons' system is almost the same as the one used last year. However, we improved the estimation method of moving-time of the robot. We describe it in this paper. We propose a method to estimate the ball's state of spin. It is useful for estimating the speed of the ball after chip-kicked with strong spin and bounced. It is desired to be implemented in the shared vision system.

1 Introduction

In the team description paper (TDP) of RoboDragons 2014, we summarize the hardware and the software architecture of our system and then describe in detail the improvement of the software done this year.

For the hardware, we developed a new robot last year. Features of the new robot are dimension with 125 mm height and 178 mm diameter cylinder, a 50 watts DC brushless motor for driving an omni-wheel, an improved chip-kicker, a simple proximity sensor, and a wireless LAN for communication. The detail is written in RoboDragons 2013 TDP [1].

Software development is crucial for the performance of the robot system. Though we use almost the same software as the 2012 system which we described in RoboDragons 2012 TDP [2], we improved the method of estimating the arrival time of the robot to the given destination. This paper shows it in detail. Moreover we propose an estimation method of the rotational speed of the spinning ball. It helps teams estimate the ball speed after bouncing when a chip-kick is taken. We hope the method will be build in the shared vision system.

2 Robot Hardware

In this section, we briefly describe our current robot. We show that the robot with/without cover in Figure 1. The features of our robot are shown as follows.

– 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.

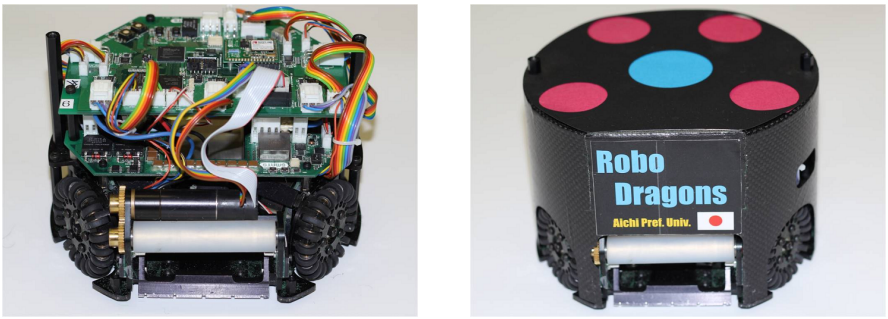


Fig. 1: Current robot developed in 2012
(Left: without cover, Right: with cover)

2.1 Components of the robot

All devices attached to the robot are shown in Fig. 2. The description of each device are presented in Table 1. The details are written in RoboDragons 2013 TDP [1].

2.2 Robot control program

The block diagram of the robot control program is shown in Figure 3. In the figure, each box named module is a thread program which run independently and other boxes are hardware which are controlled by modules. Basic control method is the same as the robot developed in 2010 [6].

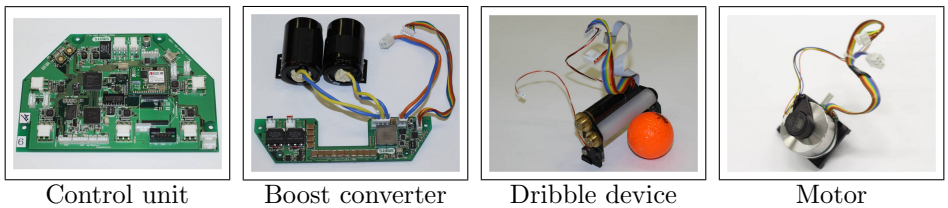
2.3 Configuration of communication packet

Thanks to the fast communication ability of the radio system, we redefined the communication packet configuration. The 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.

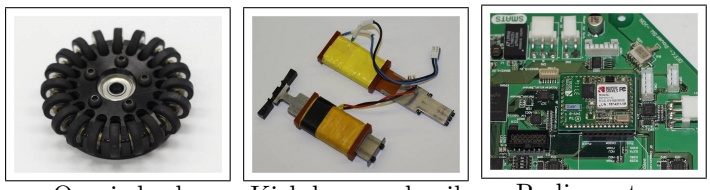
The 8 byte command is shown in Table 2. Basic idea of the command is that we give the moving vector and the angular velocity of the robot. In the 6th and 7th bytes, we give the kick command. “gggg” field selects kicker (straight/chip)

Table 1: Summary of the robot

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 Sparta-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 10 m/s velocity at maximum. Chip-kicker kicks a ball as far as 4 m distance at maximum.
Communication	IEEE 802.11g wireless LAN.



Control unit | Boost converter | Dribble device | Motor



Omniwheel | Kick bars and coil | Radio system

Fig. 2: All devices

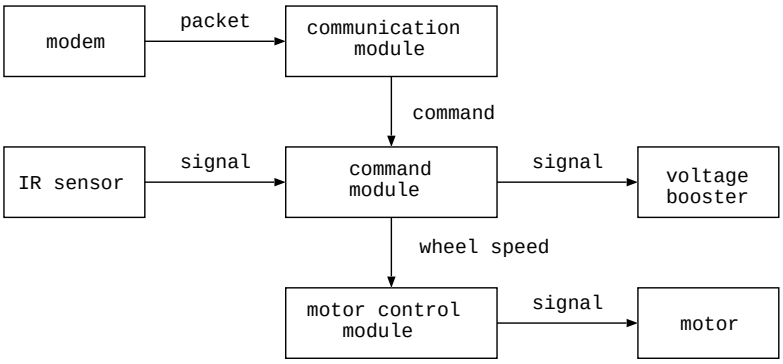


Fig. 3: Software configuration of robot

Table 2: Command for each robot

	Config.	Description
1st byte	aaaabbbb	aaaa: Robot ID, bbbb: Robot velocity
2nd byte	bbbbbbbb	bbbbbbbb: Robot velocity, 0 - 4095 (mm/s)
3rd byte	cccccccc	cccccccc: Moving direction, Resolution is $2\pi/512$ radian
4th byte	000cdeee	c: Moving direction, d: Rotation direction, 0:cw, 1:ccw eee: Angular velocity
5th byte	eeeeeeee	eeeeeeee: Angular velocity, 0 - 2047 (deg/s)
6th byte	ffffffff	ffffffff: Kick force, 256 levels
7th byte	ggggghhh	gggg: Normal/Forced kick, hhhh: Dribble velocity, 8 levels for each rotation direction (cw, ccw)
8th byte	iiiiiiii	iiiiiiii: CRC code

and kicking mean (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 4.

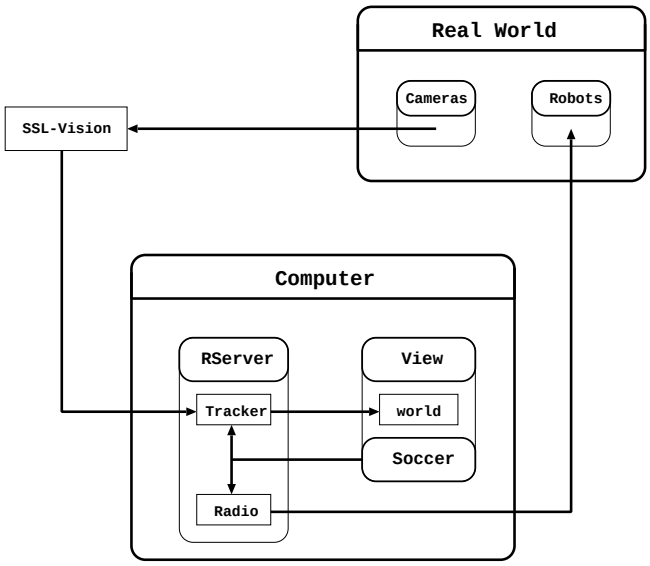


Fig. 4: Overview of software system

The host computer is a commercial one. CPU is Intel Core i7 4700MQ and main memory is 4 GB. OS is Ubuntu 13.10/Linux. Three main modules are running, each of which is composed as follows.

(1) The *Rserver* module receives SSL-Vision data and uses tracker submodule to predict the ball and robots positions by using Kalman Filter. They are stored in memory as world date, which are shared by viewed Soccer module. To send a command to each robot, a radio submodule is used.

(2) The *View* module is used to see the simulated image of real world so that they are easy to understand the situation. To do so, users set the number of robots and team color.

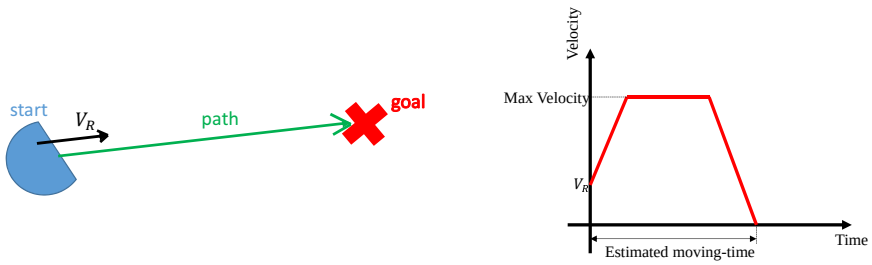
(3) The *Soccer* module makes an action command for each robot. By using the world data, the module chooses the best strategy, gives a role to each robot, and calculates a moving path for each robot.

3.2 Improvement of arrival time estimation

After the path planning for given destination, the estimation of moving-time to destination is necessary in many cases. Examples are the case finding the fastest robot to get to the destination, the case preventing an opponent robot from getting the ball, and so on.

We use the RRT algorithm [7] for path generation. However, in our system, the estimation of moving-time is quite simple, i.e. estimating it as the moving-time on the straight line connecting the current position and the destination under the predefined motion profile as shown in Figure 5.

In case that obstacles are in the field, as shown in Fig. 6, the path is not straight line. An actual path and a motion profile will be given by Figs. 6(a) and 6(b), respectively. Above approximation results in big error.

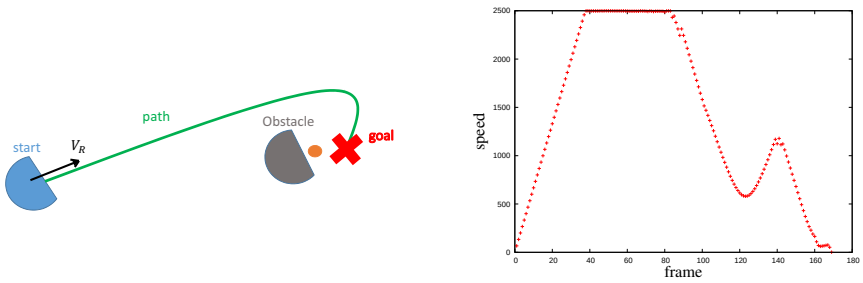


(a) A path for estimating moving-time (b) Motion profile

Fig. 5: Estimation of moving-time

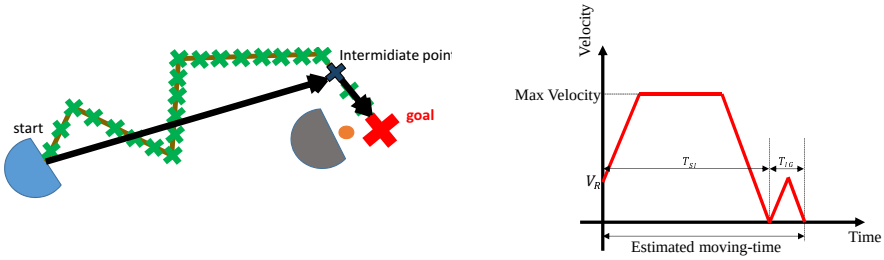
To reduce estimation error with keeping the real time computation, we adopted intermediate points method. The number of intermediate points being used depends on the situation. We use one intermediate point here. An example is shown in Figure 7. In the figure, a green line is a path generated by RRT. The intermediate point is given by the farthest point on the RRT line that the robot can move straight without colliding the obstacle. The estimated moving-time is given by the moving-time on the two straight black lines (current position - intermediate point and intermediate point - goal) under the motion profile shown in Fig. 7(b). We assume the velocity reduces to zero at the intermediate point.

With this estimation, we can reduce the approximation error by 51%.



(a) Actual path (b) Motion profile

Fig. 6: Actual path and motion profile



(a) Approximation by two lines (b) Motion profile

Fig. 7: Use intermediate point

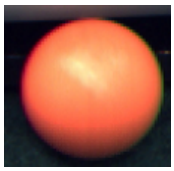
4 A Method to Estimate Ball's State of Spin by Image Processing

In RoboCup 2013, there were a few teams that a robot chip-kicked the ball with strong spin. When a ball is chip-kicked with strong spin, the spinned ball varies its speed after the ball bounced off the floor. This makes it difficult to predict the locus of the ball. If the ball's state of spin is obtained in advance, for example by image processing, it will be very useful to prevent disadvantageous situation [8].

4.1 Image processing target

In the RoboCup SSL, orange golf ball is used as an official ball. There are many dimples on the surface of the ball. These dimples cause the changes of reflection and this reflection changes according to the degree of blur. Figure 8 shows the differences of the images. Fig. 8(a) is a static ball's image and some highlight regions exist in it, on the other hand Fig. 8(b) is a spinning ball's image and it looks like a smoothed image. It will be possible to estimate the ball's state of spin by analyzing the distribution of reflection.

(a) Static ball



(b) Spinning ball

Fig. 8: Differences of the zoomed up images

4.2 Image processing method to estimate ball's state of spin

Our system is composed of three parts of image processing. First, the ball's region is extracted, then co-occurrence matrix and inertia feature are calculated in the ball's region, and finally the ball's state of spin is estimated.

The inertia feature Ine was calculated by

$$Ine = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (i-j)^2 P_\delta(i,j), \tag{1}$$

here $P_\delta(i,j)$ is a co-occurrence matrix, D_x and D_y are the differences of x- and y-coordinates of two pixels, respectively, and $\delta = (D_x, D_y)$ denotes the vector of them.

Figure 9 shows an example of the changes of the inertia feature for an image sequence in which the ball repeats spinning and stopping. (In Fig. 9, the rotational speed of the ball is constant.) It is known from the figure that Ine is changed largely between the ball's state is spinning and stopping.

Figure 10 shows that the relation between Ine value and rotational speed. It is shown from Fig. 10 that the value of inertia feature Ine decreases according to the increase of the rotational speed of the ball, so it will be possible to estimate the rotational speed of the ball. The spinning ball varies its speed after the ball bound off the floor. Thus if the rotation speed of the ball can be estimated, the trajectory of the ball after the bound off the floor will be predicted more accurately.

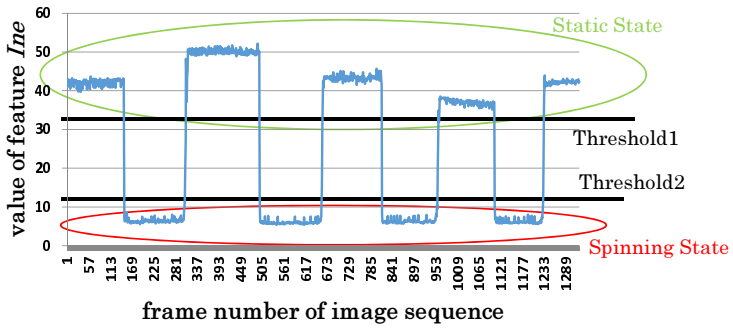


Fig. 9: The changes of the inertia feature for an image sequence

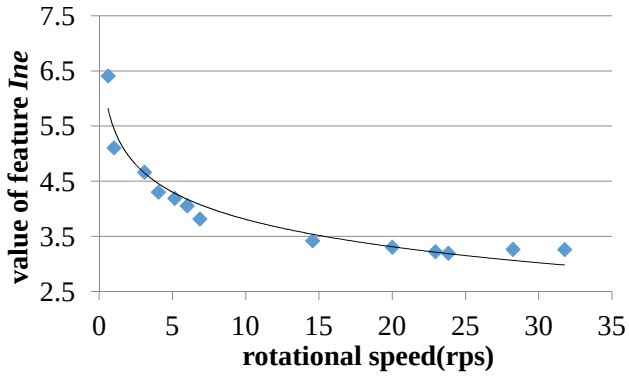


Fig. 10: Result of the image processing method to estimate ball's state

5 Conclusion

This paper described the summary of the robot and software system of RoboDragons 2014 and described an improved method of estimating the moving-time of the robot when a path is given. Moreover, this paper proposed a method to estimate the ball's state of spin. This method could be implemented in the shared vision system.

References

1. 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
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, Taro Inagaki, Hajime Sawaguchi, Yuji Nunome, Hiroaki Sasai, Yuki Tsunoda, Shinya Matsuoka, Naoto Kawajiri, Togo Sato, Kazuhito Murakami and Tadashi Naruse "RoboDragons 2012 Extended Team Description", RoboCup 2012 symposium CDROM, 2012
4. http://www.toppers.jp/en/index.html
5. http://en.wikipedia.org/wiki/TRON_project and http://en.wikipedia.org/wiki/ITRON
6. 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
7. 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
8. Yuji Nunome, Kazuhito Murakami, Kunikazu Kobayashi and Tadashi Naruse, "A Method to Estimate Ball's State of Spin by Image Processing for Strategic Learning in RoboCup Small-Size-robot League", "The Japanese Society for Artificial Intelligence", SIG-Challenge-B301-4, pp.21-25, 2013