The RoboCup-NAIST: A Cheap Multisensor-Based Mobile Robot with Visual Learning Capability

T. Nakamura* K. Terada H. Takeda A. Shibata

Nara Institute of Science and Technology Dept. of Information Systems
8916-5, Takayama-cho, Ikoma, Nara 630-0101, Japan
*E-mail:takayuki@is.aist-nara.ac.jp
URL: http://cairo.aist-nara.ac.jp/~ takayuki/robocup-naist.html

Abstract. Our contribution is composed of two parts: one is development of a cheap multisensor-based mobile robot, the other is development of robust visual tracking system with visual learning capability. To promote robotic soccer research, we need a low cost and portable robot with some sensors and a communication device. This paper describes how to construct a robot system which includes a lightweight and low cost mobile robot with visual, tactile sensors, TCP/IP communication device, and portable PC where Linux is running. In real world, robust color segmentation is a tough problem because color signals are very sensitive to the slight changes of lighting conditions. In order to keep visual tracking systems with color segmentation technique running in real environment, a learning method for acquiring models for image segmentation should be developed. In this paper, we also describe a visual learning method for color image segmentation and object tracking in dynamic environment. An example of the developed soccer robot system and preliminary experimental results are also shown.

Keywords: Multisensor-Based, Portable PC, Linux, Visual Learning, Color Image Segmentation and Tracking

1 Introduction

Robotic soccer is a new common task for artificial intelligence (AI) and robotics research[1, 2]. The robotic soccer provides a good testbed for evaluation of various theories, algorithms, and agent architectures. Through the research for accomplishing this task, a number of technical breakthroughs for AI and robotics are expected to be discovered. We focus on two points among RoboCup physical agent challenges [2]: one is **platform** and the other is **perception**.

So far, many researchers have been studying robotic soccer and have proposed a variety of theories and methods for controlling, planning and so on. They built a team of robotic platforms for playing soccer by themselves, or purchased robotic platforms (for example, [3]). There is no standard robotic platform design for robot soccer. Generally, contemporary robotic systems involve large amounts of expensive, special purpose hardware for motor control and image processing. In this paper, we describe how to construct a cheap multisensor-based mobile robot and its control system mainly made from a state-of-the-art portable PC, a battery-powered R/C model car, a CCD camera and a set of tactile sensors. Since recent portable PC is affordable and powerful, such a PC is used as a central controller which manages processing sensor information, controlling motor and communication between robots. As a chassis of the mobile robot, a 4-wheel drive R/C model car is utilized. The important feature of our robot is that this platform has all its essential capabilities on board. Our platform consists of driving, visual sensing, tactile sensing, motor control, communication and decision-making system. Since each system is made of devices commercially obtainable, we can reduce both of the cost and complexity of the system. According to our design principle for soccer robot system, those who are interested in the robotic soccer would easily utilize or build this robotic platform by themselves.

In real world, robust color segmentation is a tough problem because color signals are very sensitive to the slight changes of lighting conditions. Currently, human programmer adjusts parameters used in discriminating colored objects in response to the changes of surroundings. In order to keep visual tracking systems with color segmentation technique running in real environment, a learning method for acquiring models for image segmentation should be developed. In this paper, we apply a visual learning method to the problem of color image segmentation and object tracking in dynamic environment. To realize a visual learning, our method utilizes the competitive learning algorithm called *rival penalized competitive learning* (RPCL) [4] which can automatically find out the number of classes in the sample data that a perceived color image consists of. After this learning, our method uses discovered classes as color models for objects. Using this color models, a color image is segmented into several regions which correspond to some objects. Then, based on segmented regions, our method performs visual tracking.

To evaluate the developed system, we have implemented some behaviors for playing soccer and a visual learning method which can perform color image segmentation and object tracking. Preliminary experimental results are also shown.

2 Our Hardware Architecture

In order that our soccer robots are used by not only roboticists but also researchers in other research communities, our soccer robots should be manageable. Furthermore, in order that our robot system satisfies the requirements of a standard platforms, it is important to reduce the cost and time for building our robot system. To address this issue, we use a portable PC as a central controller of robot system which is recently affordable and powerful.

2.1 Driving System

As a chassis of the mobile robot, we utilize a 4-wheel drive R/C model car which is commercially available. Actually, we utilize a chassis of "BLACK BEAST" (NIKKOH [1]) (See **Fig. 1**). This chassis is composed of a PWS (Power Wheeled Steering) system with two independent motors. Because of this mechanical structure, our robot can rotate at the same place. This system is useful for avoiding the situation that its body gets stuck into corners. Existing motors provided by NIKKOH are comparatively powerful. However, if we put something whose weight is more than 1 Kg on the existing chassis, the body can't move around

[1] NIKKOH is a Japanese toy company. BLACK BEAST is also commercially available outside of Japan.

by those motors. In order to make the motor more powerful, a planetary gear box is attached to the existing motor. As a result, the chassis is able to carry something that weighs about 4 Kg. As the planetary gear box, we utilize the gear box [2] for a toy model car.

2.2 Tactile Sensing System

A tactile sensing system is used for detecting contact with the other objects such as a ball, teammates, opponents and a wall. It is also important for soccer robots to have tactile sensing capability, because soccer robots frequently collide with each other, walls or a ball in a soccer field. Furthermore, tactile sensing system can compensates for limitation of visual sensing. Since the field of view of the camera mounted on the robot is limited, if collision between the robots or between the robot and the wall or the ball occurs on the outside of the field of view, it is difficult to detect these happenings based on the image information. Tactile sensing system where tactile sensors are set around the body of soccer robot is very useful for solving this problem. Since the cost of producing a tactile sensing system is generally high, this prevents it being used widely.

Here, we construct a cheap tactile sensing system (See **Fig. 2**) by remodeling a keyboard which is usually used as an input device for PC. A keyboard consists of a set of tactile sensors each of which is a ON/OFF switch called a key. If a key is pressed, the switch is ON. If not, the switch is OFF. Since we can get a keyboard at a low price, it is possible to construct this tactile sensing system for soccer robots at a low cost.

If a tactile sensor (key) hits an object such as a ball or an opponent, the sensor outputs an ASCII code corresponding to the key. In case several sensors have contact with the other object, an output of this sensing system is a sequence of ASCII codes.

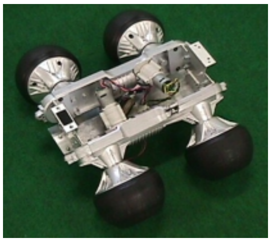


Fig. 1. Our driving system.

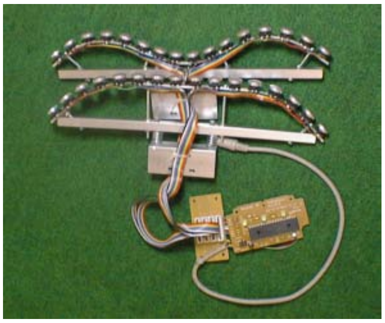


Fig. 2. Our tactile sensors made of a key board.

2.3 Visual Sensing System

Our robotic soccer project aims the development of robotic soccer players with on-board visual sensor like human soccer players. So, a visual sensing system in our soccer robot plays a fundamental role in acquiring visual information and recognizing it. Our soccer robots make a pass or tackle and shoot a ball into a goal based on the images taken by the on-board camera. In order to build such visual sensing system, we have chosen to use a commercial video capture

[2] The gear box is commercially available from a Japanese toy company TAMIYA

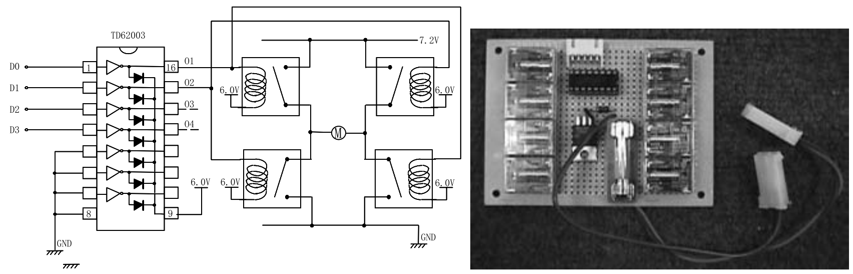
PCMCIA card (IBM Smart Capture Card II, hereafter SCCII) which can be easily plugged into a portable PC and a color CCD camera (SONY EVI D30, hereafter EVI-D30) which has a motorized pan-tilt unit.

SCCII is a PCMCIA type-II video capture card which can capture at 30 frame-per-second at maximum resolution 320-by-240 in 16-bit RGB formats. We can feed video to SCCII in NTSC or PAL format, and the card provides jacks for both composite-video and S-Video input. A device driver for the use of SCCII on Linux OS[5] is distributed as a free software. We utilize this device driver in order to capture images on Linux OS.

EVI-D30 is a high-performance color CCD camera, because it has auto target tracking function based on color information and motion detection function. We can control eyes of EVI-D30 with a motorized pan-tilt device which can be managed by a portable PC through RS232C. The pan and tilt angle of this device ranges from −100 to +100 and from −25 to +25, respectively. In this way, this camera can cover wide field of view. Since our soccer robot has such sensing capability, our robot can find a ball by moving its camera head without moving its body.

2.4 Motor Control System

A motor control system is used for driving two DC motors and is actually an interface board between a portable PC and motors on the chassis of our soccer robot (see **Fig. 3**). This control board is plugged into a parallel port on the portable PC. Our motor control system manages only the direction of current to a DC motor. The control circuit in this board consists of mainly 4 relays in terms of one motor (see **Fig. 3**). These relays are used as just like an ON/OFF switch and for controlling the direction of current. This board is powered by a 7.2 V battery for a R/C model car. As a result, this board can sends three control commands to right and left motors such as "(Forward, Stop, Backward)". The motor control command is actually 2 bits binary commands for one motor. Therefore, totally 4 bits (D0!&D1 or D2!&D3 in **Fig. 3**) in the parallel port are used for transmitting motor commands to the control board. Since we can send the motor control command to each of the two motors separately, our soccer robot has 3 sub-action primitives, forward, stop and backward in term of one motor. All together, our soccer robot can take 9 action primitives.



(a) (b)

Fig. 3. Our motor drive board.

2.5 Communication System

In the soccer game, teammates need to communicate each other for accomplishing a given task in cooperative manner. So, we set a wireless LAN device for communication on our soccer robot. The wireless LAN device is actually WaveLAN(AT&T) which can be plugged into a portable PC. The system operates in 2.4GHz frequency band. The rate of transmitting data is 2Mbps. The maximum transmission range will reach several hundred meters when there is a clear line of sight between the transmitter and receiver.

2.6 Intelligent Control System

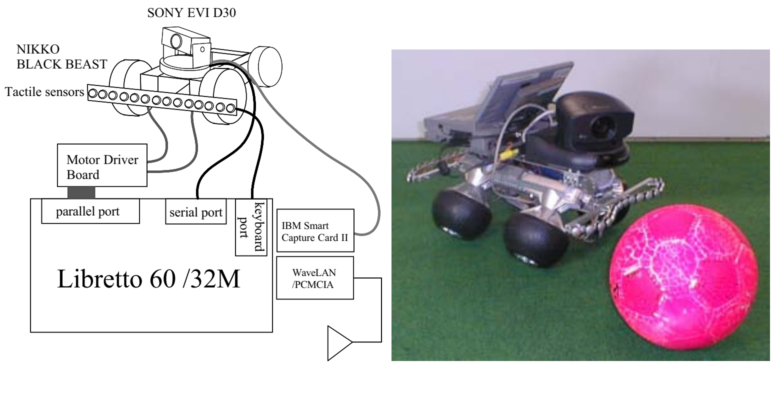
We call a central controller for processing sensor information and controlling the body of mobile robot and camera "intelligent control system". The intelligent control system consists of software, programming environment and OS. In order to adopt an OS as the central manager of robotic system, the OS should have some characteristics as follows:(1)It is possible to run multiple independent processes. (2)It is possible to make a process abort or wait for running again. (3)The system provides mechanisms for simple and high-speed process synchronization and communication.

In this work, we have chosen to use Linux OS as an OS for intelligent control system. Linux is a freely distributable, independent UNIX-like OS. Much of the software available for Linux is developed by the Free Software Foundation's GNU project. It supports a wide range of software, including X Windows, Emacs, TCP/IP networking (including SLIP/PPP/ISDN).

We cannot guarantee user-mode processes to have exact control of timing because of the multi-tasking nature of Linux. Our process might be scheduled out at any time for anything from about 10 milliseconds to a few seconds (on a system with very high load). However, for most applications in RoboCup competition so far, this does not seem to really matter. If we want more precise timing than normal user-mode processes, there is a special kernel RT-Linux that supports hard real time(See [6] for more information on this.).

3 System Configuration of Our Soccer Robot

Currently, we have developed a vision-based mobile robot for robotics soccer as shown in **Fig. 4**. As a portable PC, we have chosen to use a Libretto 60 (Toshiba) which is small and light-weight PC. The total cost of this soccer robot is about $ 4,800.



(a) (b)

Fig. 4. Our soccer robot.

4 Our Software Architecture

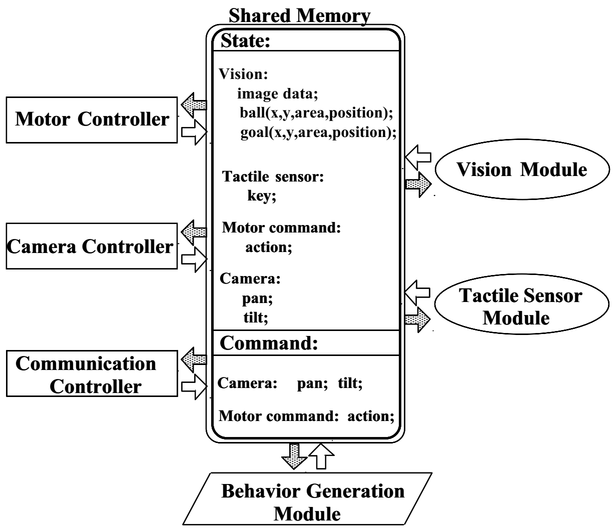


Fig. 5. Software architecture.

In order to control our hardware systems and coordinate between them, we use a shared memory [7] and 5 software components which are the motor controller, camera controller, tactile sensor module, vision module and behavior generator. **Fig. 5** shows an interactions between these software components. Note that this figure shows the software architecture of our current robotic soccer system. All software components read and write the same shared memory. Using this shared memory, they can communicates each other unsynchronously. As shown in **Fig. 5**, we define the structure of the shared memory. For example, the behavior generator takes the state of camera, vision, tactile and motor in the shared memory as input vectors. Then, it combines these information with programmer's knowledge and decides the robot's action at next time step. Finally, it writes the motor command for the motor controller on the shared memory. In the same way, other software components read states and write commands in each timing.

4.1 Motor Controller

We assume that a motor command is defined by a action primitive and its duration. In our robotic system, an action consists of a combination of 4 action primitives (move forward, backward, turn left, and turn right) and 4 kinds of the duration (100$msec$, 150$msec$, 200$msec$, 300$msec$). Furthermore, we add one action for kicking a ball strongly to the actions. This action is produced by a combination of "move forward" and 500$msec$ duration. Totally, our mobile robots can take 17 actions. Motor controller module reads the command from the shared memory every 100$msec$. If there is a command, it executes the command and rewrites the executed command as the state of motor.

4.2 Camera Controller

We can control the onboard camera (SONY EVI-D30) through RS232C with the VISCA protocol provided by Sony Corp. Using VISCA, we can control the pan, tilt angle and the focal length of the camera and take its focus. Furthermore, we can turn on and off the camera through this protocol. In robotic soccer task, panning the camera is important action for tracking the objects such as a ball, a goal, teammates and opponents. In order to detect where a ball is in the field, our robots always try to track a ball in their field of view using the motion of their camera head. Actually, pan and tilt angles are controlled so that the center of the ball image may coincide with the center of the captured image.

Since the soccer robots frequently lose the ball in the field, they must find the ball again as soon as possible. In order to realize the procedure for finding a ball, it is considered that panning the camera is very useful. We called such behavior of the camera "finding behavior". We implement the finding behavior as follows:

```
repeat
 Read the area of ball region from the shared memory.
 Make the camera pan.
 if the camera rotates to the limit,
     make the direction of its panning opposite.
until the ball is in view.
```

4.3 Tactile Sensor Module

Since our tactile sensor system is actually a keyboard, an output of the sensor system is an ASCII code corresponding to the key. We can get this ASCII code via X Event [8] which is a library function of X11 for detecting all events in X Window system. Our tactile sensor module maintains a table of ASCII codes and the configuration of tactile sensors. All tactile sensors are numbered from 1 to 32 so that the left front of tactile sensor unit might be numbered 1 and the right back 32. In case a sensor has contact with an object, the sensor module can detect which sensor has contact with the object using the table that shows corresponding between ASCII codes and the index number of tactile sensors. Then, the tactile sensor module rewrites the index number of the detected sensor as the state of the tactile sensor system on the shared memory.

4.4 Vision Module

The vision module provides some information about the ball and goal in the image.

To date, in RoboCup competition, each soccer robot tried to discriminate such objects based on color information. In our study, we also use color information for segmenting and tracking objects (a ball, goals, white lines, teammates and opponents). Furthermore, in order that such color image segmentation and object tracking should be correct even if surroundings such as lighting condition changes, our vision module has visual learning capability based on RPCL [4]. After the color segmentation, we calculate the coordinates of the center of ball and goal position, and the both maximum and minimum horizontal coordinates of the goal and so on. (See **Fig. 5**.) Then, based on segmented regions, our robots perform visual tracking. Our vision module also discriminates in which position center of ball or goal appears among three positions (right, center and left of an image).

Construction of Color Model for Segmentation In order to make initial color models for some obects, we use the competive learning alogirhtm called *rival penalized competitive learning* (RPCL) [4] which can automatically find out the number of classes in the sample data. After this learning, we use discovered classes as color models for objects. We briefly explain the procedure of RPCL according to [4]. RPCL algorithm repeats the following two steps until the prototype vectors converge on constant vectors.

STEP1: Randomly take a sample $\boldsymbol{x} = (x_1, x_2, \cdots, x_k)$ from a data set. Let $\boldsymbol{w}_i = (w_{i1}, w_{i2}, \cdots, w_{ik})$ be a prototype vector $(i = 1 \sim, N)$. Then, calculate a parameter u_i defined as follows:

$$u_i = \begin{cases} 1, & \text{if } i = c \text{ such that} \\ & \quad \gamma_c d(\boldsymbol{x}, \boldsymbol{w}_c) = \min_j \gamma_j d(\boldsymbol{x}, \boldsymbol{w}_j) \\ -1, & \text{if } i = r \text{ such that} \\ & \quad \gamma_r d(\boldsymbol{x}, \boldsymbol{w}_r) = \min_{j \neq c} \gamma_j d(\boldsymbol{x}, \boldsymbol{w}_j) \\ 0, & \text{otherwise.} \end{cases}$$

where $\gamma_j = n_j / \sum_{i=1}^{k} n_i$ and n_i is the cumulative number of the occurrences of $u_i = 1$. $d(\boldsymbol{x}, \boldsymbol{w})$ denotes a distance between $\boldsymbol{x}$ and $\boldsymbol{w}$. Generally, $d(\boldsymbol{x}, \boldsymbol{w}_i) = ||\boldsymbol{x} - \boldsymbol{w}_i||^2 = \sum_{j=1}^{k} |x_j - \boldsymbol{w}_{ij}|^2$. Moreover, $\boldsymbol{w}_c$ and $\boldsymbol{w}_r$ denote the winner vector which wins the competition for adapting to the input vector and the second winner vector called "rival", respectively.

STEP2:Update the prototype vector $\boldsymbol{w}_i$ by

$$\Delta \boldsymbol{w}_i = \begin{cases} \alpha_c(\boldsymbol{x} - \boldsymbol{w}_i) & \text{if} u_i = 1 \\ -\alpha_r(\boldsymbol{x} - \boldsymbol{w}_i) & \text{if} u_i = -1 \\ 0 & \text{otherwise.} \end{cases}$$

where $0 \leq \alpha_c, \alpha_r \leq 1$ are the learning rates for the winner and rival vector, respectively.

Fig. 6 shows examples of an image captured by SCCII and segmented image based on our method. Although the size of a captured image is 320×240 pixels, we shrink it so as to reduce computational cost of CPU on a portable PC. Actually, the size of a processed image is 80×60 pixels. As shown in **Fig. 6** (b), our color segmentation algorithm succeeds in extracting a red ball, a yellow goal, green field (ground), white line, and white wall. Currently, it takes about 230 $msec$ (about $4Hz$) for one cycle of this procedure in case $N = 57$.

Simple Color-Based Tracking Our simple tracking method is based on tracking regions with similar color information from frame to frame. We define a fitness function $\Phi_{target}(x, y)$ at a pixel (x, y) as a criterion for extracting a target region in the image,

$$\Phi_{target}(x, y) = \begin{cases} 1 \ \boldsymbol{C}(x, y) \in \boldsymbol{CM}_{target} \\ 0 \ \text{Otherwise} \end{cases}$$

,where $\boldsymbol{C}(x, y)$ and $\boldsymbol{CM}_{target}$ show a $Yr\theta$ value at (x, y) and a color model for a target represented by a cuboid, respectively. Based on $\Phi_{target}(x, y)$, the best estimate $(\hat{x}_{target}, \hat{y}_{target})$ for the target's location is calculated as follows:

$$\hat{x}_{target} = \frac{\sum_{(x_i, y_i) \in R} x_i \Phi_{target}(x_i, y_i)}{\sum_{(x_i, y_i) \in R} \Phi_{target}(x_i, y_i)}, \quad \hat{y}_{target} = \frac{\sum_{(x_i, y_i) \in R} y_i \Phi_{target}(x_i, y_i)}{\sum_{(x_i, y_i) \in R} \Phi_{target}(x_i, y_i)},$$

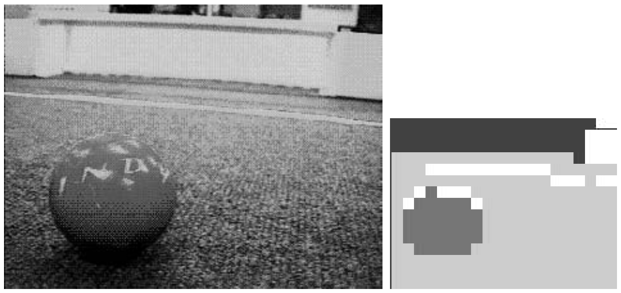

(a) (b)

Fig. 6. A example of processed images taken by the robots

where R shows the search area. Initially, R implies an entire image plane. After initial estimation for the location of the target, we can know the standard deviations $\sigma(\hat{x}_{target})$ and $\sigma(\hat{y}_{target})$ regarding $(\hat{x}_{target}, \hat{y}_{target})$. Therefore, based on the deviations, R is restricted to a local region during the tracking process as follows:

$$\begin{aligned} R : \{(x, y)| \\ \hat{x}_{target} - 2.5\sigma(\hat{x}_{target}) \leq x \leq \hat{x}_{target} + 2.5\sigma(\hat{x}_{target}), \\ \hat{y}_{target} - 2.5\sigma(\hat{y}_{target}) \leq y \leq \hat{y}_{target} + 2.5\sigma(\hat{y}_{target})\}. \end{aligned}$$

$\sum_{(x_i,y_i)\in R} \Phi_{target}(x_i, y_i)$ shows the area of the target in the image. Based on this value, we judge the appearance of the target. If this value is lower than the pre-defined threshold, the target is considered to be lost, then R is set to be the entire image plane for estimation at next time step. We set this threshold for the target area $= 0.05 * S$, where S shows the area of the entire image. This process helps to reduce the computational cost for extracting regions with similar color.

4.5 Behavior Generator

The behavior generator decides the robot's behavior such as avoiding a wall (called avoiding behavior), shooting a ball into a goal (called shooting behavior) and defending a goal from opponent's attack (called goalie behavior).

Avoiding Behavior We implemented avoiding behavior so that the robot may avoid a wall using tactile sensors. We divided 32 tactile sensors into 4 groups (G1,G2,G3,and G4);
$G1(1 \cdots 8)$: left front, $G2(9 \cdots 16)$: right front,
$G3(17 \cdots 24)$:left back, $G4(25 \cdots 32)$:light back
Avoiding behavior is implemented as follows:

```
Read the state of tactile sensor from the shared memory.
position ← the state.
switch(position)
    G1: move backward and turn right.
```

G2: move backward and turn left.
G3: move forward and turn right.
G4: move forward and turn left.

While the area of the ball region is less than a threshold, this behavior has top priority over all other behaviors. As a result, whenever the robot collides with an object in case that that condition is valid, it always avoids it. On the contrary, when the area of the ball is more that the threshold, this behavior is suppressed by the other behavior, for example, shooting behavior that is explained in the following.

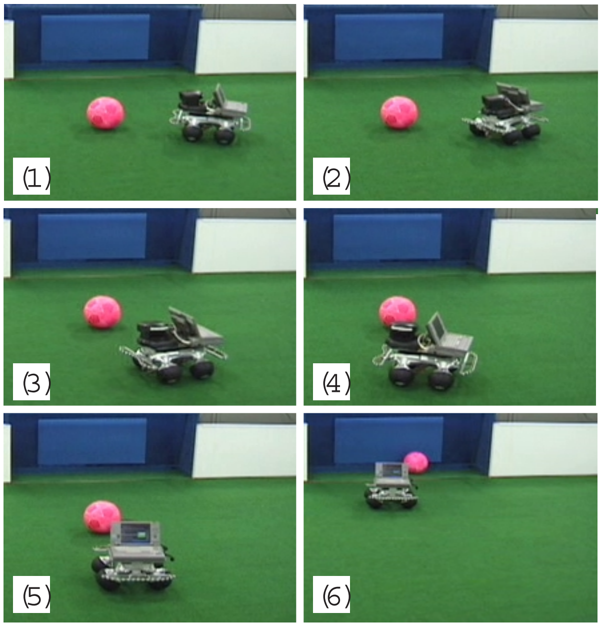


Fig. 7. Shooting behavior. (1):Approach the ball. (2),(3),(4):Round the ball. (5),(6):Kick the ball.

Shooting Behavior We make a simple strategy for shooting the ball into the goal. To shoot the ball to the goal, it is important that the robot can see both ball and goal. Therefore, the robot must round the ball until the robot can see both ball and goal with the camera toward the ball. Finally, the robot kicks the ball strongly. **Fig. 7** shows the shooting behavior.

The concrete procedure of shooting behavior is follows:

```
1)Find the ball
2)Approach the ball
    While approaching the ball
    Read the area of ball from the shared memory.
    if the area > 20 then stop
3)Round the ball
     Detect the direction of goal      d ← the direction
     switch(d)
        right: clockwise round the ball
```

```
            with the camera toward the ball
        left: counterclockwise round the ball
            with the camera toward the ball
    if the robot can see both ball and goal then stop
4)Make the body of the robot turn toward the ball
5)Kick the ball strongly
```

Here, we explain how to detect the direction of the goal which plays important role in shooting behavior. To find out the opponent's goal, the robot turns its camera head from -100 ° to +100 °. During the motion of the camera head, the robot continue to calculate the area of goal region in the image and find out the angle where the area of the goal is maximum. If the angle ranges in the right/left hand of the robot, the robot recognizes that the direction of the goal is right/left on the basis of the forward direction of the robot. In this way, the motion of the camera head enables the relative configuration between the robot and the goal to be detected.

Goalie Behavior We also make a simple strategy for preventing a ball from entering a goal. To defend the goal from opponent's attack, it is important that the robot always moves with the center of the robot body toward the ball. Therefore, the robot must coordinate the motion of camera head and its body so as to find a ball in the penalty area. When the ball is far from the robot, the robot turns its camera head to track a ball without the motion of its body. As long as the rotational angle of the camera head is within a certain range, the robot doesn't need to change its position to defend the goal. The reason is that the ball is almost in front of the robot under such situation. When the rotational angle for tracking a ball is over a certain range, the robot moves to the right/left side of penalty area in parallel with the goal line. In this way, according to the rotational angle of the camera head for tacking a ball, the goalie robot changes a strategy for defending the goal. This strategy comparatively worked well at the RoboCup-98 competition. **Fig. 8** shows an example sequence of the goalie behavior that is recorded at the RoboCup-98 competition.

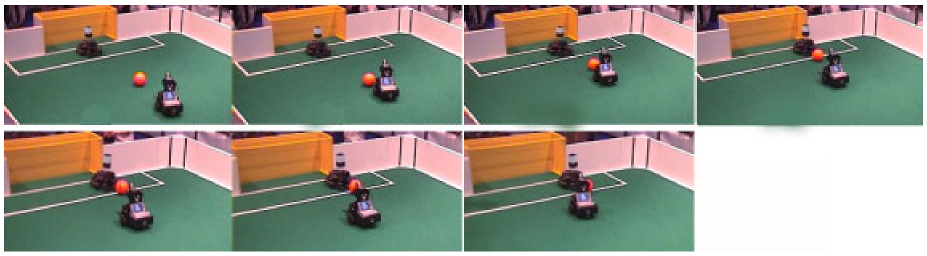


Fig. 8. Goalie behavior

5 Discussion

In this paper, we described how to construct a cheap multisensor-based mobile robot system which consists of mainly made from state-of-the-art portable PC, a battery-powered R/C model car, a CCD camera and a set of tactile sensors by remodeling a keyboard. Since these components are commercially available, we can construct the total system at comparative low cost. Our robot system might be used as a personal robot which can be used at home since its price would be low and its performance would be high. Now, we use this multisensor-based mobile robot as a standard platform for robotics soccer research. In the future, we plan to realize

- robust behavior based on sensor fusion between visual and tactile information, and
- cooperative behavior with other robots.

We also describe the software architecture of our robots and how to implement some kinds of behaviors for playing soccer. In respect of the implementation of soccer behavior, it is found that the motion of camera head on the robot gives cue for identifying the relative position between the robot and the goal or the ball. Currently, all soccer behaviors are programmed by the designers of the robot. This is a problem because these behaviors are not so adaptive to the unexpected change of the environment. Therefore, online learning method is required for adapting the robot to the unexpected events in its environment. Such learning method should operate on a mechanism that creates the adaptive behaviors in addition to the behavior programmed by the designer. In the future, we plan to realize

- online learning mechanism based on the evolution of the pre-programmed behaviors.

In this paper, we also describe a method for visual learning for color image segmentation on the basis of RPCL. Through the RoboCup-98 competition, it is found that the performance of the visual tracking based on our method works well. However, when the lighting condition of the environment suddenly changes, our robots have failed to track a ball. The reason is that the our current visual learning method is basically off-line. So, in order to make our visual tracking capability more robust, we should develop a mechanism of on-line visual learning for tracking the objects in the unexpected environment.

References

1. H. Kitano, M. Tambe, Peter Stone, and et.al. "The RoboCup Synthetic Agent Challenge 97". In *Proc. of The First International Workshop on RoboCup*, pages 45–50, 1997.
2. M. Asada, Y. Kuniyoshi, A. Drogoul, and et.al. "The RoboCup Physical Agent Challenge:Phase I(Draft)". In *Proc. of The First International Workshop on RoboCup*, pages 51–56, 1997.
3. Inc. Nomadic Technologies. `http://www.robots.com/robotdiv.html`.
4. L. Xu, A. Krzyzak, and E. Oja. "Rival Penalized Competitive Learning for Clustering Analysis, RBF Net, and Curve Detection". *IEEE Trans. on Neural Networks*, 4:4:636–649, 1993.
5. Linux. `http://www.linux.org`.
6. RT-Linux. `http://luz.cs.nmt.edu/~rtlinux`.
7. W. Richard Stevens. *UNIX NETWORK PROGRAMMING.* Prentice Hall, Inc., 1990.
8. Adrian Nye. *Xlib programming manual : for version X11 of the X Window System, 3rd ed.* O'Reilly, 1992.