

An Internet Robotic System Based Common Object Request Broker Architecture

Songmin Jia and Kunikatsu Takase
Graduate School of Information Systems,
University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-City, Tokyo 182-8585, Japan

Abstract

Because the elderly population is growing while the number of people to take care of them is declining, we propose an Internet telerobotic system to assist the aged or disabled in their homes when their caregivers are away. In this paper we analyze typical teleoperation robotic systems and design an Internet telerobotic system using CORBA (Common Object Request Broker Architecture). We detail the system's features, architecture, and implementation. The proposed system gives users the ability to control the robotic system remotely by an intuitive user interface. The client can transparently invoke methods on the application servers across the network without knowing where the application servers are located, what programming language the application is written in, or what operating system is being used. This is because, in developing and implementing the system, CORBA for the distributed, object-oriented applications was used.

1. Introduction

With the rapid development of a standard communication protocol and a standard human interface, the technology of the teleoperation of robots, combined with network technology, facilitates a new research field, that of Internet telerobotic systems. Many Internet robotic systems have been developed on the Web, such as the Underwater Remotely Operated Robot, the Museum Tour-Guide Robot, and the Entertaining Robot. Until now, Internet telerobotic systems have typically been operated using two ways communication links between a client and a remote robot system [1]. One is to use CGI (Common Gateway Interface). The Mercury project [2], the CMU Xavier project [3], and KhepOn-TheWeb [4][5] use this method. CGI circumvents the problems of static pages by creating a new page for each HTTP request. Thus a new process must be started for each request. In addition, a complete HTML page must be generated with each request, and the resulting page is still static. So the speed is limited. The other way is to use Java to implement networking connections. The USA PumaPaint project [6] and the Guiding Robot through the Web project [7] use this method. This method avoids the limitations of CGI because Java programs can be run as applets within the browser and supply an interactive interface instead of a static one, since Java is executable within a Web page. But in this case the

client must know the location of all servers to which it wants to connect. Besides that, Java allows applets to connect only to the host they were served from, because of security restrictions. In this paper we propose an Internet robotic system using CORBA. This system allows a client to invoke a method on a server across a network transparently without knowing where the application servers are located, or what programming language and operating system are used.

2. System Features

This system aims to aid aged and disabled persons around the house when human caregivers are away. The primary task of the system would be to supply food and a cup of water or juice to the aged and disabled. To accomplish these tasks, it is very important that the robot system can recognize and manipulate common tableware. In our research, we first implement an Internet hand-eye robotic system using CORBA. In order to aid aged and disabled persons the human caregivers can control the telerobotic system to retrieve and manipulate the desired tableware, by viewing live image feedback from the robot's site through a cellular phone and notebook computer or a common computer. This system has the following features [8]:

- In this system, Common Object Request Broker Architecture (CORBA) was selected because it combines the benefits of object-oriented and distributed computing and uses an Object Request Broker (ORB) as the middleware that establishes client/server relationships between objects.
- This system allows clients to invoke methods on server objects transparently across the network because the ORB is capable of delivering the client requests to the objects that can implement the client's requests, passing it the parameters and returning the resulting information across the Internet.
- The components of the system can be implemented by different programming languages. The Internet robotic system integrates many elemental technologies such as image processing, force sensing, motion planning, and motion control. It is very necessary to use the different programming language that is optimal to each technology, in order to create the optimal system.

- The CORBA-based system can be implemented by a number of independent subsystems and these subsystems can further be implemented by a number of the independent objects. These components of the system can be run independently to implement the application. These components of the system are also integrated easily into new systems [9].

3. System Hardware

Fig. 1 shows the architecture of the system.

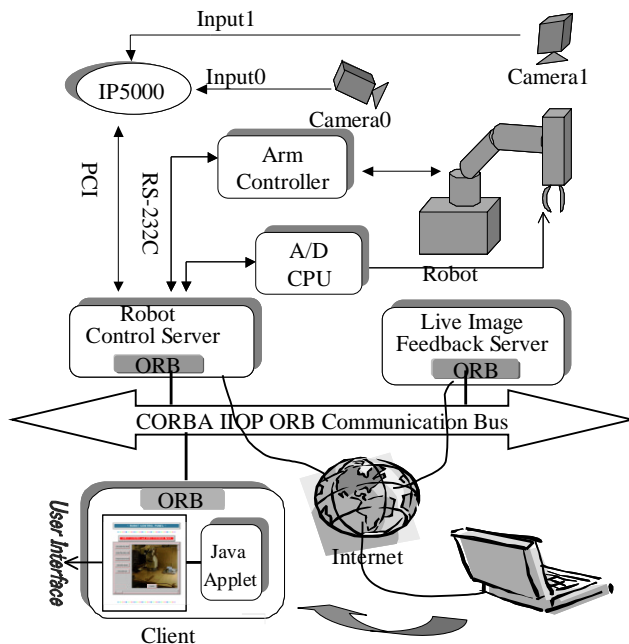


Figure 1. Architecture of the system

3.1. Vision system

The vision system consists of cameras and an image processing board. Two VC-CI MKII cameras are used. Camera0 is positioned at the place where it can provide user an feedback image easy to understand, Camera1 is used to take an image of the tableware scattered on the table in this system. A high-speed image processing board, IP5000, was used for processing the obtained images. The resolution of the vision system is 512 x 512 pixels, and it takes 3.6 milliseconds to process each image.

3.2. Robot arm and its controller

The Mitsubishi Movemaster Super RV-E3J was used in this system. It consists of a manipulator with five degrees of freedoms, a controller, and a teaching box. The robot arm controller is connected to the host computer via an RS232-C link and controls the arm according to the commands coming from the host computer.

3.3. Robot hand and its controller

To prevent the robot from breaking tableware it handles, force sensors are affixed to the robot fingers. We designed the CPU control system to measure the grasp force and the servo-driving circuit to drive the fingers (Fig. 2). Futaba's S3801 high-torque metal gear servo was used to drive the fingers. Its weight is about 107 g, its operating speed is 231°/sec, and its output torque is 14.0 Kg-cm. This servo can act on PWM (Pulse Width Modulation). The basic cycle is 20 ms. We generate input control pulses by a PTC (Programmable Timer Controller) to control the servo. The robot hand works according to the commands coming from the CPU controlled by the host computer via an RS232 link.

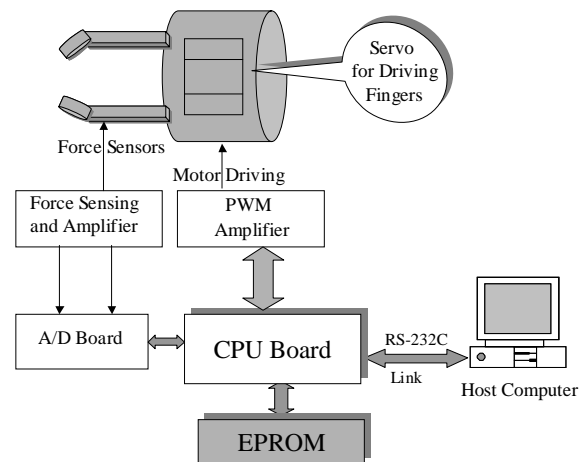


Figure 2. Configuration of control circuit

3.4. Host computer

A workstation with a Pentium III processor running Windows NT 4.0 was used as the host computer. RS232-C Port 0 is connected to the manipulator's controller, and RS232-C Port 1 is connected to the CPU board.

4. System Software Implementation

4.1. Web-based user interface

The challenge of designing a user interface is to provide enough information to let the user operate this system easily while minimizing transmitted data and maintaining a simple layout. Besides that, all kinds of complex calculations should be hidden from the user in order to support a wide range of users. The Web user interface designed is shown in Fig. 3. It consists of:

- Live image feedback part: This allows the user see the robot working in the remote site.
- Robot control commands part: To cope with time delays on the communication path, we provide user task-

level robot control commands to control the robot. These commands include “Grasp the green cup,” “Coffee, Please” etc.. Once one task-level command button is pushed, the other task-level control command buttons will be invalid until this task is finished for safety.

- Message text box: When the user pushes the button “grasp the spoon”, this command will be sent to server and the necessary result message will be feedback to the client by popping one message text box on the Web user interface.

If the user’s computer can link to the Internet, he or she can access the robotic system anywhere by using this intuitive user interface.

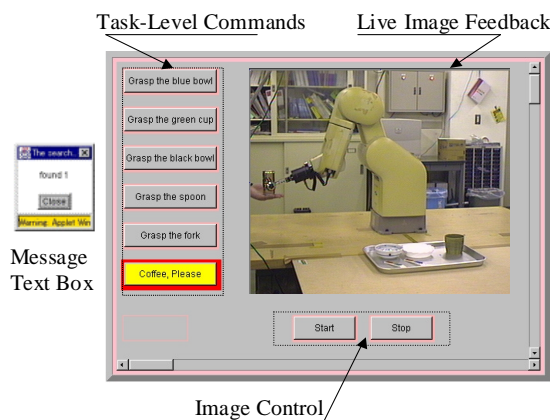


Figure 3. Web-based user interface

4.2. Task-level robot control

In our system, we have implemented robot control server by the optimal programming language C++ because C++ is the best language for the core functions of the system, such as imaging processing, robot control. This server gives user the ability to control the robotic system remotely at a task-level [10]. The client can transparently invoke the methods on this application server across the Internet to recognize and manipulate the tableware that they want. Fig. 4 shows robot control server and its interface. The assignments of the server is as follows:

- Receive commands from the remote World Wide Web users.
- ORB intercepts the “call” and is responsible for finding an object that can implement the request, passes it the commands.
- Find the location and orientation of the tableware scattered on the table by the vision system.

- Generate the task plan and send the commands to the devices that can implement the tasks.
- ORB returns the feedback results to the remote clients.

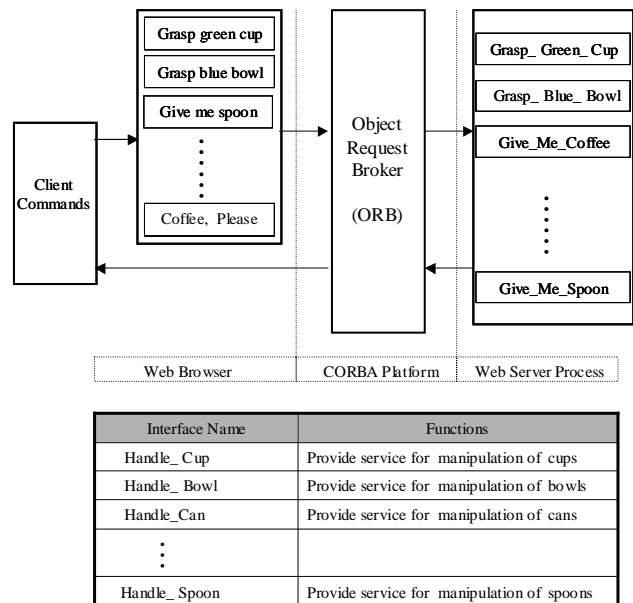


Figure 4. Robot control server and its interface

For one method on the robot control server, it consists of (Fig. 5):

- The information part consists of a vision part and a force measure part. It is the source of information for the system.
- The task planning part receives the information from the information part, recognizes the location and orientation of the tableware scattered on the table, transforms these coordinates to the manipulator’s coordinate system, and generates task plan to achieve the goal.
- The implementation part executes motion scheduling generated by the task planning part, and implements the task according to the commands coming from the server computer.
- The communication part is responsible for the communication between the server computer, the robot’s arm and the robot hand’s controller via RS232-C links.

4.3. Live image feedback

This task is provided by the live image feedback server implemented by C++. This server allows live image feedback from the camera and continuously sends live images

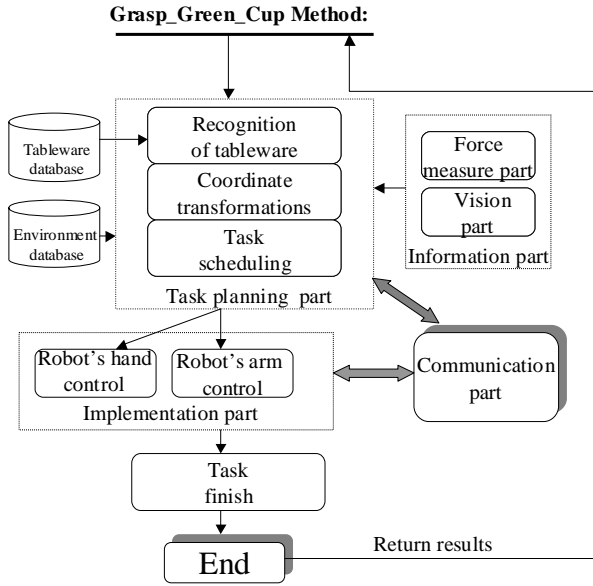


Figure 5. Method of robot control server

of the working robot to the client according to the user's requests. It works as:

- Receive image control commands from the remote users.
- ORB intercepts the commands and transfers the requests to live image feedback server.
- Get the new image by camera0 and change the image into a BMP format by IP5000 image processing board.
- Compress this image into JPEG format.
- Return the new image with the last information to the client by ORB.

4.4. Server implementation by IDL

In CORBA, we use OMG interface definition language (IDL) to define the object interface and specify the types and operations that the object supports. IDL has features like language independence, distributed service specification, definition of complex data types, and hardware independence. By compiling this interface with an IDL compiler for mapping to client/server source code in a specific programming language, we can get client stub and server skeleton source code for communication between client and server. Fig. 6 shows our server implementation by IDL in this system.

5. Recognition Strategy for Tableware

For long tableware, if we want to handle them exactly, we must know not only the location but also its orientation. In

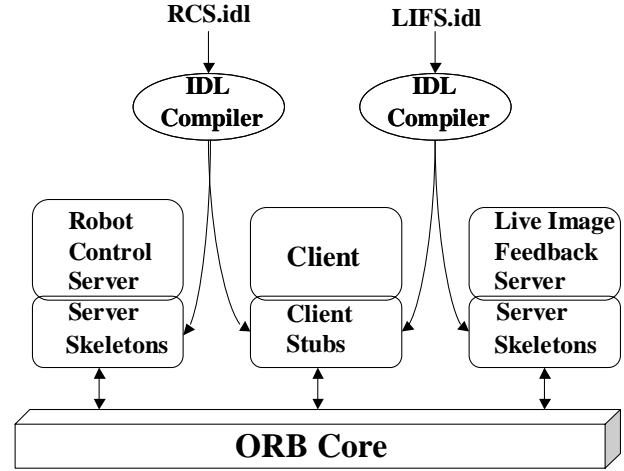


Figure 6. Server implementation by IDL

order to implement real-time recognition of the tableware, the color-based technique was used. In this technique, we assume that spoons and forks consist of two different color parts. Using this technique, long tableware recognition can be replaced by the simple mark recognition because there is one-to-one correspondence between tableware and different combination of color marks. We can create the tableware's geometric model and entry them beforehand. By doing this we can recognize the tableware easily and quickly [11].

Assume that there are two differently colored parts on

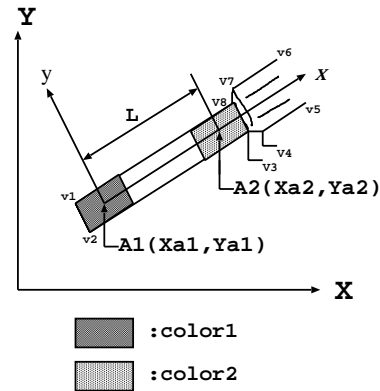


Figure 7. Model of tableware

one fork (Fig. 7). We can get each center point of the color part by using image processing. Let us call them A1 and A2. X_{a1} , Y_{a1} , X_{a2} , and Y_{a2} are their coordinates with respect to an absolute world reference coordinate (X, Y). A1 is assumed to be the origin of the vector, the unit of vector along the x and y axes, and the length of this vector can be given by

$$e_x = \left(\frac{X_{a2} - X_{a1}}{L}, \frac{Y_{a2} - Y_{a1}}{L} \right)^T \quad (1)$$

$$\mathbf{e}_y = \left(-\frac{Y_{a2} - Y_{a1}}{L}, \frac{X_{a2} - X_{a1}}{L} \right)^T \quad (2)$$

$$L = \sqrt{(X_{a2} - X_{a1})^2 + (Y_{a2} - Y_{a1})^2} \quad (3)$$

In order to know the shape of the tableware, we define some different feature points for different kinds of tableware. For example, we can choose v1, v2, v3, v4, v5, v6, v7, and v8 vertices as the feature points of the fork, and we can find their x, y coordinates under the object coordinate frame by:

$$\begin{cases} x_{vi} = (X_{vi} - X_{a1}, Y_{vi} - Y_{a1}) \cdot \mathbf{e}_x \\ y_{vi} = (X_{vi} - X_{a1}, Y_{vi} - Y_{a1}) \cdot \mathbf{e}_y \end{cases} \quad (4)$$

Here, X_{vi} , Y_{vi} , X_{a1} , Y_{a1} can be gotten by moving the cursor on the monitor. The posture angle (with respect to X axis) of the tableware can be calculated by

$$\theta = \begin{cases} \arctan \frac{Y_{a2} - Y_{a1}}{X_{a2} - X_{a1}} & \text{if } Y_{a2} - Y_{a1} > 0 \\ 2\pi + \arctan \frac{Y_{a2} - Y_{a1}}{X_{a2} - X_{a1}} & \text{if } Y_{a2} - Y_{a1} < 0 \end{cases} \quad (5)$$

For circular tableware, the quick correlation calculation was used. Correlation calculation is a method that evaluates the matching between the template image registered beforehand and the input image obtained from the camera. The value of a correlation calculation varies from 0 to 1 according to the case. It equals 1 if the part of the search area is completely the same as the template image. It is better that the illumination of the template gotten is the same as that of the applied condition. In spite of doing this, the template is also circumscribed the other problems. We have done some experiments such as changing the position and the angle seen from the fixed camera coordinate frame. The experiment results are shown in Fig. 8.

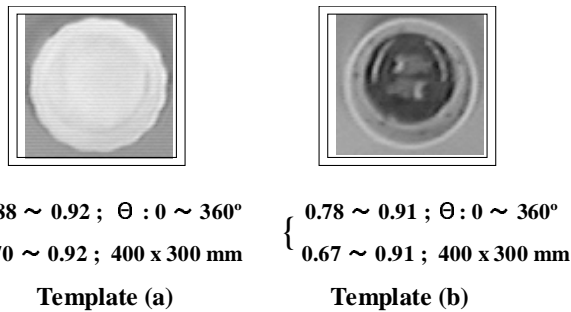


Figure 8. Templates and its experiment results

According the results of our experiments, we know not only a different object has a different correlation value, but

also the same object will have a different correlation value in different condition. For example, template (a)'s correlation value will change from 0.88 to 0.92 if we change the angle seen from the fixed camera coordinate frame from 0° to 360° , and correlation value will change from 0.70 to 0.92 if we move it to different position on the table. This results from tableware's no uniform distribution and no uniform illumination under the application environment. So, we must measure the minimum correlation value and it should be bigger enough than the one that can distinguish itself from the other tableware and the environment. We have done some experiments and made up tableware's correlation matrix shown in Fig. 9. According this, we can know whether the tableware can be distinguished itself from the other tableware and the background or not. For example, for template(b), the correlation value of background is about 0.2, the maximum of the correlation value of the other tableware is about 0.3, the correlation value of the template(b) changed from 0.67 ~ 0.91. According to this, we can distinguish template(b) from the other tableware and the background.









Template Target					Background
	0.60 0.92	--	--	--	0.38
	--	0.70 0.92	0.2	--	0.1
	0.2	0.1	0.77 0.91	--	0.33
	0.3	--	--	0.67 0.91	0.2

Figure 9. Tableware's correlation matrix

6. Experimental Results

In our system, we have implemented two application servers (Robot Control Server and Live Image Feedback Server) by C++. We also used Java for the client because Java applets are capable of directly accessing CORBA application servers via IIOP and are easy to run within Web browsers. This system have been accessed by the operators from Tama-City (Client1), Saitama-City (Client2), Kyoto University (Client3), Japan and Chicago (Client4), America etc.. The specifications of PCs and the communication paths of Client1, Client2, Client3, Client4 are shown in Fig. 10. Some intervals between the users pushing the "start" button and the new image with the latest information showing have also been recorded. The average time of 10 times for

Client1 is about 26 seconds, for Client2 is about 10 seconds, for Client3 is about 8.8 seconds, and for Client4 is about 6.8 seconds. These values may vary according to the specifications of the user's computer, the communication path and the time the operator accesses the system. Fig. 10 shows the scene of users accessing this system. When the remote user pushes the command "Coffee, Please", the system can automatically find the coffee can and execute this task robustly.

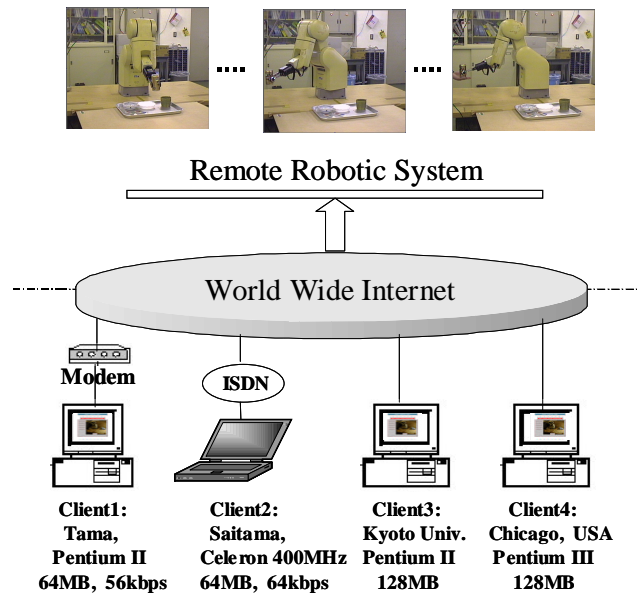


Figure 10. Remote machines on the Internet

We also have done the experiments as: the Robot Control Server is working and the Live Image Feedback Server is down. In such a case, the user can still access the Robot Control Server normally. Similarly, the user can still access the Live Image Feedback Server normally when the Robot Control Server is down.

7. Conclusions

By introducing CORBA communication into a Java applet, a Java client is capable of directly accessing CORBA application servers via IIOP instead of static, it can run on different platforms, and it can transparently invoke methods on the application servers across the network. This lets the system overcome the shortcomings of the other typical Internet telerobotic systems. Besides that, The CORBA-based system can be implemented by a number of independent subsystems and these subsystems can further be implemented by a number of the independent objects. These components of the system can be run independently to implement the application. The other components of the system can work normally even if there are troubles with some of them. So, using CORBA for implementing Internet teleoperation systems is very effective and much more flexible and robust.

8. Acknowledgments

We would like to thank Dr. Hirohisa Hirukawa (ETL) for his helpful comments.

References

- [1] Barney Dalton and Ken Taylor, A Framework for Internet Robotics, Proc. of 1998 IEEE/RSJ, Conference on Intelligent Robots and Systems; Workshop on Web Robots, pp.15-23, 1998.
- [2] K.Goldberg, K.Mascha, M.Genter, N.Rothenberg, C.Sutter and J.Wiegley, Desktop teleoperation via the world wide web, Proc. of 1995 IEEE Conference on Robotics and Automation, pp.654-659, 1995.
- [3] Reid Simmons, Xavier: An Autonomous Mobile Robot on the Web, Proc. of 1998 IEEE/RSJ Conference on Intelligent Robots and Systems; Workshop on Web Robots, pp.43-49, 1998.
- [4] Patric Saucy, KhepOnTheWeb: One Year of Access to a Mobile Robot on the Internet, Proc. of 1998 IEEE/RSJ Conference on Intelligent Robots and Systems; Workshop on Web Robots, pp.23-31, 1998.
- [5] Patrick Saucy and Francesco Mondada, Open Access to a Mobile Robot on the Internet, IEEE Robotics and Automation Magazine, Vol.7, No.1, pp.41-47, 2000.
- [6] Matthew Stein, Painting on the Web, The PumaPaint Project, Proc. of 1998 IEEE/RSJ Conference on Intelligent Robots and Systems ; Workshop on Web Robots, pp.37-43, 1998.
- [7] Roland Siegwart, Cedric Wannaz, Patrick Garcia and Remy Blank, Guiding Mobile Robots through the Web, Proc. of 1998 IEEE/RSJ Conference on Intelligent Robots and Systems ; Workshop on Web Robots, pp.1-7, 1998.
- [8] Seán Baker, CORBA Distributed Objects Using Orbix, ACM Press, UK, 1997.
- [9] Nestor Michelena, Christopher Scheffer, Ryan Fellini, and Panos Papalambros, CORBA-Based Object-Oriented Framework for Distributed System Design, MECH. STRUCT. & MACH., 27(4), 365-392, 1999.
- [10] Songmin Jia and Kunikatsu Takase, Network-Based Human Assist Robotic System Using CORBA, The Sixth International Symposium on Artificial Life and Robotics, pp.105-109, 2001.
- [11] Y.Hada and K.Takase, Task-Level Feedback Control of a Robot Based on the Integration of Real-Time Recognition and Motion Planning, Proc. of 28th International Symposium on Robotics, pp.1353-1363, 1997.