Robotica (2009) volume 27, pp. 459–468. © 2008 Cambridge University Press
doi:10.1017/S026357470800489X Printed in the United Kingdom

# A CORBA-based simulation and control framework for mobile robots

Zhang Zhen*, Cao Qixin, Charles Lo and Zhang Lei

*Research Institute of Robotics, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, 200240, P.R. China.*

(Received in Final Form: June 15, 2008. First published online: August 12, 2008)

**SUMMARY**

This paper presents a distributed multiple mobile robots framework which allows programming and control of virtual and real mobile robots. The system provides the map building, path planning, robot task planning, simulation, and actual robot control functions in an indoor environment. Users can program the virtual robots in a customized simulation environment and check the performance of execution, i.e., if the simulation result is satisfying, users can download the code to a real robot. The paper focuses on the distributed architecture and key technologies of virtual robots simulation and control of real robots. A method for construction and transfer of a key index value (which stores the robot configuration) is proposed. Using this method, only the robot key configuration index is needed to build the robot in the virtual environment. This results in reduced network load and improved real time performance of the distributed system. Experiments were conducted to compare the performance of the proposed system with the performance of a centralized system. The results show that the distributed system uses less system resources and has better real time performance. What is more, this framework has been applied to Yaskawa's robot "SmartPal." The simulation and experiment results show that our robotic framework can simulate and control the robot to perform complex tasks.

KEYWORDS: Mobile robot; CORBA; Simulation; Real robot control.

## 1. Introduction

In recent years there is an increase in the use of robots in fields of replacing humans to fulfill complex and dangerous tasks. The robotic system must be reliable and coordinated while finishing different subtasks such as perception, planning, and navigation. Thus some robotic platforms have been developed to test the control algorithms and to evaluate the robot performance.

Currently, we at Research Institute of Robotics in Shanghai Jiao Tong University are developing a robotic simulation and control framework in collaboration with robot producer Yaskawa Electric Corporation, Japan, entitled "Simulation of Mobile Robots Navigation (SMRN)," based on their interest in distributed precise simulation and control for mobile robot SmartPal.[1] This project was initiated and a new framework proposed due to the lack of specific functionalities which are prerequisites specified by Yaskawa for their SmartPal robots, such as, support for multiplatforms (OS) or system portability, 2-D and 3-D generator and simulator, etc., which are not currently supported in other simulation frameworks. These other frameworks will be briefly described in the next section.

Our previous work realizes a centralized simulation and control system. The users can easily program a dual-arm mobile robot, preview, and check the robot motion in a 3-D simulation environment.[2] However, due to a need for a large amount of precise calculations that need to be performed, the system uses a considerable amount of computing resources, which makes it difficult for it to be used on a normal PC for multiple mobile robot simulations. Thus, a distributed flexible navigation simulation system based on CORBA[3–5] was developed. We have also implemented our system into SmartPal robots. There are currently no other existing CORBA systems which provide such a complete set of services for mobile service robots.

Our system can be divided into several components: a MapEditor server, a simulation server, and robot clients. They are connected via the CORBA bus and can be deployed in different PCs with different operating systems, which extends the portability of the system. The system provides map building, path planning, simulation, and actual robot control services. A robot "key index" is proposed to describe the robot's model. Each robot's kinematics motion can be calculated in different client modules separately. Each client processes computationally expensive tasks such as calculating the related robot's kinematics motion according to its key index value and rebuilding the robot and its local environment scene. Other key technologies, such as multirobot simulation and control mechanism, and seamless migration between simulated and actual robots are also proposed.

The remainder of the paper is organized as follows: Section 2 introduces the related works. Section 3 presents our distributed simulation and control simulation system architecture. Section 4 introduces the related mobile robot SmartPal. The proposed key technologies are presented in Section 5. Experiments to test the distributed system are shown in Section 6. And the conclusion is given in Section 7.

* Corresponding author. E-mail: zzh2200_0@126.com, zzh2000@sjtu.edu.cn

## 2. Related Works

With the rapid progress in computer and communication technology, robotic systems are fast becoming larger and more complicated. Therefore, a framework is required that can integrate reusable components for which various companies and individuals contribute their technologies. Many researchers have proposed and implemented their solutions respectively. ORiN (Open Resource interface for the Network/Open Robot interface for the Network) is a middleware framework, which offers the standard communication interface over various FA (factory automation) equipment including a robot, but it is mainly developed for industrial robots in some structural environments.[6,7] Orocos is a free software project that includes a set of class libraries and application framework, and a hard-real-time kernel for all possible feedback control applications.[8,9] Toshiba has proposed the open robot controller architecture (ORCA) based on the robot technology (RT) reference model proposed by Toshiba, so as to allow RT components to be easily packaged. ORCA uses distributed object technology to enable such components to be used transparently anywhere via the network.[10,11] SONY is actively promoting OPEN-R, which involves the use of modular hardware components, such as appendages that can be easily removed and replaced to change the shape and function of the robots, and modular software components that can be interchanged to change their behavior and movement patterns. However it is only developed for SONY's four-legged entertainment robot prototype.[12]

BREVE is a simulation environment meant for the development of artificial life in a physically simulated world. It uses a scripting language that allows control strategies and event-based reactions to the environment for large numbers of agents.[13] CARMEN[14] uses the middleware framework MARIE (Mobile and Autonomous Robot Integrated Environment)[15] to build the mobile robot control and simulation system. However, there is still not an integrated platform that supports customized environment modeling, graphical programming, virtual robots simulation, and real robots control functions. Some platforms, such as Player/Stage/Gazebo, provide environment modeling, simulation, and real robot control, but it can only be used on a Linux operating system.[16,17] The Microsoft® Robotics Studio is a Windows-based environment for hobbyist, academic, and commercial developers to create robotic applications for a variety of hardware platforms. It includes a lightweight REST-style, service-oriented runtime, a set of visual authoring and simulation tools, as well as tutorials and sample code to help users get started. However, all of these systems are mainly developed for general robots, so it is difficult to realize an appropriate simulation and real control for redundant dual-arm mobile robots.

## 3. System Architecture

The architecture of the distributed system is presented in Fig. 1. It comprises three components: a MapEditor server, a simulation server, and robot clients. The MapEditor server is responsible for an indoor simulation environment model building and path planning services. The simulation server provides the virtual sensor and simulation services to allow obstacle avoidance. Its Omni database keeps records of all the information of the simulation. The robot client invokes the methods from the CORBA server and presents the 3-D simulation result to the users. These functional parts communicate via the CORBA bus.

As a result of using CORBA, the service implemented object in these servers can be remotely and transparently invoked from the clients regardless of their hardware, operating systems, and programming language. All the servers and clients can be distributed on different computers using CORBA middleware, the JavaTM IDL developed by Sun Microsystems.[18] JavaTM IDL is freely available and is a fully compliant implementation of the CORBA standard. It provides interoperability between applications on different machines in heterogeneous distributed environments and

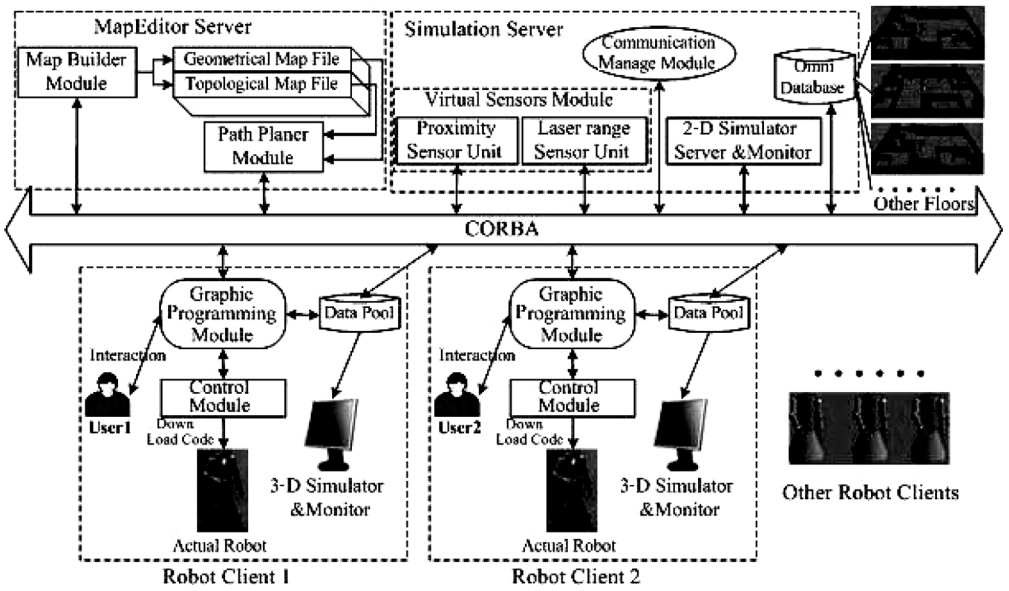



Fig. 1. System architecture.

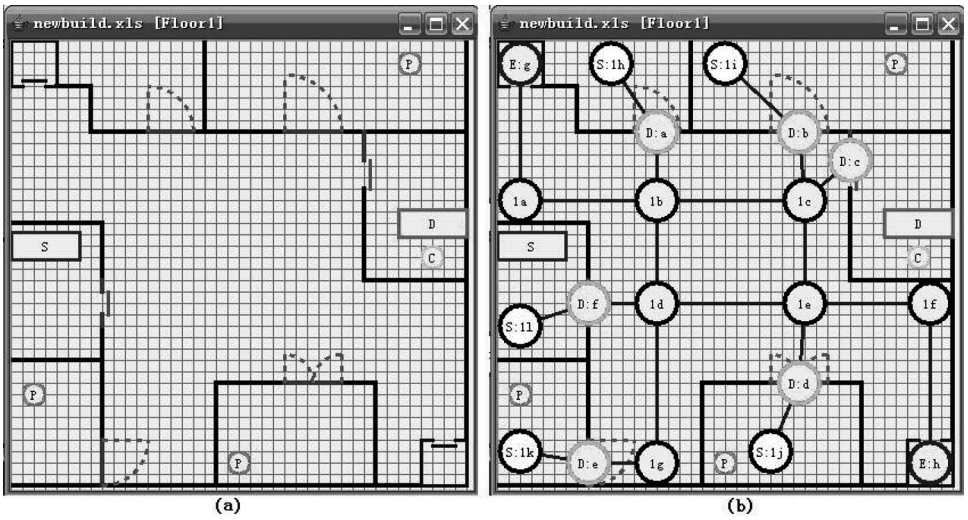



Fig. 2. Two maps in a MapEditor: (a) geometrical map, (b) topological map.

seamlessly interconnects multiple-object systems. Each component of the system is presented as follows.

### *3.1. MapEditor server component*

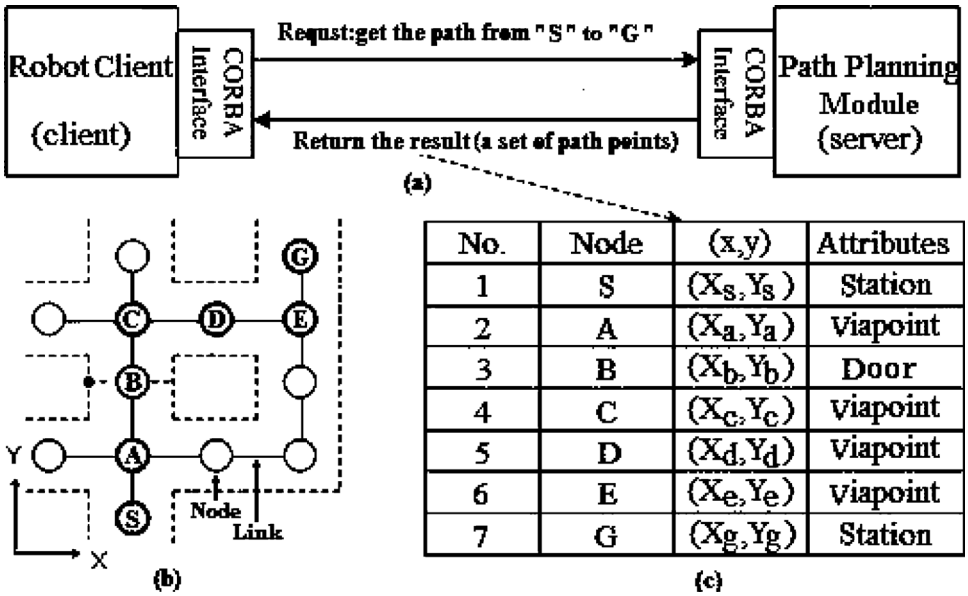
The MapEditor server is used as a unique virtual environment for multiple mobile robots to work in. The server contains two modules: a map building module and a path planning module. The user can build a customized environment according to a real world using a map building module, save the objects' shapes, positions, and other geometrical data in a geometrical map, and save the path nodes in a topological map as well (Fig. 2).

The path planning module is used to determine a feasible path between the start and the end points specified by the user. Figure 3 shows a communication example between the robot client and the path planning module. In Fig. 3(a), the client specifies the start point and the end point, calls the "getPath" method using the CORBA interface, and receives a set of path points from the path planning module. Figure 3(b) shows the map for topological information. Figure 3(c) shows the format of path points returned by the server.

### *3.2. Simulation server component*

In the simulation server, the communication management module is responsible for recording all the registered information from the clients and to realize a simple load balance. All robot clients that want to join the simulation environment must first register with this module. The management module accepts the client's request and puts the client's name in the register list. The robot client can call on the required methods using the CORBA interface. If the number of clients reaches the upper limit of the permissions allowed, the communication management module will disallow other clients from logging on to the server.

All the simulation data is stored in the Omni database. It contains an elevators data list and a floor data list which




| No. | Node | (x,y) | Attributes |
|---|---|---|---|
| 1 | S | $(X_s, Y_s)$ | Station |
| 2 | A | $(X_a, Y_a)$ | Viapoint |
| 3 | B | $(X_b, Y_b)$ | Door |
| 4 | C | $(X_c, Y_c)$ | Viapoint |
| 5 | D | $(X_d, Y_d)$ | Viapoint |
| 6 | E | $(X_e, Y_e)$ | Viapoint |
| 7 | G | $(X_g, Y_g)$ | Station |

Fig. 3. Path planning communication: (a) communication process, (b) metric and topological identifier, (c) Data structure.

records multiple floor information of a whole building. This data structure makes it easier to exchange the single floor's data between the Omni database (simulator server) and the data pool of a robot client.

The 2-D simulator server loads the data from the Omni database and draws the simulation scene in a 2-D image. This tool has the capability of displaying the simulation scene and to allow the user to monitor the simulation in the whole building. In the monitor panel, users can observe different floor scenes, see the robots' positions and velocities, and receive the data from the virtual sensors.

The virtual sensor module provides virtual laser range finder service and proximity sensors service. It receives the robot client's request with parameters such as the position of the robot and the position and direction of the virtual sensor, and returns laser scan angles and distance information. The information is similar to the information returned by a real sensor but it does not take into account the sensor error and the noise. Usually, a virtual robot just follows a trajectory predefined by a graphic programming module, and the virtual sensor data are used by the virtual robot to detect the dynamic obstacles which are not predefined. So the virtual robot can take into account the obstacles present and feed this data into the path planning module and obtain a new feasible path.

### *3.3. Robot client component*

The robot client is a development module for users to program, control, and observe a virtual robot in a simulated environment or an actual robot in the real world. It is structured in the CORBA client that calls methods from different server components. One robot client component stands for one virtual or actual robot. The modules in the robot include the following:

- *Graphic programming module (GPM)*: It is used to specify the robot's task by using a list of motion icons (Fig. 4). The user can edit the robot key index values in the teaching box to define a motion and check the result in its 3-D viewer. Once an icon is programmed, it can be saved into an icon list. Using this module the user can program the robot to complete the motion tasks.
- *Control Module*: There are two modes: virtual robot mode and the actual robot mode. In the virtual mode, the module calls the SmartPal robot's virtual "Kinematics Engine"[19] to calculate each linkage position, and returns the results to the GPM. Then the GPM sends the robot's data to the data pool for 3-D simulation in the robot client. In the actual robot mode, the module can download the codes to an actual robot to execute the related task.
- *Data Pool*: It is used to save simulation data related to the current robot. For example, if a virtual robot is on the second floor in a building simulation environment with multiple floors, its data pool only keeps the objects' information on the second floor.
- *3-D Simulator & Monitor*: It loads the simulation data from the data pool, and constructs the robot in a 3-D virtual environment. When the simulation is verified, users can download the codes to a real robot for execution.

## 4. Mobile Robot SmartPal

The system which is presented in this paper has been realized on the dual-arm mobile robot SmartPal. So far it is mainly used as a service robot. Figure 5 shows a SmartPal used for experiments in an indoor environment. It is equipped with a pair of 7-DoF (degrees of freedom) arms and grippers. An omni-directional wheel platform is used for planar motion. The sensors equipped in the robot include a laser range sensor in the waist and eight proximity sensors mounted around the omni-directional wheel platform. The Liquid Crystal Display and touch panel are attached to the front of the robot's upper body. They are used for displaying the robotic current internal state (e.g., working, waiting, exceptional) or for user interaction (e.g., choosing the work model).

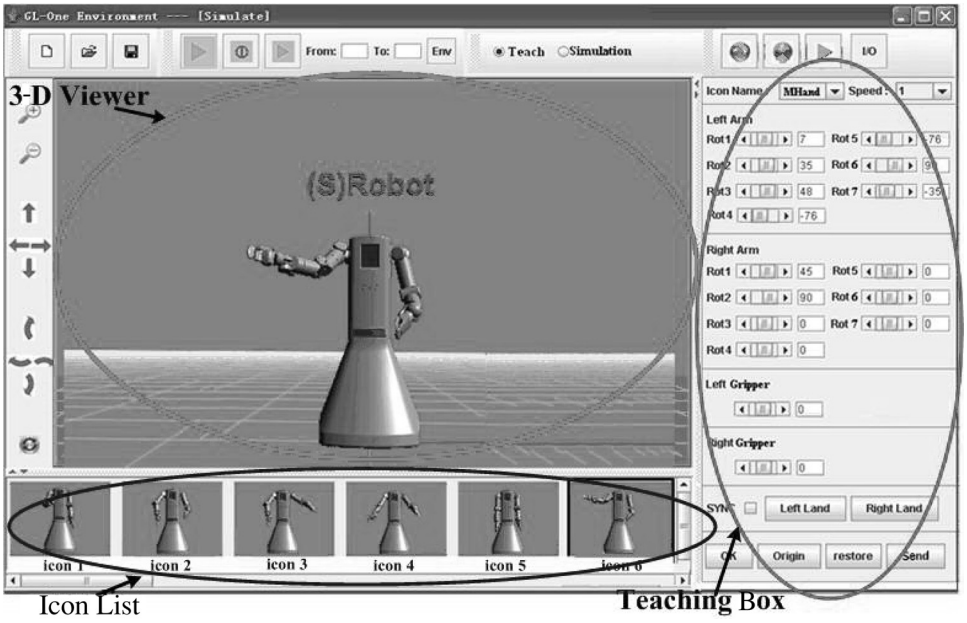



Fig. 4. The graphic programming module.

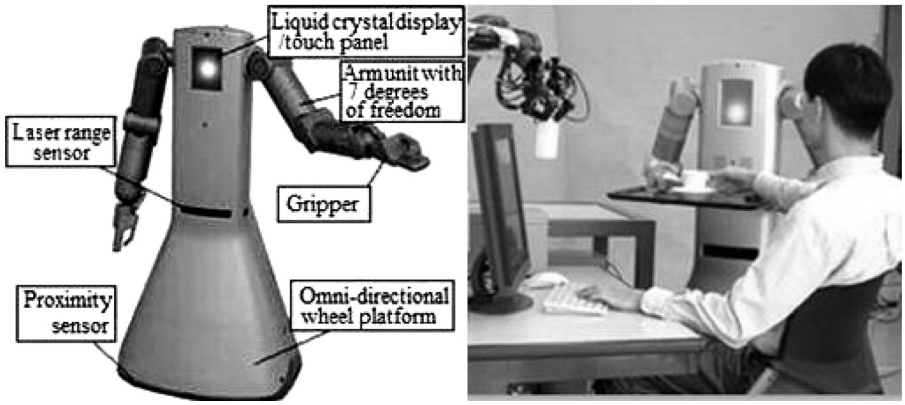



Fig. 5. SmartPal robot.

## 5. Key Technologies in the Distributed Simulation and Control

### *5.1. Robot's key index definition and construction*

The virtual robot is constructed using Java3D. The virtual robot comprises several basic parts which are described by VRML (Virtual Reality Modeling Language). To assemble a SmartPal robot, the basic parts are defined as a Branch Group and the robot joints are defined as a Transform Group. These joints and parts of a SmartPal robot, shown in Fig. 6, are described in Java3D as a node chain.

We define the robot joints' angles, positions, and the distance values in $x$ and $y$ directions as the key index. Only key index is needed to construct the robot.

In order to realize the key index transfer in the distributed system, we define the robot's key index data structure in the CORBA interface. Its UML is shown in Fig. 7.

Users can invoke the methods "send_KeyIndex()" and "receive_KeyIndex()" to transfer and share multiple robots' key indexes. Once the key index is obtained, we use it to rebuild the related robot and control the virtual or real robot. Figure 8 displays the coordinate frame chain, rotation directions, and angles of robot rotation. In this way, the position of each joint in robot structure can be described.

### *5.2. Multirobot control and simulation mechanism*

Multithreading in Java is used to realize the multirobot control or simulation. Two threads are set up: a manipulation

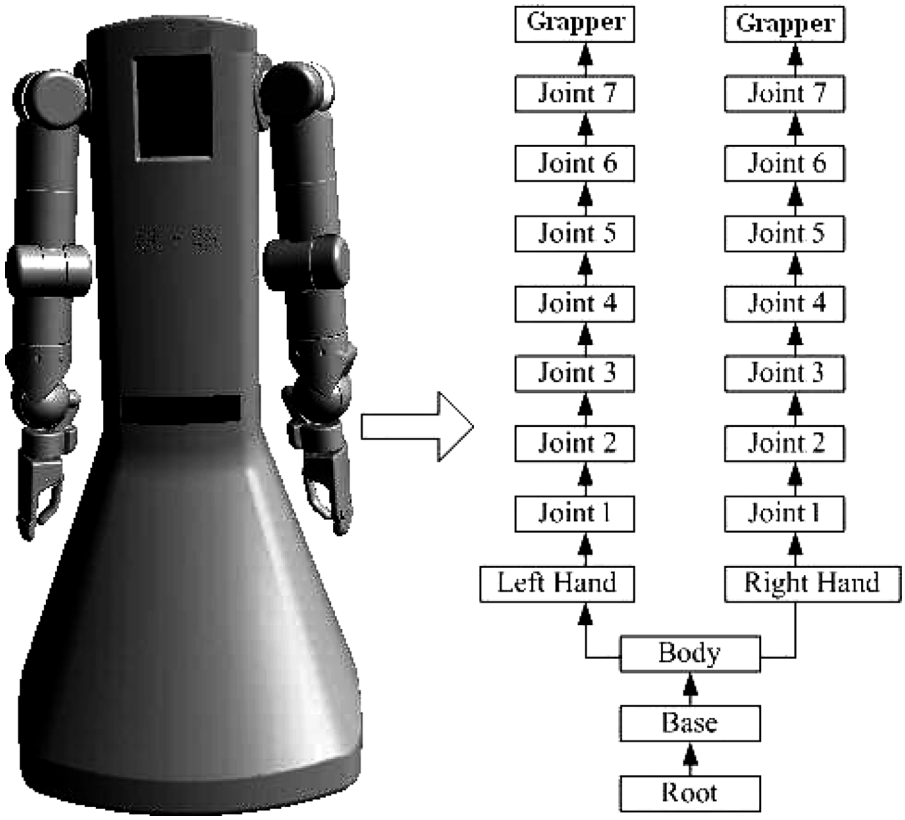



Fig. 6. Architecture of robot basic nodes.

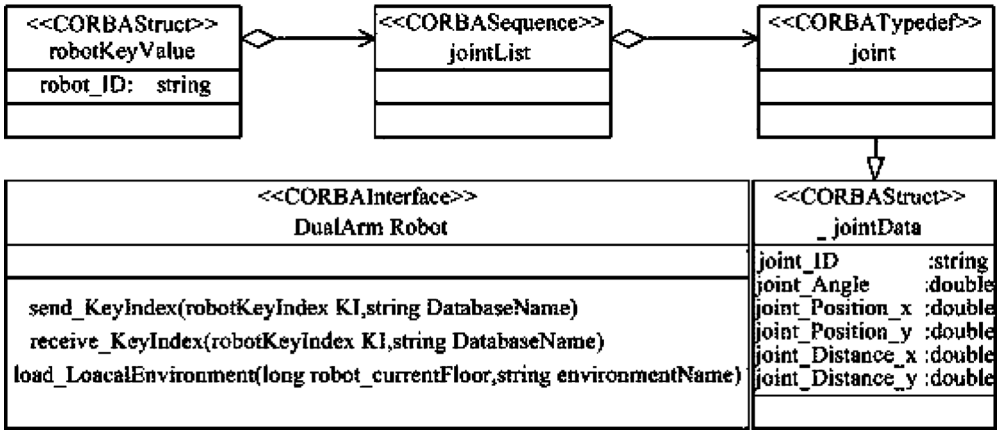



Fig. 7. The UML of CORBA interface.

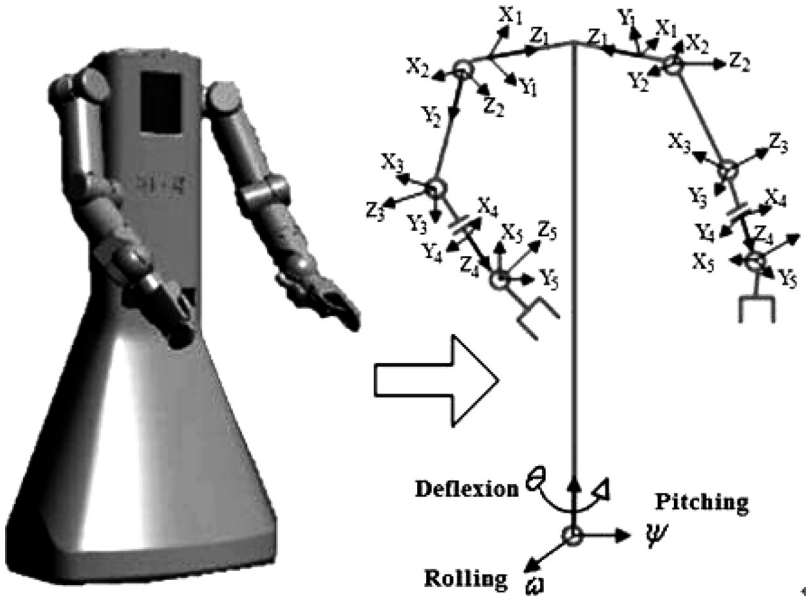



Fig. 8. Coordinate chain on robot rods.

thread and a detection thread. In order to improve the real time performance of the system, a data pool is created in the client to store the current local environment and robots' state information. The robot client need not always invoke the whole Omni database information on the server side, but just call the local environment data from its own data pool. The threads mentioned access the data pool alternately. There is a "supply and demand" relationship between them. As presented in Fig. 9, the manipulation thread calls the control module to obtain the current robot information and updates the data in the data pool. The data pool also exchanges data with the Omni database. For more efficiency, if a robot is positioned on a certain floor, in the client's data pool, only the data of the robots and objects on that specific floor is loaded. The detection thread checks the data pool constantly and, as soon as there is a change in the data pool, the robot is rebuilt in the 3-D simulator and monitor module. This mechanism is implemented for all the robots in a multirobot system.

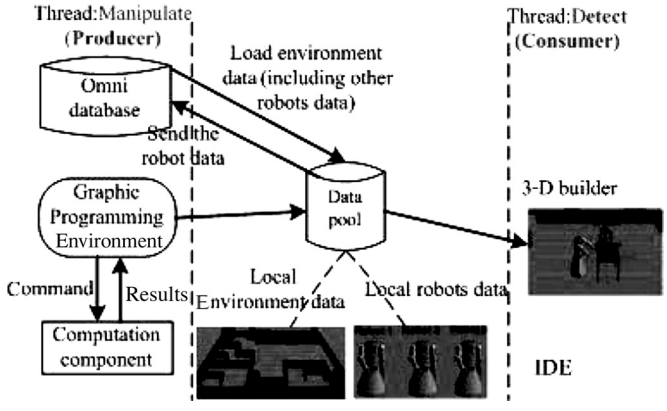



Fig. 9. Multirobot simulation mechanism.

*5.3. Seamless migration between simulated and actual robots*

This system can be applied to both virtual and physical robots. If the program for a new job is satisfied in simulation,

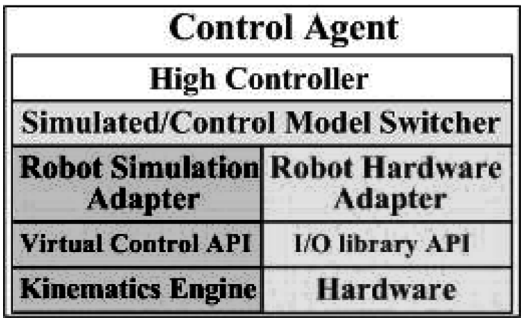



Fig. 10. The architecture of migration between simulated and actual robots.

the code can be downloaded to real robots. The related architecture is shown in Fig. 10.

The control component uses a model switcher to set either a simulated or a real robot control model. If the switcher is in "simulated model," the high controller will use a "robot simulation adapter" to invoke the "virtual controller API" which reads the robot's current status from the "Kinematics Engine,"[19] which is provided by Yaskawa, and return to the graphic programming environment. If the switcher is in "Control Model," the high controller will use a "Robot Hardware Adapter" to invoke the "I/O library API" to control the hardware.

## 6. Experiments and Results

Experiments were performed to compare the system load between the centralized system[2] and the distributed system which is used for controlling actual robots or simulation.

Each computer in the experiments had a Celeron (R) CPU 2.40 GHz and 1230 M usable memory (512 M physical memory and 718 M virtual memory).

First, the centralized system was used to perform the multiple-robot simulation. Figure 11 presents the utilization rate of the CPU (Fig. 11(a)) and the memory (Fig. 11(b)) in the centralized system. The utilization rate of both the resources increased linearly with the number of robots in the simulation. The experiment with four or more robots failed due to lack of memory.

In the distributed system, the MapEditor server and the simulation server were installed on a server computer. Five virtual robot clients were created to work together in a virtual floor environment. In order to show the CORBA-based system's independence, the MapEditor server and simulation server ran on Microsoft Windows XP Professional with Service Pack 2, and the other robot clients ran on Linux (Ubuntu 6.10 kernel 2.6.17-11-generic). Each robot executed its motion list separately and followed a predefined trajectory. Each robot's kinematics status was calculated via the related motion module in each robot client and sent to the simulation server. The simulation is shown in Fig. 12. Because there were no dynamic obstacles predefined in this

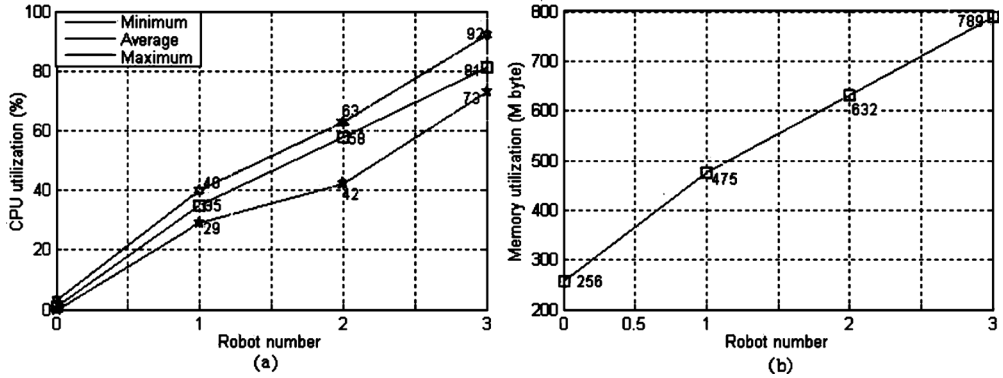



Fig. 11. Centralized system load: (a) CPU utilization, (b) memory utilization.

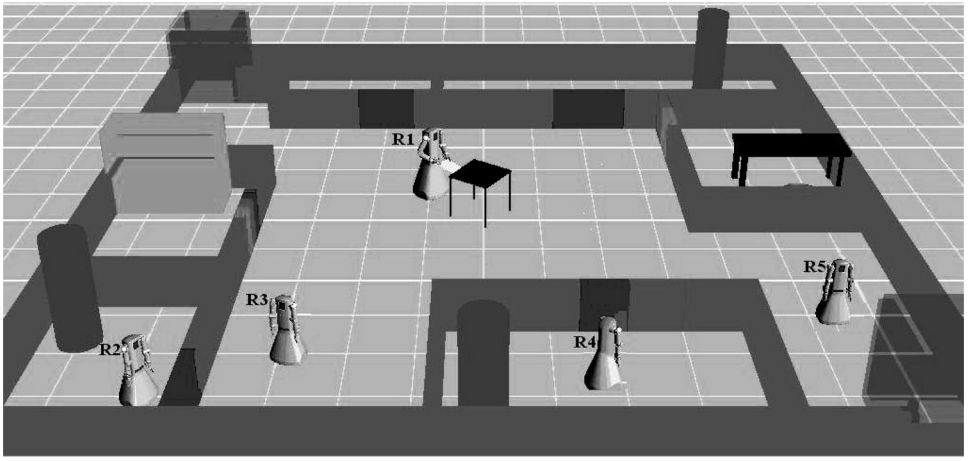



Fig. 12. The scene of the simulation.

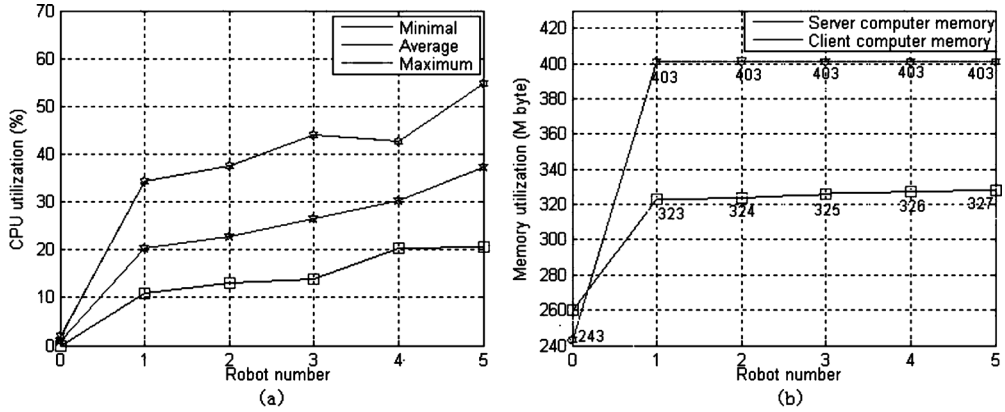



Fig. 13. Distributed system load: (a) server computer CPU utilization, (b) server computer memory utilization.

experiment, the virtual sensors were not used. The computers were interconnected via a fast Ethernet (10 Base T). Figure 13 presents the rate of the CPU and the memory utilization with an increase in the number of connected clients. Both rates of increase in the CPU usage and memory usage of the distributed systems were less than in the case of the centralized system. In this case that three clients connected to the server computer, for a three-robot simulation (Fig. 13(b)), shows that the server computer only spends 63 MB of memory, and the CPU average utilization is 26.58% (Fig. 13(a)). The maximum CPU usage is 43.8% which is less than in the case of the centralized system.

The advantage brought about by the distributed system demanded less CPU and memory usage. However, its shortcoming is that the clients need a longer response time between the start of a method invocation and the arrival of the returned data. Figure 14(a) presents the instantaneous response time recorded over 20 s of simulation. The average response time value increased with the number of robots in the simulation, as shown in Fig. 14(b). In the case where five clients called the service from the server, the maximum of instantaneous response time is less than 100 ms, which meets the requirements of a mobile robot simulation.

Finally, the code of robot I is downloaded to a real robot. It is designed to handle a task of handing over a piece of paper. The code is composed of a sequence of subtasks: like "move to the desk A," "pick up the paper," "turn back," "move to door B," and so on. Figure 15 displays the screens of the simulation and the actual robot. It shows that the proposed system can control the robot SmartPal to finish complex tasks.

## 7. Conclusions

In this paper, a CORBA-based distributed programming system is proposed to realize the control and simulation of multiple mobile robots. In comparison to some non-CORBA systems, such as Player/Stage/Gazebo, Microsoft Robotics Studio, and so on, our system facilitates interoperability of different operating systems (the experiment in Section 5 executed in Windows and Linux OS has demonstrated this feature). In comparison to some CORBA systems, such as ORCA, BREVE, CARMEN, and so on, our system provides the robot's map building, path planning, graphic motion planning, simulation, and actual robot control function. Users can program virtual robots in a customized simulation environment, and check the executing performance; if the simulation result is satisfying, users can download the code to a real robot to execute. The robot key index for configuration is proposed and transferred between different robotic clients, and the robots are built in a virtual environment. This results reduced network load and improved real time performance of the distributed system. What is more, we have implemented our system to Yaskawa's "SmartPal" robot. The simulation and experiment results show that our robotic framework can simulate and control the robot to perform some typical daily tasks such as picking and placing everyday objects (e.g.,

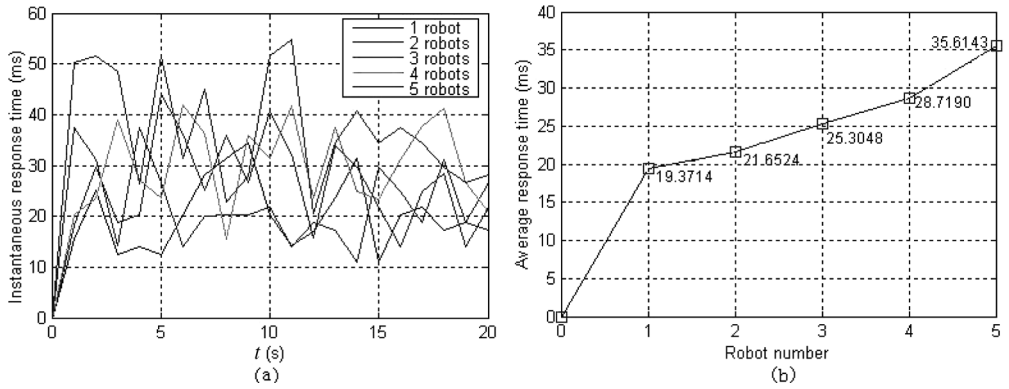



Fig. 14. Response time in distributed system communication: (a) instantaneous response time, (b) average response time.

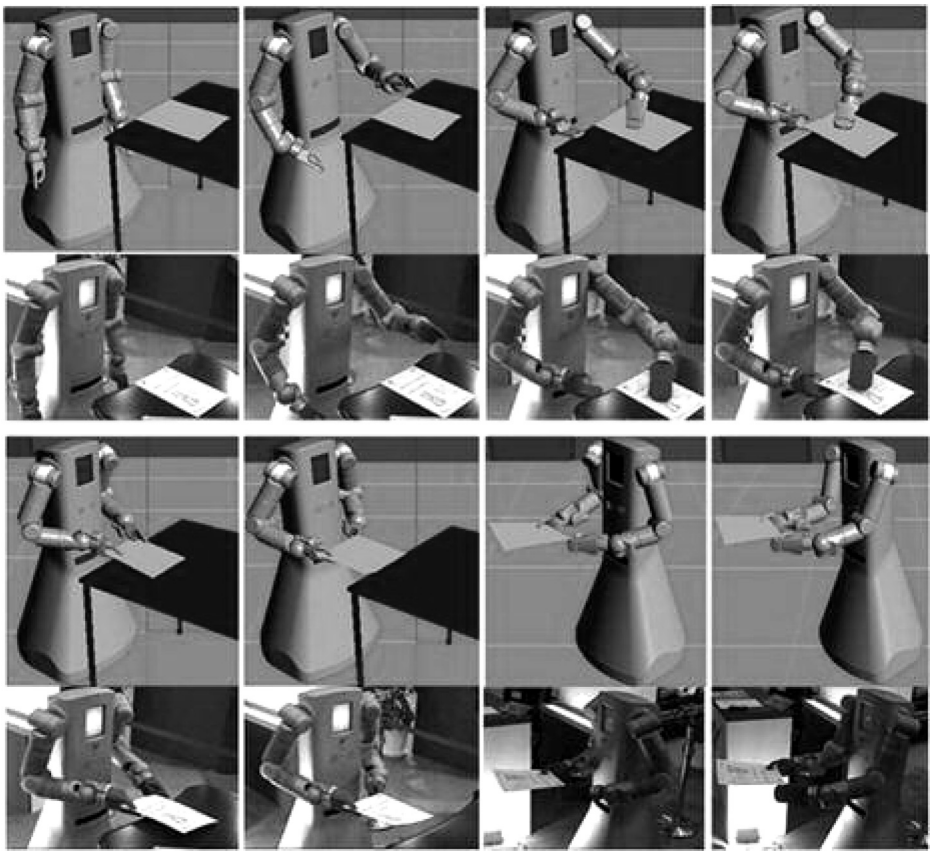


Fig. 15. Seamless migration between a simulated and real robot.

cups, glasses, bottles, plates, and cutlery) as well as operating switches (e.g., light, coffee machine, and cooker) and handles (e.g., doors, drawers, and refrigerators). It is expected that this framework will significantly aid in the development of dual-arm mobile robots in the home environment.

## Acknowledgments

This work was supported in part by the National High Technology Research and Development Program of China under Grants 2006AA04Z261 and 2007AA041703, and supported in part by the National Natural Research under Grant 50705054. The authors gratefully acknowledge the support from Yaskawa Electric Corporation for supporting the collaborative research funds and the SmartPal robot. They also thank Mr. Ikuo Nagamatsu and Mr. Kazuhiko Yokoyama at Yaskawa for their cooperation.

## References

1. K. Matsukuma, H. Handa and K. Yokoyama, "Vision-Based Manipulation System for Autonomous Mobile Robot 'Smartpal"'. *Proceedings of the Japan Robot Association Conference*, Yaskawa Electric Corporation, Japan (Sep. 2004).
2. Qiu Chang-wu, Cao Qi-xin, Ikuo Nagamatsu and Kazuhiko Yokoyama, "Graphical programming and 3-D simulation environment for Robot," *Robot* **27**(5), 436–440 (Sep. 2005).
3. Object Management Group. White paper on benchmarking, Version 1.0, OMG document bench/99-12-01 (1999).
4. M. Henning and S. Vinoski, *Advanced CORBA Programming with C++* (Addison Wesley, Reading MA, 1999).
5. Object Management Group. OMG Robotics Domain Special Interesting Group (DSIG) Homepage. Available: http://robotics.omg.org.
6. M. Mizukawa, H. Matsuka, T. Koyama, T. Inukai, A. Noda, H. Tezuka, Y. Noguchi and N. Otera, "ORiN Open Robot Interface for the Network – The Standard Network Interface for Industrial Robots and its Applications," *International Symposium on Robotics Stockholm* (ISR2002), No.45 (Oct. 2002).
7. M. Mizukawa, H. Matsuka, T. Koyama, T. Inukai, A. Noda, H. Tezuka, Y. Noguchi and N. Otera, "ORiN: Open Robot Interface for the Network – The Standard and Unified Network Interface for Industrial Robot Applications," *SICE Annual Conference*, Osaka (2002), pp. 1160–1163.
8. Orocos: Open Robot Control Software. http://www.orocos.org.
9. C. Schlegel and R. Worz, "The Software Framework SmartSoft for Implementing Sensorimotor Systems," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS '99, Kyongju, Korea (Oct. 1999) pp. 1610–1616.
10. Fumio Ozaki, "Open Robot Controller Architecture (ORCA)," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2004), Workshop on Robot Middleware toward Standards*, Sendai, Japan (Sep. 2004).
11. Fumio Ozaki, "Open Robot Controller Architecture (ORCA)," *Advanced Intelligent Mechatronics (AIM2003) Workshop: Middleware Technology for Open Robot Architecture*, Kobe, Japan (Jul. 2003).
12. Kohtaro Sabe, "Open-R: An Open Architecture for Robot Entertainment," *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM2003) Workshop: Middleware Technology for Open Robot Architecture*, Kobe, Japan (Jul. 2003).
13. J. Klein, "BREVE: A 3-D Environment for the Simulation of Decentralized Systems and Artificial Life," *Proceedings of Artificial Life VIII, 8th International Conference on the*

*Simulation and Synthesis of Living Systems* (MIT Press, 2002) pp. 329–334.
14. M. Montemerlo, N. Roy and S. Thrun, "Perspectives on Standardization in Mobile Robot Programming: The Carnegie Mellon Navigation (CARMEN) Toolkit," *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas (2003) pp. 2436–2441.
15. C. Côté, Y. Brosseau, D. Létourneau, C. Raïevsky and F. Michaud, "Robotic software integration using MARIE," *Int. J. Adv. Robot. Syst.* – (Special Issue on Software Development and Integration in Robotics) **3**(1), 55–60 (2006).
16. B. P. Gerkey, R. T. Vaughan and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," *Proceedings of the International Conference on Advanced Robotics* (ICAR 2003), Coimbra, Portugal (Jun. 30–Jul. 3, 2003) pp. 317–323.
17. B. P. Gerkey, R. T. Vaughan, K. Støy, A. Howard, G. S. Sukhatme and M. J. Mataric, "Most Valuable Player: A Robot Device Server for Distributed Control," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems* (IROS 2001), Wailea, Hawaii (Oct. 29–Nov. 3, 2001) pp. 1226–1231.
18. Sun Microsystems Inc. Java IDL and RMI-IIOP Tools. Available: http://java.sun.com/j2se/1.5.0/docs/tooldocs/index.html#idl (2004).
19. R&D Center Yaskawa Corporation. Instructions for RTLab API (Ver 1.1.2). Yaskawa Robotics Technology R&D Dept (2004).