The Kyushu United Team 2003 in the Four Legged Robot League

Yoshihiro Yasutake, Kentaro Oda, Toshiyuki Ishimura, Takeshi Ohashi

Dept. of Artificial Intelligence, Kyushu Institute of Technology
kato@mickey.ai.kyutech.ac.jp

Abstract. This paper presents our approach to the Sony Four Legged Robot League of the RoboCup 2003. Our system consists of inter-robot communication, meta planning, base-level planning, vision, localization, behavior and walking modules. We introduce several techniques for these modules: the images based cooperation, TSL color space for robust color classification, accurate localization through observing the nearest maker and enhanced parameterized walk. We also develop a set of powerful tools such as a Color Classification Tool, an Interactive Robot Control Environment and a Simulator. The color classification tool allows us to extract accurate thresholds fast. The interactive robot control environment allows us to control robot's behavior while it is running, and it also provides us functionality to observe robot's internal states. The simulator allows us to test a robot's program effectively in the ideal environment where specified conditions can easily be reproduced.

1 Team Development

1.1 Team members

[Kyushu Institute of Technology (KIT)]
Professors
Dr. Takeshi Ohashi,Dr. Takaichi Yoshida
Graduate Students
Yoshihiro Yasutake, Kentaro Oda, Toshiyuki Ishimura, Takeshi Kato,
Kazuteru Matsumoto, Hiroki Najima, Keiichi Kii,
Hideo Tateiba, Atsushi Hashimoto, Takayuki Kadowaki,
Kana Imamura, Masayuki Nitta
[Fukuoka Institute of Technology (FIT)]
Professors
Dr. Takushi Tanaka, Dr. Masaru Ishii, Dr. Hiroaki Shirakawa
Undergraduate Students
Sho Kawakami, Takahiro Kudo, Yoshitake Furuie

1.2 Team introduction

We, the highly involved members of the united team, are from Kyushu Institute of Technology (KIT) and Fukuoka Institute of Technology (FIT). KIT team is organized

into two groups. One is involved in reinforcement learning, image processing and has participated in the middle league in 1999 and 2000. The other consists of experts in distributed systems and object-oriented databases who involve applying their techniques for robotics field.

For more information, please visit our web page:
http://www.asura.ac/

The year 2001 four legged league of the RoboCup was our first participation, and our aim was to reach the world competitive level. As the year 2000 winner, the UNSW team had successful result due to their technical advances in locomotion, vision, localization and repertoire of behavior [1], our basic system was based on their successful sophisticated system. Since it was not sufficient, we introduced some techniques and tools. This year 2003 in RoboCup Japan open held at Niigata, we successfully got the first position.

The strategy of the year 2003 was to build collaboration among the robots, and fast development and tune-ups of strategies and behavior. So we introduced some new techniques and tools.

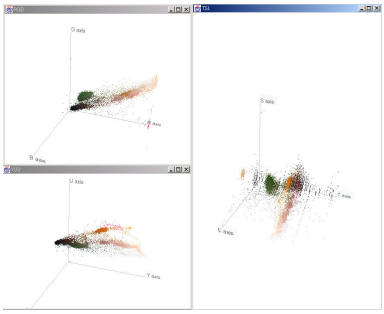

Fig. 1. Three-dimensional cumulative histograms of the soccer field pictures

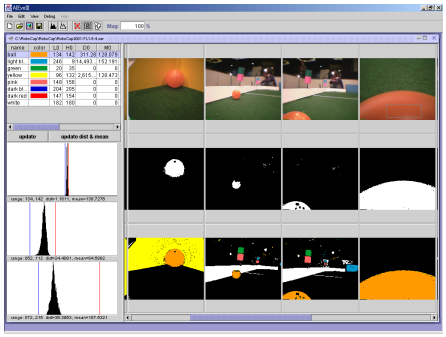

Fig. 2. The Color Classification Tool

2 Vision

The vision system must be robust and accurate since it is the main source of the environmental information. Its main task, the color classification is to classify each pixel into the eight colors which appear in the field, for example, orange for the ball, blue, pink and green for the markers, yellow and blue for the goals and dark blue and dark red for the robots. As far as we know in the four legged league, the color classification was achieved by the built-in hardware color lookup table or software approaches like UNSW [1]. Both of them are based on YUV color space, U and V represent color and Y is luminance, which is native color space in the NTSC standard (YIQ for exact). Since historically reasons, YUV color space is not suitable for classify the colors; it

was designed for keeping compatibility with monochrome televisions. Consequently, the luminance change seriously affects U and V components. To overcome this problem, we apply the TSL color space [2] for the color classification. T stands for tint, S for saturation and L for luminance. HSV or HSI spaces can be called similar color spaces in respect to T and H represent hue. TSL color space was originally designed to extract skin color from complex backgrounds. The advantage of TSL color model is that it can extract a given color robustly while minimizing luminance influence in compare with YUV, HSV or HSI color spaces. We can classify a given color reasonably accurate with only determining 6 parameters: minimum and maximum thresholds of T, S and L components (linear separation), whereas the UNSW team uses polygons to capture the colors on a given non-linear U-V plane (total up to 14 planes) with learning algorithm [1]. The dark blue color several teams reported difficult to separate, can be separated reasonably. Through minimizing the number of parameters, we can adapt the parameters easy and fast without complex tasks such as optimizing functions and learning algorithms. This advantage gives us the faster adaptation and manually tune-ups of the color table to a new lighting environment, for example, we will take about 15 minutes to whole setup including taking sample pictures. Figure 2 shows our color classification tool built with Java. Figure 1 demonstrates three-dimensional cumulative histograms of the soccer field pictures in RGB, YUV and TSL. In TSL color space, the colors are well clustered in the space. This tool will be available public, please visit http://www.asura.ac/.

3 Localization

Localization is the task to locate a robot itself in the field through observing 6 markers and goals. We employ stochastic gradient descent localization [1], which is able to locate the robot with one marker observation. However, we experienced significant amount of the error occurred when the robot could not see near markers. Because a far marker occupies tiny area in the image, the estimated distance of the robot to the far marker is unreliable. For example, 1 pixel height error in the pixel area of the furthest marker could lead about 20 cm errors in the estimated distance. Therefore, we introduced an observation strategy such that the robot try to see the nearest marker if higher accuracy is needed.

To achieve collaboration among the robots, the sharing information about positions of the ball and the teammates are important. For the collaboration, we introduce communication among the robots via the TCPGateway. Using this communication, our robots can share information about the ball, and the other robots in the field. Each of robot is keeping relative positions of any other robots and objects in the field, and each of which possesses relative position error. This position error will occur with the potision error of any objects possessed by another robot when sharing postision information among robots. We have to reduce confidence of the shared information accordingly. These confidences are calculated as follows:

$$cf = \frac{ocf \times rcf}{1000} \tag{1}$$

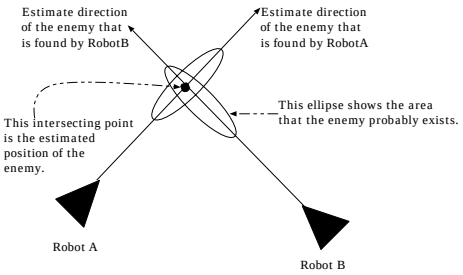

Fig. 3. Estimation about enemy's position

where cf is confidence of the shared information, ocf is confidence of the object's information when it was found, and rcf is the confidence of information about the robot that found the object.

When some robots can see the ball simultaneously, we can reduce position errors of the ball using weighted average method as follows:

$$bx = \frac{\sum (bcf_i \times rcf_i \times bx_i)}{\sum (bcf_i \times rcf_i)} \tag{2}$$

$$by = \frac{\sum (bcf_i \times rcf_i \times by_i)}{\sum (bcf_i \times rcf_i)} \tag{3}$$

$$bcf = \frac{\sum (bcf_i \times rcf_i)}{1000 \times num} \tag{4}$$

where bx, by and bcf are x/y-positions and confidence of the ball estimated by above mentioned method respectively, rcf_i is confidence of i-th robot that found the ball, bx_i, by_i and bcf_i are x/y-positions and confidence of the ball estimated by i-th robot respectively. The num is the number of robots that found the ball.

The position of an enemy are also important for strategies. However the position of the enemy cannot estimate by a single image, because distance information of the enemy is not reliable. If another robot can see the same enemy, the enemy's position can be estimated by combining the information of the other robot. Figure 3 shows how to estimate the enemy's position in that case. In this figure, RobotA and RobotB found the same enemy. Two ellipses show the area where the enemy is in, and two arrows show the estimated enemy's direction. The enemy's position can be estimated using the intersecting point of the lines that are drawn from the detector's position to the enemy's direction.

4 Locomotion

Highly sophisticated locomotion module parameterized walk[1] invented by UNSW allows us to control the robot with three degree of freedom; forward, left and rotation parameters. Despite of its power, we have to find out a set of offset parameters which

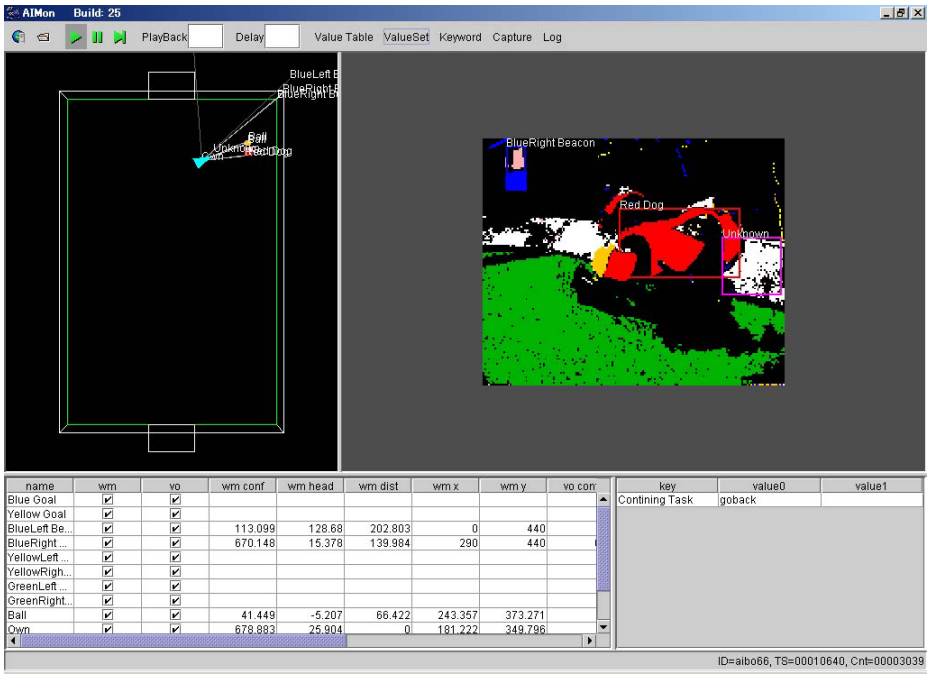

Fig. 4. The Monitoring Tool

defines the posture during a walk. Since posture seriously affects stability and speed, it has to be determined carefully. UNSW chooses a set of offsets according to the context of game; aggressive mode takes faster but unstable offsets for approaching the ball around where seems to be no enemy. Our approach is that according to a given three control parameters (forward, left and rotation), enhanced parameterized walk module chooses the most suited a set of offset parameters automatically so as to gain speed and stability. This approach guarantees the offsets to be used as optimal. This enhanced parameterized walk was used for goalie but not for attacker because the offset transition sometimes causes the robot unstable. As a future work, a smooth transition of the offsets and automatic offset extraction (through like learning algorithms) are necessary.

5 Interactive Robot Control Environment

In our observation, the most time-consuming tasks of the robots' programming in the four-legged league are the strategy programming, the calibration of the walk, the vision and the localization, and exploration of new behavior, gaits and postures. The strategy programming is very complex, and it includes the behavior tuning. These tasks tend to apply trial-and-error iteration intensively. By minimizing the iteration time, we can be able to find a better strategy, a better calibration and new behavior.

In order to minimize the iteration time, we introduce an Interactive Robot Control Environment. This environment consists of two tools: one is the interpreting language to control robot's behavior dynamically, and the other is a monitoring tool.

We introduce the scheme interpreting language, a small and clean lisp dialect, in order to control robot's behavior dynamically. According to the four-legged league rule in year 2002, wireless communication has been introduced. Then the collaboration among the robots become more important, and caused to become the strategy programming more difficult. In order to develop the strategy programs in an easy way, we choose the scheme interpreting language, as it possesses high abstraction power. The scheme interpreting language can be used for parameter calibration of walking and tuning behaviors, and postures. And also it can be used to write the strategies in a highly abstracted form. Despite of its abstraction power, we have to pay the penalty of GC (garbage collection) which gives interruption while computing the strategy. To overcome this problem, the strategy is partially written from the scheme language and the other part is written from the C++ language in order to gain appropriate power and efficiency of the computation. Using the scheme interpreting language, we can send a new program via wireless communication while a target robot is running. Then the robot changes its behavior on the fly. This is done by invoking a single keyboard short-cut on emacs editor. This greatly improves the efficiency of the development.

It is also important to observe robot's internal states such as the results of the color classification module, the results of the localization module and parameters that decide robot's behavior. In order to observe the robot's internal states, we develop a monitoring tool. Our monitoring tool observes the states as follows:

- The color-segmented image that the robot is seeing now.
- The localization results such as the estimated position of the robot, and the positions of the other objects in the field.
- Labels and values of the parameters that decide the next behavior. For example, the monitoring tool displays the current role of the strategy and current behavior.

The observation of the robot's internal status allows us to find the reason of the robots if they execute in an undesired behavior. The visualization of the color-classification results allows us to check thresholds easily. The robot sends these states to the monitoring tool via wireless communication. Figure 4 shows our monitoring tool build with Java.

6 Simulator

Robots' programming are difficult because of their complexity, continual changes of environments, limitation of resources and so on. Considering environment changes, for example, the changes of the lighting condition affect robot's behavior seriously. It is difficult to clear problems of the robot's strategies in the real environment because in each testing time sensory values such as camera images and the effectors will change. In order to clear the problem, it is needed to check strategies in exactly same environment. There are lack of resources such as robots and fields, so many developers have to share them. To maximize the efficiency of the development, it is needed to provide enough resources. Within the development cycle of the soccer robots, there are some overheads:

Fig. 5. The Simulator Server

copying programs into a memory-stick, and the start-up time of a robot. These overheads cause many difficulties for developing the robots' strategies. Then it is needed to minimize these overheads in order to maximize the efficiency of the development.

To provide the ideal environment that can be reproduced, to get rid of any limitations of resources, and to minimize the overheads, we develop a Simulator. The purpose of our simulator is to improve the efficiency of the development of vision-based robots' strategies. Therefore the simulator synthesizes virtual camera images and it simulates the robot's motion in three degree of freedom such as Forward, Left (cm/cycle) and Rotation (degree/cycle). The simulator provides 3D virtual soccer field environment, which accommodates multiple robot agents (virtual AIBOs) connected via TCP/IP. Figure 5 shows our simulator build with Java and Java3D.

The simulator consists of a simulation server and clients (the virtual robot). Figure 6 shows the simulator architecture. The simulator works as follows:

1. The server synthesizes agent's camera images of the virtual environment and sensory values such as tilt and pan.
2. Then the server sends the images and the sensory values to simulator clients.
3. The stub of each client receives them and passes to client's strategy program.
4. Then the strategy program decides next behavior, and sends back to the stub as a command of the Parameterized Walk.
5. The stub sends it to the server.
6. Then the server updates the position of the robot agent in the virtual environment.

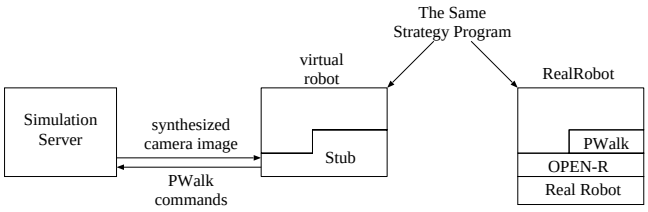

Fig. 6. Simulator Architecture

Using the stub, the same robot agent's strategy program is run in both the simulator environment and the real robot environment. The simulator allows us to develop the strategy programs without using the real robots. Then it allows us to reduce the overheads mentioned above. The server and the robot agents communicate via LISP-like strings. For example, a command of the Parameterized Walk is sent to the server as follows:

```
(P 3.0 0.0 0.0 40 0.0 0.0 0.0 8 0)
```

where `p` means a command of the Parameterized Walk, and rest of the parameters mean `forward`, `left`, `rotation`, `number of cycle`, `head's tilt`, `head's pan`, `types of walking style`, and `types of head motion`. Using LISP-like string, it is easy to extend the protocol between the server and the agents.

The simulator provides us some powerful functionalities. They are as follows:

- Initialization script language
- Communication among the agents
- Visualization of view frustum
- Message Filtering
- Sending pre-acquired images

These functionalities improve the efficiency of the development of the strategies greatly.

Initialization script language: The software robot agents can anytime join into the virtual environment provided by the simulation server. It is also possible to add balls into the environment anytime. The simulator provides a simple script language to specify initial positions of the objects such as the ball and the robots. This allows us to test the agent's strategy in the exactly same environment because the simulator can reproduce it repeatedly.

Communication among the agents: The simulator provides communication among the agents instead of the TCPGateway. With this communication functionality, it is possible to develop a strategy to collaborate the agents without using the real robots.

Visualization of view frustum: In the development of the soccer robots, enhancement of head's motions is important to find the ball efficiently and to accomplish accurate localization. The simulator provides functionality to visualize the view frustum that each of the agents is now seeing. This visualization is useful in order to tune-up head's motion.

Message Filtering: The simulator clients send commands of the Parameterized Walk to the server as messages. The Message Filtering functionality means that the server filters out undesired messages. For example, when we want to tune-up the robot's head motions, the server filters out all the messages received from the agent except the head motions. When we also want to examine an only one robot agent in the environment among the multiple robot agents, the server filters out all the messages except ones received from the target agent. The Message Filtering functionality allows us to choose the desired ones to be examined in the virtual environment.

Sending pre-acquired images: The simulator supports to send pre-acquired images of a real environment as camera images to a robot agent. These images are acquired from the monitoring tool might contain some noises. Using this functionality, we can evaluate the Noise-Elimination Functionality of the Vision Module without using the real robots. In fact, we enhance the noise elimination module to avoid blue/yellow goal noises appeared on the edges between the white lines and a green field.

7 Conclusion

With these techniques, we could get successful results, and became the best 8 position in the RoboCup 2002 competition game. These techniques improve the development efficiency and provide an infrastructure for collaboration among the robots. However, there're some problems left.

The first one relates finding thresholds of the color classification. We apply TSL color space for the color classification to minimize parameters. In fact, in a fluorescent-lighted environment, we can classify colors using only 6 parameters: minimum and maximum thresholds of T, S and L component. However, when a color temperature of the environment is low, we cannot adjust white-balance correctly. Hence it makes the color classification more difficult with only 6 parameters. We extended color classification module to accept a few set of thresholds as a single color. This means that the extraction of thresholds become more difficult. We are considering to apply learning algorithms to the color classification, or finding better color space that make extracting thresholds more easily.

The second one relates collaboration among the robots. To accomplish collaboration, we have to find more efficient behavior such as collision avoidance with other robots and passing the ball. The behavior for collaboration requires more accurate localization and information sharing. With wireless communication, our robots can share information such as the position of the ball. Position information of the objects such as the ball and the other robots, is based on the detector's position and it might contain position errors. As a result of sharing this information, the resultant information may

contain more errors. We apply weighted-average method to reduce these errors, and also apply estimation method of an enemy's position. However, we have to find methods to share strategic information such as role and intention.

We enhanced several modules originally developed by the UNSW team, and we would like to express our sincere thanks to the UNSW team efforts.

References

1. Bernhard Hengst, Darren Ibbotson, Son Bao Pham, John Dalgliesh, Mike Lawther, Phil Preston and Claude Sammut, The UNSW RoboCup 2000 Sony Legged League Team, in RoboCup 2000: Robot Soccer World Cup IV, Springer-Verlag, pp.64-75, 2000
2. Jean-Christophe Terrillon and Shigeru Akamatsu, Comparative Performance of Different Chrominance Spaces for Color Segmentation Detection of Human Faces in Complex Scene Images, in Proc. of Vision Interface'99, Canada, pp.180-187, 1999