Pseudo-local Vision System Using Ceiling Camera for Small Multi-robot Platforms

Yasuhiro Masutani, Yukihisa Tanaka, Tomoya Shigeta, and Fumio Miyazaki

Graduate School of Engineering Science, Osaka University
Toyonaka, Osaka 560-8531, Japan
soccer@robotics.me.es.osaka-u.ac.jp
http://robotics.me.es.osaka-u.ac.jp/OMNI/

Abstract. *Pseudo-local vision system*, which simulates visual information derived from an on-board camera of mobile robot based on a ceiling camera image, is proposed. It consists of a vision server and a client module which communicate with each other in the SoccerServer-like protocol. An image processing module for the on-board camera in the control program is replaced with this system. The simulated visual information is not a two-dimensional image data but a one-dimensional array which represents the nearest edge in each direction around the robot. However, it contains some of essential information of the on-board camera image. This concept was implemented for our robot system for the RoboCup Small-Size League. The server can transmit the edge data to 10 clients 30 times per 1 second. The time lag between grabbing image on the server and extracting visual information on the client is about 10[ms].

1 Introduction

Coordination of multiple autonomous robots with vision is an interesting research theme. However, autonomous robots equipped with an on-board camera tend to be large and require large experimental environment like the RoboCup Middle-Size League. On the other hand, we developed a small multi-robot platform equipped with an on-board camera and participated in the RoboCup Small-Size League. Since it is not easy to load all of the system on the robot for reasons of space, power, CPU capability, and so on, the video signal of the on-board camera is transmitted by radio link to PC outside the field and is processed on it (Fig. 1 [1]). Accordingly trouble in the video radio link (e.g. interference) is fatal for our system and we could not demonstrate the fruit of our research in the RoboCup2001 competition.

To avoid such a trouble, we propose a *pseudo-local vision system*, which generates the on-board camera information approximately from a ceiling camera image and has compatible function with an image processing module for the on-board camera. It is a client/server system. The server periodically sends visual information computed from the ceiling camera image to the clients which control the robots. Using the earlier version of this system, we participated in the RoboCup2002 competition (Fig. 2[3]).

D. Polani et al. (Eds.): RoboCup 2003, LNAI 3020, pp. 510–517, 2004.

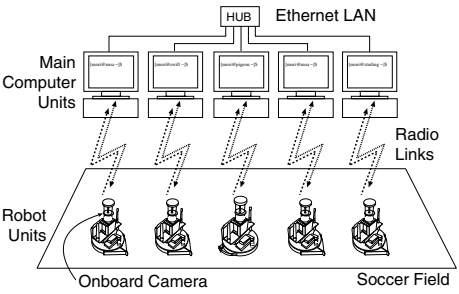

Fig. 1. Real local vision system

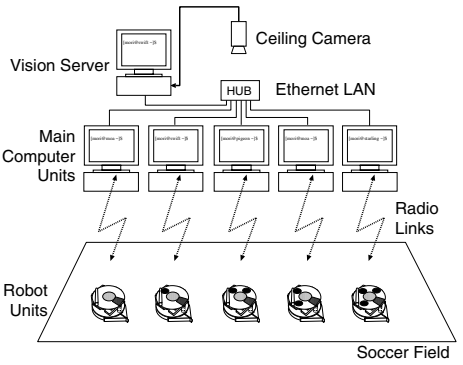

Fig. 2. Pseudo-local vision system

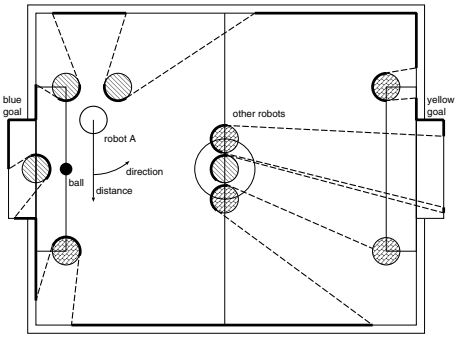

Fig. 3. Nearest edge

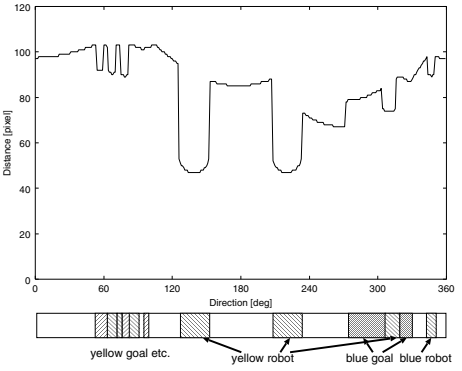

Fig. 4. Pseudo-local vision information

In such a system, there are some methods of generating the on-board camera information from the ceiling camera image. It is hard to make two-dimensional image data and to send it for 10 clients 30 times for 1 second. Therefore, in this paper, we propose a method of sending one-dimensional array data which represents the nearest edge on the floor in each direction around the robot. Fig. 3 and Fig. 4 show examples of the nearest edge of robot A and the one-dimensional array data respectively. We call this edge data *pseudo-local vision information*. It is not a real image (two-dimensional data), but contains some of essential information of the on-board camera image. For example, it represents a hidden object and overlapping objects. This system is useful as a research platform of autonomous mobile robot not limiting to the RoboCup.

This paper is organized as follows. In section 2 we discuss functions required for the pseudo-local vision system. Next the developed system is explained in section 3. In section 4 we evaluate the system. Finally conclusion and future works are described in section 5.

2 Discussion about Required Functions

2.1 Requirement

Let us consider generally requirements for the pseudo-local vision system. As a useful platform, we require that (P1) It can simulate image or alternative information derived from the on-board camera, (P2) Program code for on-board camera robot is almost available without change, (P3) Special equipments and devices are not necessary, and (P4) It can provide data for multiple robots 30 times per 1 second (same as the standard video rate). Moreover, as a tool for the RoboCup competition, we require that (R1) All teams can share the server impartially, (R2) It can serve 10 robots maximum with small time lag, and (R3) The protocol is easy to use.

2.2 Solutions

Since what is derived from the ceiling camera image is two-dimensional information on the floor plane, it is impossible to directly convert it into the on-board camera image of three-dimensional object on the floor. Thus, it is necessary to re-construct an image from the positional data derived from the ceiling camera and three-dimensional geometric models. Using recent high-performance processor, one can make simulated images with high fidelity at high speed. However, it is hard to make 10 images 30 times per 1 second.

If the server can output the simulated image as analog video signal, whole system including the frame grabber can be tested and our team does not need to change software at all. However, other teams which do not have on-board camera system need additional hardware to use the pseudo-local vision system. In case that the server transmits the simulated image as digital data, under condition, 8bit/pixel, 640×480pixel/frame, and 30frame/s without data compression for 10 clients, the necessary communication capacity is about 700[Mbps], which is too much for the standard Ethernet.

2.3 Proposed Method

According to our experience of participating in the RoboCup Small-Size League with robots equipped with omni-directional on-board camera, the most significant in the on-board camera information is the free space around the robot and the type of object beyond it in each direction. In such situations, a hidden object and overlapping objects are important problems. Therefore, we propose a method of using a one-dimensional array which represents the nearest edge on the floor around the robot in each direction as a simulated local vision information instead of the two-dimensional real image data. We call the array *pseudo-local vision information*. For example, in Fig. 3 bold lines show the nearest edge of robot A in each direction. Fig. 4 shows the contents of one-dimensional array of the nearest edge of robot A. The horizontal axis means the direction of $0 \sim 360$[deg]. The upper part shows the distribution of distance to the nearest edge. The lower part shows the distribution of type of the nearest edge.

3 Developed System

3.1 Structure

The developed pseudo-local vision system is a client/server system. It consists of the vision server and the client module which is a part of the robot control program (Fig. 5). Two parts are connected with network. Maximum of 10 clients can connect with the server considering that two teams use the server together in the RoboCup competition. The vision server derives position and orientation of all robots and ball from the ceiling camera image, computes the nearest edge data for each robot, and send it to each robot periodically. The client module communicates with server and converts the edge data into a form which can be used by the upper modules. The module replaces the image processing module for the on-board camera.

3.2 Protocol

The communication protocol is similar to that of the SoccerServer[5] on UDP/IP. The method of initializing is same as the SoccerServer. At first the client sends `(init)` message to the server and then receives visual information from the port specified by the server. The vision server accepts a team color (blue or yellow) and a robot number ($1 \sim 5$) in `(init)` message unlike the SoccerServer.

The visual information sent by the vision server consists of the time when the image is grabbed, the nearest edge array, and the ball data. The reason why the ball data is separated is that it is smaller than other objects. The format of the visual information is shown as follows.

`(see` *Time* `((edge)` *EdgeSstring*`) ((ball)` *Distance Direction Size*`))`

Time	Time when the image is grabbed.
EdgeString	Character string which represents the nearest edge data. The edge data in one direction is represented by three characters. The first character means the type of edge. The second and third characters means the distance of the edge on the image plane in hexadecimal notation [pixel]. The method of representing the type of edge for robot has three options as follows. – The edge types of all robots are same. – The team color is distinguishable by the edge type. – The robot number is distinguishable by the edge type.
Distance	Distance to the ball on the image plane [pixel]. In case that the ball is hidden, `-1` is set.
Direction	Direction of the ball [deg].
Size	Size of the ball on the image plane in angle [deg].

An example of the visual information is shown as follows.

```
(see 20306.148 ((edge) r59r5ar5av3e··· ) ((ball) 61 121 3))
```

EdgeString contains 1080 characters in our current version.

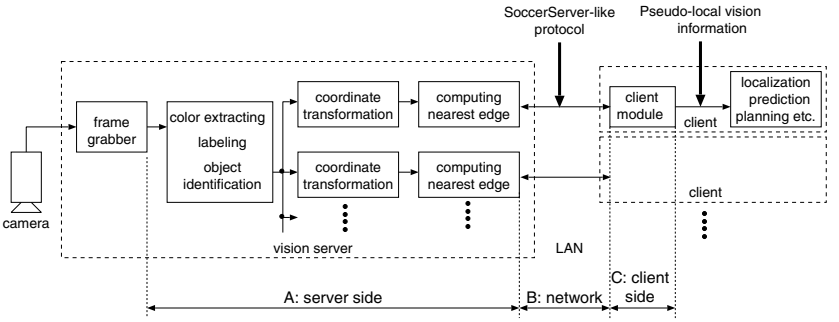

Fig. 5. Structure of the pseudo-local vision system

3.3 Vision Server

The vision server extracts color markers on the robots from the ceiling camera image, distinguishes them, and derives their positions and orientations. It also measures the position of ball. A inexpensive frame grabber is employed to digitize video signal. Modified CMVision [4] is embedded in our program to obtain color region in the image. Moreover, Kalman filter is introduced to improve the quality of information.

After that the processes are carried out for each client separately. At first the coordinates of all objects are transformed into the processed robot's coordinate system. In this system the nearest edge is found from intersection points between the object boundary and line through the origin. The boundary of robot and ball is modeled as a circle. Next the distance of the nearest edge is converted into the distance on the image plane in integer.

In the current implementation, the view angle is 360[deg] (omni-directional), the direction is quantized every 1[deg], the conversion of the distance on the floor into the distance on the image plane is based on the model of omni-directional hyperboloidal mirror used for our on-board camera. The conversion model is the following form.

$$R = \mathrm{int}(f^{-1}(r)), \qquad f(R) = \frac{R}{A - B\sqrt{C + R^2}} \tag{1}$$

where r is the distance on the floor plane from the robot center, R is the distance on the image plane from the image center, A, B, and, C are constants, $\mathrm{int}(x)$ is function to make x integer. In the current version all they are fixed. However, in the future version, they will be variable.

The above mentioned processes are carried out for maximum of 10 clients. After computing of the nearest edge for all clients, they are sent to the clients in random order in the format specified by the protocol.

3.4 Processes on Client Side

The client module replaces the on-board camera information with the information made from the edge data sent by the server. In our program for robot

control, the multi-thread library is employed, the lowest image processing (grabbing image frame, extracting color, labeling, and AND operation with template) is assigned to one thread. Thus this thread is replaced with a new thread with the client module. The other threads need no change. Based on output of the substituted thread, the upper threads can execute detecting free space around the robot, self-localization, estimation of the ball position, and so on in the same manner as on-board camera case [2].

3.5 Clock Synchronization between Server and Client

In order to use image of a moving object, it is indispensable to precisely detect the time when the image is grabbed. Since the vision server which grabs image and the client module which uses visual information are running on different machines generally, clock synchronization between them is necessary. We found that a method of using the system clock of Linux and NTP (Network Time Protocol) does not give enough precision for this purpose. Therefore, we developed a method of time measurement and clock synchronization using RDTSC (ReaD Time Stamp Counter) of Pentium processor referring to Bernstein's clockspeed[6]. As a result, we can measure time among multiple machines to a precision of under one millisecond. Using this, time lag in the system is evaluated, control method of considering time lag is carried out.

4 Evaluation

The above mentioned system is implemented on Linux. In this experiment for evaluation, the server PC equipped with Pentium IV 2[GHz] grabs the ceiling camera image with frame grabber using BTB878 in 640×480 [pixel]. 10 client programs are connected with the server. Some clients are executed on the same machine. The performance of PC for clients is various. The client whose performance is measured is executed alone on a machine with Pentium III 500[MHz]. The server PC and client PCs are connected with 100Base-TX.

4.1 Example of Vision Processing

Fig. 6 is a simulated on-board camera image of robot A in Fig. 3, which is made from the nearest edge array shown in Fig. 4 by drawing it on omni-directional image plane. This robot is also equipped with an omni-directional camera. An image taken with the camera at the same point is shown in Fig. 7. Comparing the two images, there is no information of the part beyond the nearest edge in Fig. 6. However, except for this point, the nearest edge data well simulates the real image. Note that Fig. 6 is not used in the control program but is help to human to understand the situation.

4.2 Processing Period and Time Lag

The server can accept 10 clients and has capability of processing image and sending information simultaneously every 33[ms] under the above mentioned

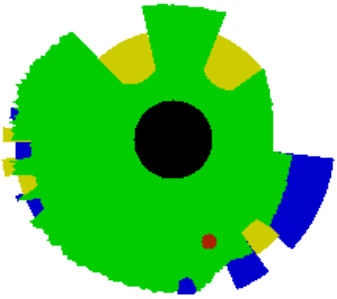

Fig. 6. Pseudo-local vision image

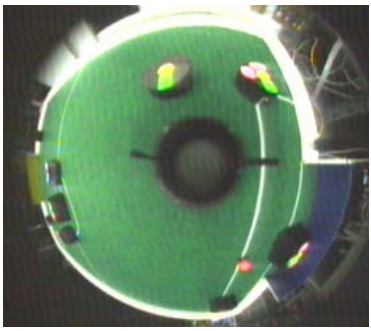

Fig. 7. On-board camera image

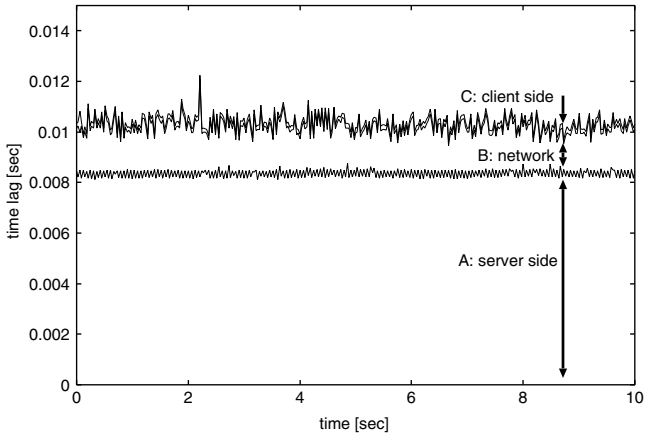

Fig. 8. Time lag in vision processing

condition. Fig. 8 shows the trend of time lag between grabbing image on the server and extracting visual information on the client. A, B, and C in the figure mean the time lags on server side, network, and client side respectively (see also Fig. 5). The time lag is almost constant, the total is about 10[ms]. Time lag from time when the shutter is released is added 33[ms] to it, which is spent in grabbing image.

In case that our robot control program for the on-board camera is executed on PC with Pentium III 500[MHz], the time lag between grabbing the image in 320 × 240[pixel] and extracting information is about 23[ms]. Introducing the high-performance PC for the vision server, the time lag of the pseudo-local vision is smaller than that of the real local vision, even if the server works for 10 clients.

4.3 Resolution and Hidden Ball

In case that the ceiling camera image is grabbed in 640×480 [pixel], the resolution is almost uniform all over the field, one pixel corresponds to 5×5[mm] on the

field. On the other hand, in case that the image of our omni-directional on-board camera is grabbed in 320×240[pixel], one pixel corresponds to 4.4×4.4[mm] on the field at the nearest and available point, almost same as the ceiling camera case at point 220[mm] far from the robot, worse than it at farther point. However, it is only an example. If the view angle of the on-board camera is narrower than 360[deg] and/or the resolution is higher, the current version cannot simulate in enough precision. Moreover, there is a problem that the robots sometimes hide the ball from the the ceiling camera. It means that the system cannot completely simulate a property of the on-board camera that a nearer object is easier to observe. To overcome these problems, multiple ceiling cameras will be introduced in the future work.

5 Conclusion

In this paper we propose the practical method of simulating the on-board camera information using the ceiling camera for multiple mobile robot platform. Although the simulated visual information is simple, it represents some of essential problem of the on-board camera information. Evaluating the developed system, we found that the processing period and the time lag are small enough for the system to be used practically. Actually we participated in the RoboCup2002 using the earlier version of proposed system.

In the near future, we will open the source code of the vision server to the public on the Internet. By using this framework we may be able to realize "the Pseudo-Local Vision League". Moreover it will be a bridge between the simulation league and the real robot leagues in the RoboCup.

References

1. D. Sekimori et al.: The Team Description of the Team OMNI, RoboCup 2001: Robot Soccer World Cup V, Springer (2002)
2. D. Sekimori et al: High-Speed Obstacle Avoidance and Self-Localization for Mobile Robots Based on Omni-Directional Imaging of Floor Region, RoboCup 2001: Robot Soccer World Cup V, Springer (2002)
3. Y. Masutani et al.: Team OMNI in 2002 — Pseudo Local Vision Approach —, RoboCup 2002: Robot Soccer World Cup VI, Springer (2003)
4. J. Bruce et al.: Fast and Inexpensive Color Image Segmentation for Interactive Robots, Proceedings of 2000 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2000)
5. SoccerServer, `http://sserver.sourceforge.net/`
6. D.J.Bernstein: clockspeed, `http://cr.yp.to/clockspeed.html`