THE DEVELOPMENT OF R-CUBED MANIPULATION LANGUAGE: A SYSTEM DESIGN FOR SUPPORTING VARIOUS TYPES OF CLIENTS

Dairoku Sekiguchi, Naoki Kawakami, and Susumu Tachi

Information Physics and Computing, Graduate School of Information Science and Technology,
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, JAPAN
+81-3-5841-6917
{dairoku, kawakami, tachi}@star.t.u-tokyo.ac.jp

ABSTRACT

The concept of R-Cubed (Real-Time Remote Robotics: R3) aims to provide a way to telexist anywhere in the world by controlling remote robots over a network. RCML (R-Cubed Manipulation Language) is considered to be a bottom-up approach of the R-Cubed concept. The design of an RCML system utilizes existing infrastructures and devices such as the Internet and PC, and system users can use it easily and intuitively.
RCML is a language for describing the interface for controlling remote robots in an R-Cubed concept. In this paper, we will show a new implementation of the RCML 2.0 system, which supports PDAs and cellular phones as clients in addition to ordinary desktop PCs. We also implemented an experimental system based on RCML 2.0 specifications for the communication device called the RobotPHONE, which uses a bilateral control method.

1. INTRODUCTION

R-Cubed (Real-Time Remote Robotics: R3)[1] is a concept that enables a user to telexist anywhere in the world with the sensation of actually being there. This is accomplished by controlling remote robots over a network. Users of an R-Cubed system feel and act as if they really existed in a remote environment, regardless of the physical limitations of time and space [2].
RCML (R-Cubed Manipulation Language) is considered to be a bottom-up approach of the R-Cubed concept. The design of an RCML system utilizes existing infrastructures and devices such as the Internet and PC, and users of the system are able to use it easily and intuitively. In a manner similar to the way in which a VRML browser provides a standard method for accessing the virtual world, we intend to provide a standard method for accessing the remote real environment with an RCML system.
In this paper, we will show a new system implementation, which can be used with various types of clients, such as PCs, PDAs, cellular phones, etc. Based on RCML 2.0 system specifications, we also implemented an experimental system for the RobotPHONE, a communication device that uses a bilateral control method.

2. RELATED WORK

Currently, network robotics is an active research topic. Many implementation methods have been examined. The simplest implementation is the combination of CGI (Common Gateway Interface) and HTML (HyperText Markup Language) [3]. The implementation that uses a web browser and Java applet is widely used [4][5][6].
Other methods, which use ORB (Object Request Broker), such as CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model) have been studied in an attempt to develop methods that are more general and sophisticated. Hirukawa et al. [7] use CORBA to implement their teleoperation system. ORiN (Open Robot Interface for the Network)[8], which was developed by JARA (the Japan Robot Association), uses DCOM. These ORBs are mechanisms for handling distributed objects and do not define the interfaces between each object. Hence, it is necessary to define a standard method (API) that can adapt to various robots, but it is very difficult to define such a general interface in advance of actual system implementation. Until now, several implementations that use an ORB have been proposed, but a standard method for controlling a remote robot has not been established yet.
With the spread of non-PC devices such as PDAs and cellular phones, it is increasingly important to develop support system for them. However, currently there is little teleoperation system, which supports such non-PC devices. In addition, unlike an ordinary PC, each non-PC device has a wide variety of device configurations. For instance, the CPU speed, memory size, and display size are different

in each device. Therefore, a systematic approach is very important to support various non-PC devices flexibly.

3. RCML 2.0 SYSTEM

The RCML 2.0 system consists of RCML 2.0, which is a language for describing the remote robot, RCTP/2.0 (R-Cubed Transfer Protocol), a protocol for controlling a remote robot and RXID 2.0 (RCML Extensible Interface Definition), a language for defining GUI (Graphical User Interface) [9].

In the RCML 2.0 system, a target robot is described as a set of variables that are necessary for controlling a robot, and the control of a target robot is considered to be equivalent to accessing variables. Figure 1 shows a simple example by two degrees of freedom with a pan/tilt camera. In Figure 1, there are two variables that correspond to each pan/tilt axis, and the camera is controlled by accessing these two variables. At this point, we show a very simple example that has only two variables. When more than two variables need to be managed, it is better to use a tree structure than a flat structure. Thus, the RCML 2.0 system manages variables in a tree structure, which is called an RCML data structure. RCML 2.0 is an XML (eXtensible Markup Language)-based language that uses an XML capability to describe a tree structure.

RXID 2.0 is a new language for supporting various kinds of clients in the RCML 2.0 system. RXID 2.0 supports well-known common GUI elements such as the window, scroll bar, button, and text input and can define the properties for each element, such as position, size, and caption. Hence, a user can easily design various kinds of user interfaces for controlling the remote robot. RXID 2.0, which is an XML-based language as is RCML 2.0, has mechanism for a one-way link to an RCML data structure (Fig.2). This one-way link defines the relationship between a GUI element described in an RXID file and a variable in an RCML data structure defined by an RCML file. By linking these two elements, the input from the GUI side is transferred to an RCML data structure, and the change of variables in the RCML data structure is transferred to the GUI side. Thus, a user can control remote robots by GUI and know the status of the remote robot. Because RXID 2.0's one-way link starts from the RXID file side, it is not necessary to modify the RCML file when describing the RXID file. This mechanism provides for the complete separation of the control of the robot and user interface. Hence, multiple user interfaces for one RCML file can be defined without modifying the RCML file, or one integrated user interface for multiple RCML files can be defined. By using this one-way link mechanism, the RCML 2.0 system can systematically support various kinds of clients, which have wide variety of device configuration.

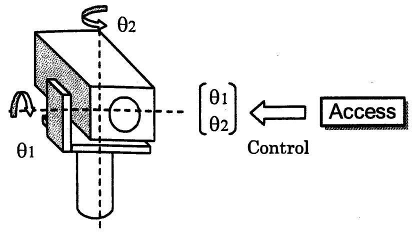

Fig.1 An example using pan/tilt camera

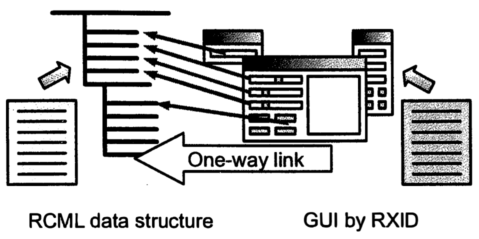

Fig.2 One-way link in RXID 2.0

4. IMPLEMENTATION

4.1. The implementation for PC, PDA and Cellular phone clients

To show that our system design can support various types of client seamlessly, we built three experimental systems for different types of clients. In this experiment, we choose a PDA and a cellular phone as clients in addition to an ordinary PC. We used the same server program and the pan/tilt camera (CANON VC-C3) for each client system (Fig.3).

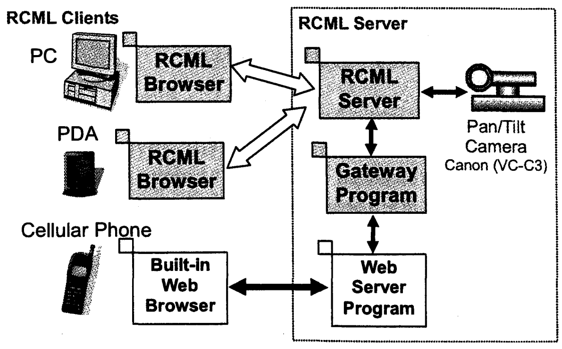

Fig.3 Configuration of the experimental system

4.1.1. PC client system

The system consists of the RCML server, which controls the robot, and the RCML browser, which is used at the client site for operating remote robot. The RCML server consists of the main process, the child processes, which handle each session to an RCML client, and the robot

driver processes, which control the robot. The RCML browser is an independent application that connects to the desired RCML server by typing URL as in an ordinary web browser.
The RCML server and the RCML browser are written by C++ and use "XML for C++ (Version 2.3.1)" as an XML processor. In this experiment, we used Windows 2000 for the server and the client machine.
The RCML browser can dynamically display a GUI panel for controlling remote robots according to the description of an RXID file. Figure 4 shows all the GUI elements that are supported by the current RCML browser. Figure 5 shows the screen-shot image of the RCML browser connected to the remote pan/tilt camera. The images in Fig.6 show a different control panel for the same pan/tilt camera used in Fig.5. These control panels can be displayed by simply changing the RXID file.

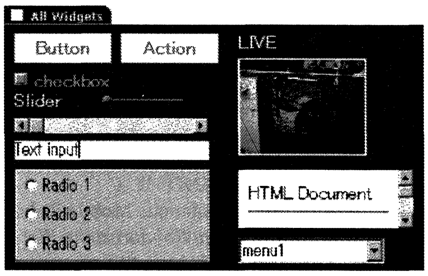

Fig.4 All the GUI elements supported by the browser

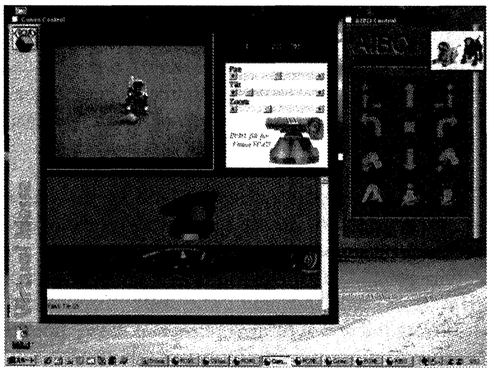
Fig.5 Screen-shot images of the RCML browser

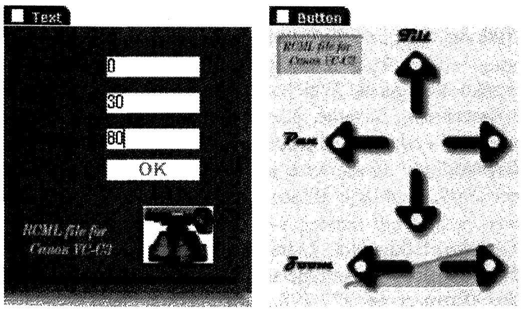

Fig.6 The control panels that have different design

4.1.2. PDA client system
We used the Windows CE 3.0 platform (Pocket PC made by Hewlett-Packard, CPU: SH-3 133MHz, Memory 32MB) and the Personal Java runtime environment for the Windows CE. The RCML browser designed for a PDA is written by Java. This RCML browser uses JAXP Ver.1.0 XML processor and has almost the same function with the RCML browser on a PC. By loading a different RXID file, this RCML browser can display a different GUI in the same way as the RCML browser running on PC.
The PDA used with this experiment had a small display area (240 x 320), and the display aspect ratio was also different from the PC's display. Therefore, we prepared the user interface, which optimized for the PDA by creating a new RXID file. Figure 7 shows the PDA screen image that is displayed when controlling the remote pan/tilt camera.

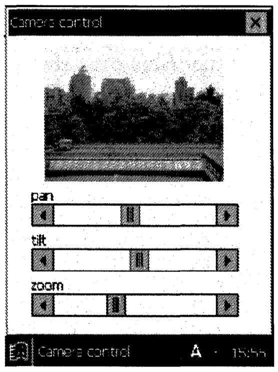

Fig.7 Screen-shot image of the PDA

4.1.3. Cellular phone client system
This system supports a cellular phone that has a built-in HTML browser. The gateway program and the web server are placed between the RCML server and the cellular phone. This gateway program uses a JAXP Ver.1.0 XML processor and dynamically produces HTML files from an RCML file and an RXID file. Therefore, by changing the settings on the gateway program, this system can be used with various robots and clients such as the RCML browser and can adapt to various robots and clients by changing the URL. Figure 8 shows the image that is displayed on a cellular phone when controlling the remote pan/tilt camera.

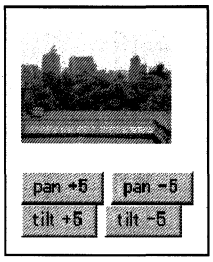

Fig.8 Screen-shot image of a cellular phone

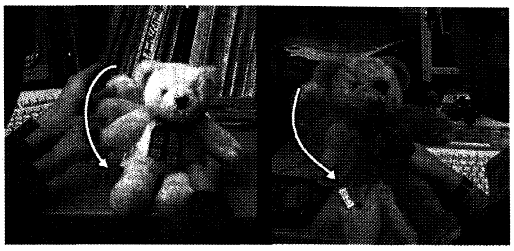
Fig. 9 Movements of the teddy bear-like RobotPHONE

4.2. The implementation for RobotPHONE

To show that our system design can support various types of device seamlessly, we also built an experimental system for RobotPHONE.

The RobotPHONE is a Robotic User Interface (RUI) that uses robots as physical avatars for interpersonal communication [10]. Using the RobotPHONE, users in remote locations can communicate shapes and motions with each other (Fig.9). The RobotPHONE system uses robots that are called shape-sharing device. The shape and motion of remote shape-sharing devices are always synchronized by the symmetric bilateral control method.

The RobotPHONE system is a completely symmetrical system and there is no distinction of server and client. On the other hand, the design of the RCML 2.0 system is basically based on server client models. Therefore, to build an RCML 2.0 system for the RobotPHONE, we introduced a module called an RCML coordinator to the system (Fig.10).

As shown in Fig.10, both ends of the system are RCML servers. The RCML coordinator acts as an RCML client for both RCML servers and mediates between the two servers. The first negotiation between the two RCML servers is done through the RCML coordinator. However, the control data is transferred directly by an RCTP/2.0 data stream, which is set up by an RCML coordinator between the two servers.

The control cycle of the system was 21[ms], and the bilateral control was performed the same way when two controllers were connected directly.

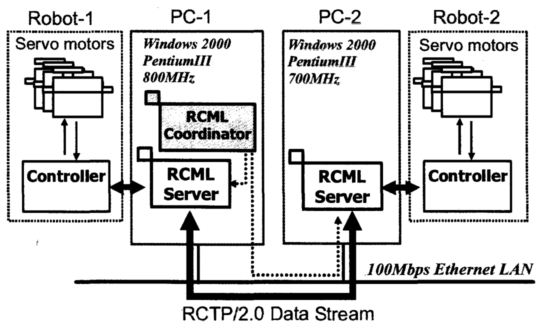

Fig.10 The experimental system for the RobotPHONE

5. CONCLUSION

In this paper, we showed a system design for the support of various types of clients. By introducing RXID 2.0 into the system, GUI and the control information of the robot were completely separated, and the systematic support for various types of client was archived. We implemented an experimental system for a PC, a PDA, a cellular phone, and a RobotPHONE and showed that our system design can adapt to various types of device.

6. ACKNOWLEDGMENTS

The part of this work has been supported by CREST of JST(Japan Science and Technology Corporation).

7. REFERENCES

[1] MITI of Japan, R-Cubed WG ed.: *"R-Cubed"*, Nikkan Kogyo Shinbun, (1996).

[2] S. Tachi: "Real-time Remote Robotics - Toward Networked Telexistence", *IEEE Computer Graphics and Applications*, pp. 6-9, (1998).

[3] R. Simmons: "Xavier: An Autonomous Mobile Robot on The Web", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 43-47, (1998).

[4] M. R. Stein: "Painting on the World Wide Web: The PumaPaint Project", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 37-42, (1998).

[5] Roland Siegwart, et al.: "Guiding Mobile Robots through the Web", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 1-6, (1998).

[6] P. Saucy, F. Mondada: "KhepOnTheWeb: One Year of Access to a Mobile Robot on the Internet", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 23-29, (1998).

[7] H. Hirukawa and I. Hara: "The Web Top Robotics", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 49-54, (1998).

[8] Mizukawa,M., Matsuka,H., Koyama,T., Matsumoto,A.: "De-facto standard API for Open and Networked Industrial Robots", *Proc. 30th Int. Symp. on Robotics*, pp.455-462, Oct. 1999

[9] Dairoku Sekiguchi, Weichung Teng, Yasuyuki Yanagida, Naoki Kawakami, Susumu Tachi: Development of R-Cubed Manipulation Language - The design of RCML 2.0 system, *Proc. 10th International Conference on Artificial Reality and Tele-existence 2000*, pp. 44-51, 2000

[10] Dairoku Sekiguchi, Masahiko Inami, Ssusumu Tachi: RobotPHONE: RUI for Interpersonal Communication, *CHI2001 Extended Abstracts*, pp. 277-278, 2001.