Development of a Simulator of Environment and Measurement for Autonomous Mobile Robots Considering Camera Characteristics

Kazunori Asanuma[1], Kazunori Umeda[1], Ryuichi Ueda[2], and Tamio Arai[2]

[1] Chuo University, 1-13-27 Kasuga, Bunkyo-ku, Tokyo 112-8551, Japan
{asanuma, umeda}@sensor.mech.chuo-u.ac.jp
http://www.mech.chuo-u.ac.jp/umedalab/
[2] The University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
{ueda, arai}@prince.pe.u-tokyo.ac.jp
http://www.arai.pe.u-tokyo.ac.jp/

Abstract. In this paper, a simulator of environment and measurement that considers camera characteristics is developed mainly for RoboCup four legged robot league. The simulator introduces server/client system, and realizes separation of each robot's information, introduction of each robot's difference and distribution of processes. For producing virtual images, the simulator utilizes OpenGL and considers the effects of blur by lens aberration and so on, random noise on each pixel, lens distortion and delayed exposure for each line of CMOS device. Some experiments show that the simulator imitates the real environment well, and is a useful tool for developing algorithms effectively for real robots.

Key words: Simulator of Environment and Measurement, Camera Model, Modeling of Measurement Error, Server/Client System

1 Introduction

So as to develop robots or a multiple robots' systems that work in real, dynamic environment, it is necessary to assume a lot of conditions and test the robots or robots' systems for the conditions. However, this requires much cost by much trials and errors. To reduce the cost, simulation is effective. Many kinds of simulators have been developed for various applications, e.g., teleoperation[2], autonomous mobile robot[3]. It is also the case in RoboCup and simulators have been developed [4, 5]. They simulate kinematics or motions, but sensing is not well considered that suffers noises, distortions, etc.

In this paper, we develop a simulator for multiple mobile robots that have the ability to produce images by considering explicitly the characteristics of a camera, and the ability to simulate the real robots' behaviors. The concrete target as the multiple mobile robots system is the RoboCup four legged robot league.

In developing the simulator, we focus on the following features:

– programs for real robots can be applied directly,
– multiple robots can share environment and interact with each other,
– images obtained by a real robot can be imitated well.

2 Outline of the Simulator

In the field, a lot of information are obtained from the ball, the landmarks, the robots, etc., and multiple tasks are executed simultaneously in real robots. The simulator is designed so that it produces virtual images precisely for each robot with arbitrary position and orientation, and that the following tasks can be executed virtually in the simulator. Consequently, debugging of programs and verification of algorithms can be effectively performed virtually on PC. Fig.1 illustrates the relation between a real robot and the simulator.

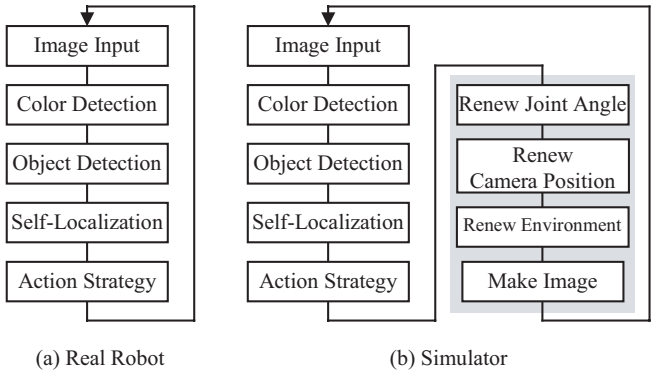


Fig. 1. Tasks in image input cycle of a real robot and of the simulator

Objects in the virtual field that is produced on the simulator are modeled with precise size by using OpenGL[7](see Fig.2.). Each robot on the virtual field produces images by evaluating its camera coordinate that is calculated using the position and orientation of the robot, the elbow angles of its legs, the pan/tilt angle of its neck(see Fig.3(a).). By considering the characteristics of its CMOS camera, the simulator can produce images that imitate real images precisely(see Fig.3(b).). Each robot applies a method of color detection for the synthesized images, and detects 8 kinds of color regions(see Fig.3(c).).

3 Server/Client System

So as to realize the requirement that multiple robots can share environment and interact with each other, server/client system is introduced in the simulator; tasks are performed cooperatively by two kinds of processes, i.e., a server process

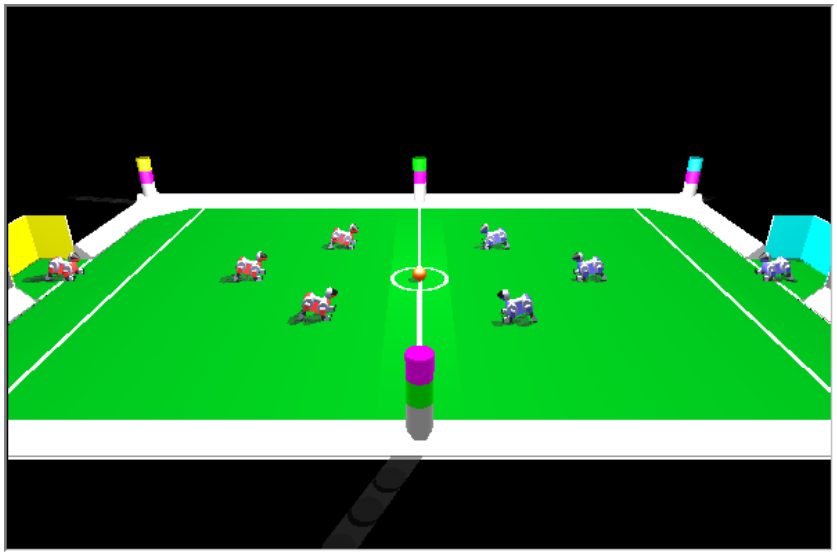
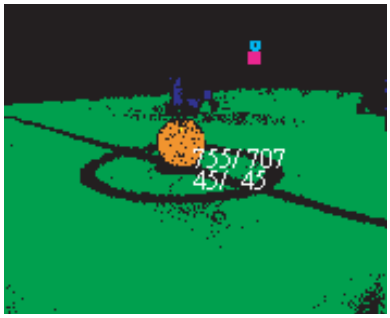


Fig. 2. Virtual field

(a) Original OpenGL image

(b) With CMOS filter



(c) Color extraction for (b)

Fig. 3. Virtual image

and client processes. The server process administrates the environment of the field totally, and the client processes execute robots' behaviors. By introducing the server/client system,

1. separation of each robot's information,
2. realization of each robot's difference,
3. distribution of processes

are realized. By 1., the condition is satisfied that each robot cannot acquire other robots' information explicitly without communication. By 2., robots' behaviors with different algorithms for each team can be implemented and a new algorithm can be easily tested. By 3., each process can be run on a different PC by using TCP/IP protocols, and calculation cost for each PC can be reduced.

Fig. 4 illustrates the structure of the server/client system. Production of a virtual image, analysis of the image, active sensing and simulation of some behavior strategy are performed by clients. Each client program is assigned to the control of each robot. By executing multiple client programs simultaneously and connecting them to the same server, multiple robots in the same field are realized. At the same time, the server administrates the environment on the field; it administrates the connected clients and the ball, recognizes the collisions, etc. Each client can recognize the information on the field through the server. Information is communicated at a constant period among the server and the clients so as to adjust the field environment.

4 Consideration of Camera Characteristics

The images are generated by OpenGL assuming an ideal pinhole camera(see Fig.3(a)), and thus they are different from real images obtained by the CMOS camera on the real robots. Therefore, we consider the camera characteristics of the CMOS camera so as to produce images that closely simulate the real images. The parameters of the ERS-2100's camera are as follows.

- CMOS 1/6inch, 178×144 pixels
- Lens F2.0, f=2.18mm
- Angle of View H:57.6deg, V:47.8deg
- White Balance: 4300K
- Frame Rate: 25fps
- Shatter Speed: 1/200sec

We consider the following characteristics.

- blur by lens aberration, focusing error, etc.
- random noise on each pixel
- lens distortion
- delayed exposure for each line of CMOS device

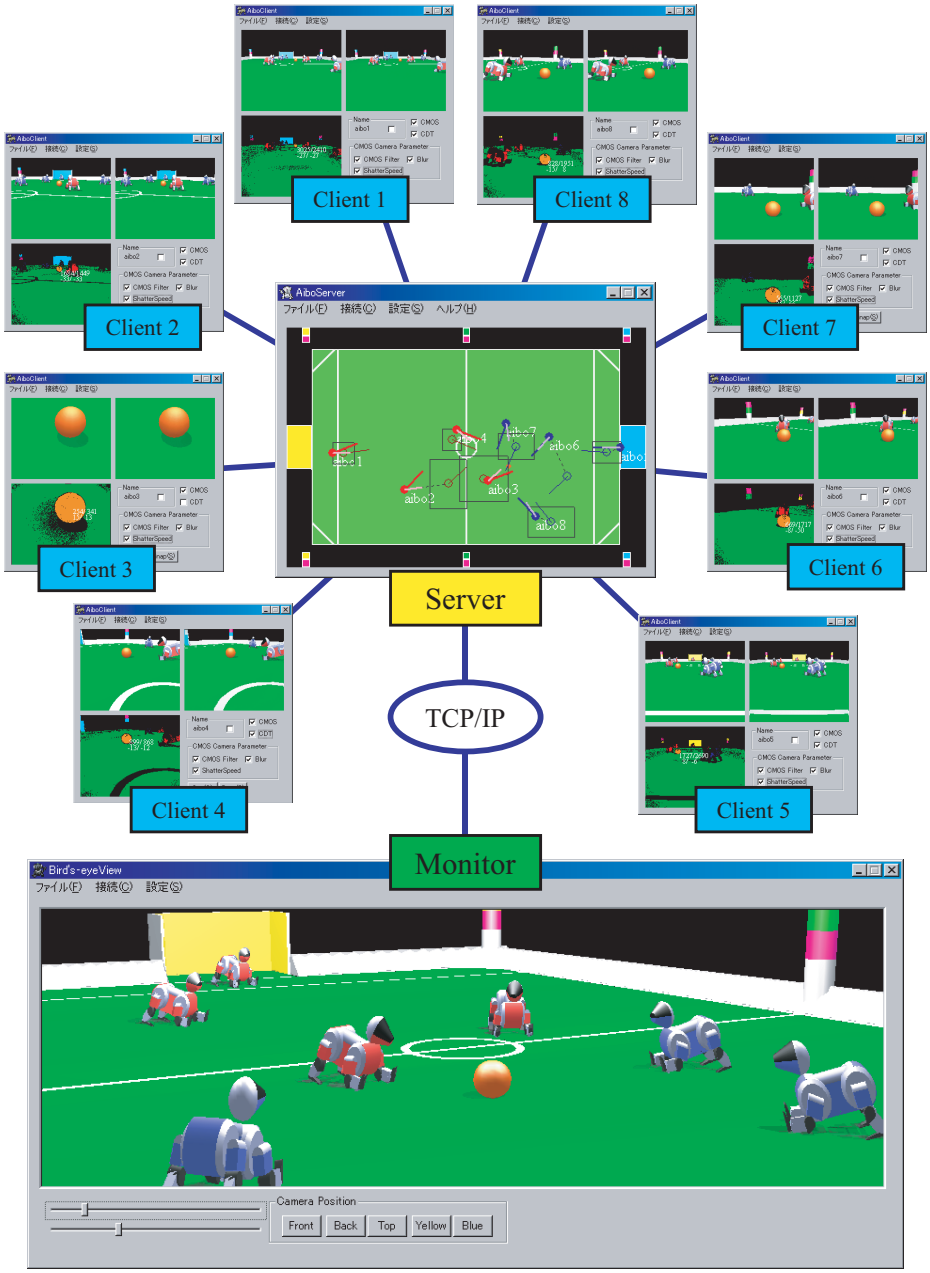


Fig. 4. Server/client system

4.1 Blur by Lens Aberration, etc.

Blur is generated by various lens aberration, focusing error, etc. Supposing the blur of each pixel is approximated by Gaussian distribution, we apply Gaussian filter to the virtual image. For an image $f(u, v)$, two dimensional Gaussian distribution function

$$G(u, v) = \frac{1}{2\pi\sigma^2} \exp(-\frac{u^2 + v^2}{2\sigma}) \tag{1}$$

is convoluted, and the blurred image is obtained. In the simulator, the area of convolution is set to 5×5, and σ is set to 1.

Fig.5(a) shows an image by the real camera for the ball with the distance of 1m. Fig.5(b) is the virtual image generated by the simulator on the same condition for Fig.5(a). Fig.5(c),(d) are the close-ups of Fig.5(a),(b). Fig.5(e) is the image generated from Fig.5(d) by applying Gaussian filter. It is shown that Fig.5(e) is closer to the real image, and additionally, aliasing that appears in Fig.5(d) is not observed.

4.2 Random Noise on Each Pixel

Random noise is inevitable, and it is remarkable in principle for CMOS device. The simulator adds the random noise on each pixel with the variance that is evaluated a priori by experiments. Fig.5(f) shows the image generated from Fig.5(d) by adding random noise. Additionally, we consider limb darkening generated by the lens as shown in Fig.5(g). Parameters of limb darkening can be evaluated a priori by experiments.

4.3 Lens Distortion

Images are distorted by lens distortion. As the model of lens distortion, we apply the equation approximating a radial distortion:

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix} = \frac{2}{1 + \sqrt{1 - 4\kappa(u^2 + v^2)}} \begin{bmatrix} u \\ v \end{bmatrix} \tag{2}$$

to the virtual image, where (u, v) is the coordinates on an original image and $(\tilde{u}, \tilde{v})$ is that on the distorted image. κ in eq.(2) can be evaluated by camera calibration. Fig.6 shows an example of the simulation of lens distortion.

4.4 Construction of Virtual CMOS Filter

In order to make the calculation time shorter, we integrate the multiple processes for camera characteristics into one filter, called CMOS filter. By using this filter, the cycle time of producing virtual images became shorter from 208[msec] to 112[msec] by PentiumIII 600MHz. Fig.7 shows the final image by applying the CMOS filter.

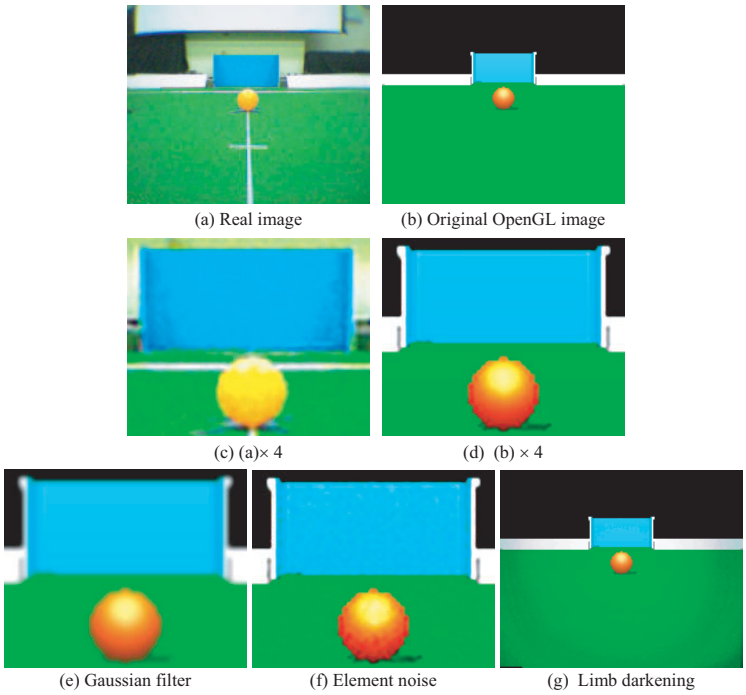


Fig. 5. Consideration of camera characteristics

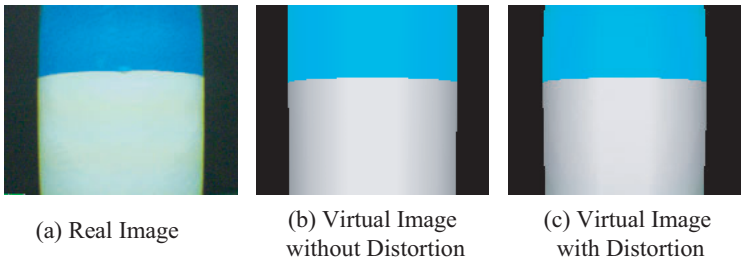


Fig. 6. Lens distortion

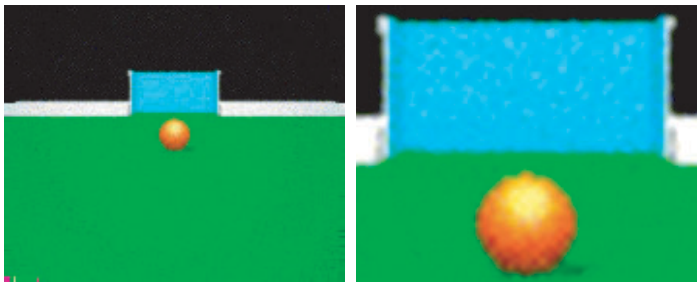


Fig. 7. CMOS filter: total consideration of camera characteristics

4.5 Delayed Exposure for Each Line

For motions of the robots, two more camera characteristics have to be considered. One is the motion blur. This is regardless of the kind of the camera, and depends on the exposure time. The other is specific to CMOS cameras as the ERS-2100's: exposure for each line of the CMOS device is asynchronous. There exists about one frame time delay of the timing of exposure between the first and last line of a CMOS device as illustrated in Fig.8. For ERS-2100, the phenomenon occurs especially when it fast rotates its head. Fig.9(a) shows an example for the velocity of 0.30deg/msec.

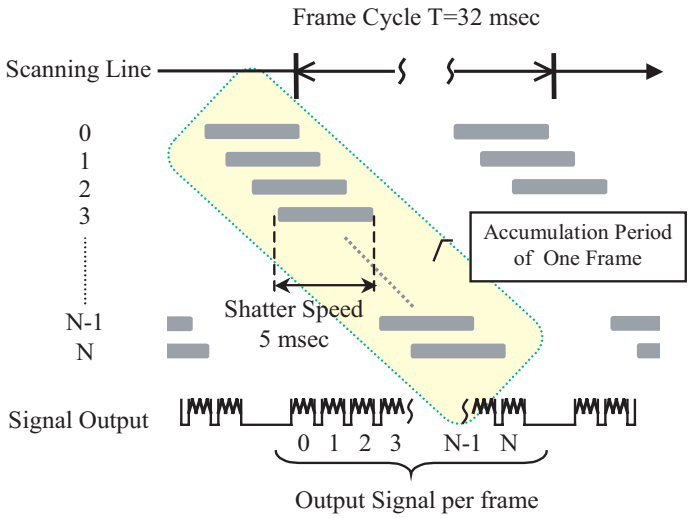


Fig. 8. Line exposure and shutter speed

(a) Real Image

(b) Virtual Image without Exposure Delay

(c) Virtual Image with Exposure Delay

Fig. 9. Exposure delay and shutter speed

5 Evaluation of the Developed Simulator

In this section, we show some experimental results to verify that the proposed simulator well imitates the real environment. One experiment is measurement of the distance to a ball, and the other is self localization, both of which are essential and important technologies for the robocup games.

5.1 Evaluation for Ball Measurement

The distance to a ball was measured by a real robot and by a robot in the simulator. The algorithm to measure the distance was the same for both the real and the virtual robots: distance was measured from the number of pixels of the ball region, the coordinates of the center of the ball region, edge information, etc. The ball was set from constant distances from the robot as shown in Fig.10, and the distance was measured 100 times and the average and the standard deviation were evaluated for each distance. Fig.11 and Fig.12 show the results of the average and the standard deviation respectively. In Fig.11, the simulator measures almost the same distance as the real robot, especially when the camera characteristics are considered. In Fig.12, when the camera characteristics are not considered, the standard deviations are zero; and when the characteristics are considered, the deviations by the simulator show the same tendency as that by the real robot. Fig.10 shows color detection for distance measurement with/without the consideration. When considered, the color detection seems much more realistic.

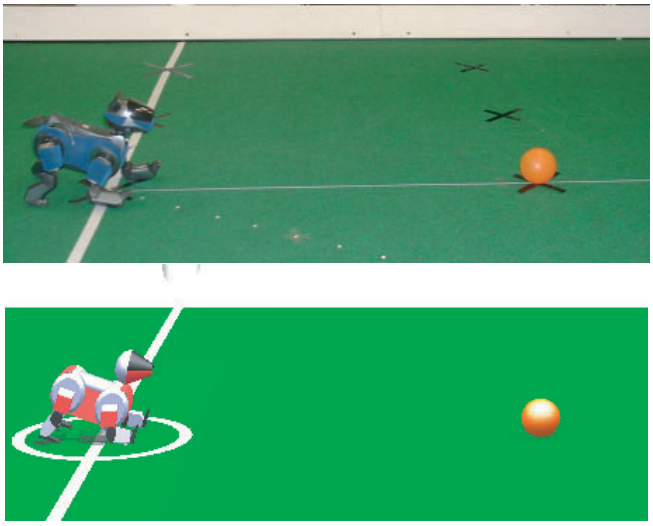


Fig. 10. Setting of distance measurement by real robot(upper), virtual robot in the simulator(lower)

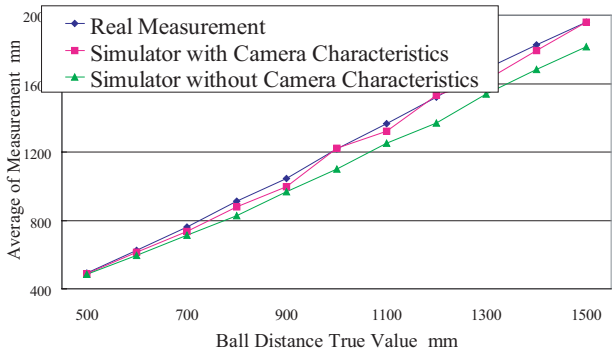


Fig. 11. Comparison of ball distance measurement: average

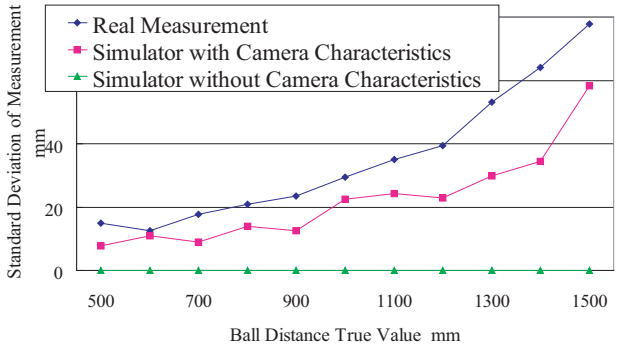


Fig. 12. Comparison of ball distance measurement: standard deviation

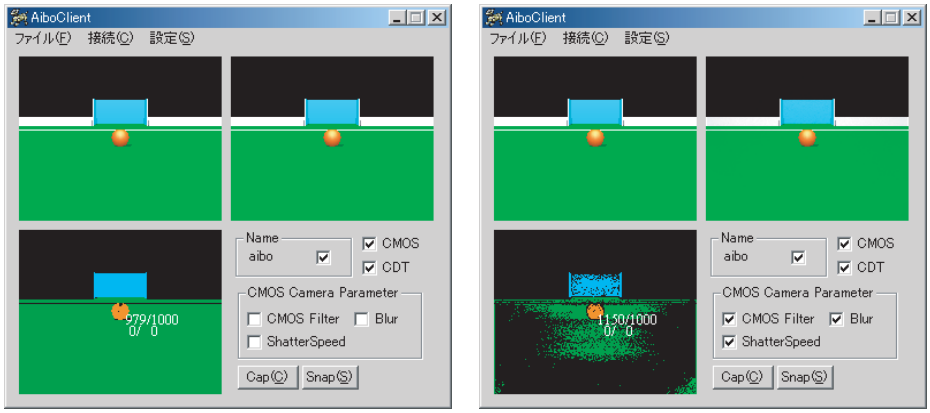


Fig. 13. Color detection for distance measurement by the simulator with/without considering camera characteristics

5.2 Evaluation for Self-Localization

We have been applying Monte Carlo Localization(MCL) that is a kind of Markov Localization methods for self localization of real robots[6]. MCL represents position and orientation of each robot by distributed probabilistic distribution of a lot of sample points. Experiments were performed at 36 positions and orientations of a robot for 9 positions (intersection points of x=0, 1000, 1750[mm] and y=0, 500, 1000[mm]) and 4 orientations ($\theta = \pm 45, \pm 135$[deg]). Self localization was performed by rotating the camera for 30[sec]. Table 1 shows the results.

It is shown that the real robot and the simulator obtain similar results for self localization. However, the errors of the simulator are larger than of the real robot. The reason is perhaps that in the simulator, production of images and the cycle of waving head synchronized, and consequently, images had some bias.

Table 1. Comparison of self-localization

(a) Real robot

	Mean squared error	Max. error	Ave. of division width	True value in division
x	186 mm	393 mm	580 mm	61%
y	141 mm	375 mm	395 mm	58%
θ	6.5 deg	16 deg	19 deg	67%
xy	233 mm	393 mm	/	47%
$xy\theta$	/	/	/	47%

(b) Simulation without camera characteristics

	Mean squared error	Max. error	Ave. of division width	True value in division
x	272 mm	731 mm	603 mm	58%
y	218 mm	542 mm	527 mm	61%
θ	9.9 deg	23 deg	39 deg	77%
xy	349 mm	704 mm	/	42%
$xy\theta$	/	/	/	42%

(c) Simulation with camera characteristics

	Mean squared error	Max. error	Ave. of division width	True value in division
x	291 mm	731 mm	570 mm	55%
y	201 mm	564 mm	426 mm	60%
θ	9.4 deg	26 deg	26 deg	66%
xy	368 mm	807 mm	/	45%
$xy\theta$	/	/	/	45%

6 Conclusion

In this paper, we developed a simulator of environment and measurement that considers camera characteristics. The simulator introduced server/client system, and realized separation of each robot's information, introduction of each robot's

difference and distribution of processes. For producing virtual images, the simulator utilized OpenGL and considered the effects of blur by lens aberration, etc., random noise on each pixel, lens distortion and delayed exposure for each line of CMOS device. Some experiments show that the simulator imitates the real environment well, and is a useful tool for developing algorithms effectively for multiple mobile robots.

References

1. Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa and Hitoshi Matsubara: "RoboCup: A Challenge Problem for AI and Robotics," Proc. of Robot Soccer World Cup I, pp.1-19, 1997.
2. N. Y. Chong, T. Kotoku, K. Ohba: "Development of a Multi-telerobot System for Remote Collaboration," Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 1002-1007, 2000.
3. O. Michel: "Webots: Symbiosis Between Virtual and Real Mobile Robots," Proc. of ICVW '98 Paris France, pp.254-263, 1998.
4. T. Rofer: "An Architecture for a National RoboCup Team," Proc. of Robot Soccer World Cup VI, pp.388-395, 2002.
5. Mark M. Chang, Gordon F. Wyeth: "ViperRoos 2001," Proc. of Robot Soccer World Cup V, pp.607-610, 2001.
6. R. Ueda, T. Fukase, Y. Kobayashi, T. Arai, H. Yuasa and J. Ota: "Uniform Monte Carlo Localization – Fast and Robust Self-localization Method for Mobile Robots," Proc. of ICRA-2002, pp.1353-1358, 2002.
7. J. Neider, T. Davis and M. Woo: OpenGL Proggraming Guide, Addison-Wesley, 1992.