Proceedings of the 13th
KACC, October 1998

Development of GUI for Industrial Robot Systems

Seong Ho Lee and Jae Wook Jeon
School of Electrical and Computer Engineering
SungKyunKwan University
Suwon 440-746, Korea
Tel: +82-331-290-7237
Fax: +82-331-290-7231
Email: lsh@ece.skku.ac.kr, jwjeon@yurim.skku.ac.kr

Abstract

This paper proposes a graphical user interface for industrial robot systems. Previous user interfaces for industrial robot systems were based on the text. In order to enable operators to handle robots more efficiently, a set of graphical tools is provided. The graphical tools contain a control panel for operating robots and compiling robot programs, a graphical teaching panel for handling virtual robots and a graphical monitoring panel for checking robot status. Furthermore, the proposed GUI can be used to operate remote robots because it has network utilities.

This system consists of the virtual mode and the real mode. The user can handle a 3D virtual solid model of the robot in the virtual mode and an actual robot in the real mode.

1. Introduction

Nowadays, as computers have been more widely used from home to industry, wherever these have been used, a user environment becomes one of the most important factors for these. Despite that some systems have good performances, operators have turned away their face due to difficult user interfaces. In particular, a distributed computing environment is on the rise so that each computer in industry is allowed to share resources and is allocated tasks through the network. Robot systems that execute a variety of operations have to easy user interfaces so that they are efficient to handle. Nevertheless, because previous user interfaces of robots did not offer such environments, the unexperienced operator had to spend a lot of time to deal with robots. Graphical user environments or visual robot program tools for robot systems have been developed and these proposed a set of convenient graphical tools to the operator [1-3]. However, industrial robot systems which use teaching pendants have text-based user interfaces, and offer little extensibility for a distributed controlling environment because of the exclusive cable between them. The primary purpose of this paper is on the development of the easy and reliable user environment for robot systems.

We have designed and implemented a virtual robot model for a simulation, device driver, application program interfaces (API's). Since hardware resources of robots are complicated and dedicated, higher level needs a hardware abstraction in order to access hardware resources easily. Therefore, a layered and modular access to interface hardware resources is used, Also it is working on event-driven basis in order to fulfill real-time objectives. A client-server approach provides a remote interface between the user and robot's resources and a connection to interplay between lower level to handle its resources and higher level to write application programs.

This paper is structured as follows. Section 2 describes the system architecture of the robot, especially the software framework. Section 3 introduces GUI environment for controlling robots. In section 4, this is applied to operate an actual robot. Section 5 presents a conclusion and future work.

2. System Architecture

Overview

A general robot system is implemented as illustrated in Fig. 1. In this paper, GUI for robot systems is developed on the X window and the network [4]. This enables the operator to conveniently control though network. Essentially, GUI must provide the user with an intuitive and easy environment to deal with robots and monitor a response to hold a firm belief for activity done by the user. Therefore, this GUI offers a mechanism, a virtual technique to consider a inevitable delay occurred through network when the operator puts a command into robots.

A modular and layered software framework from

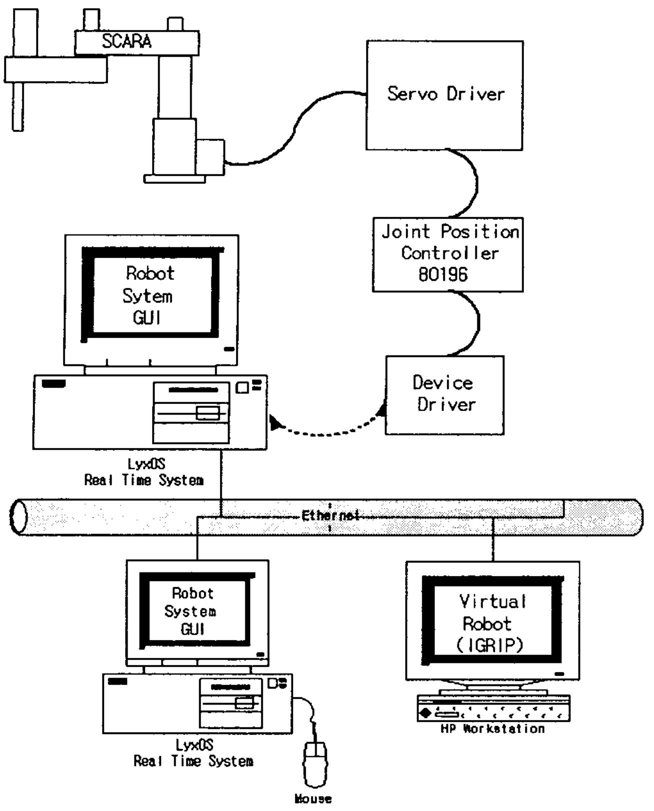

Fig. 1. A proposed robot system.

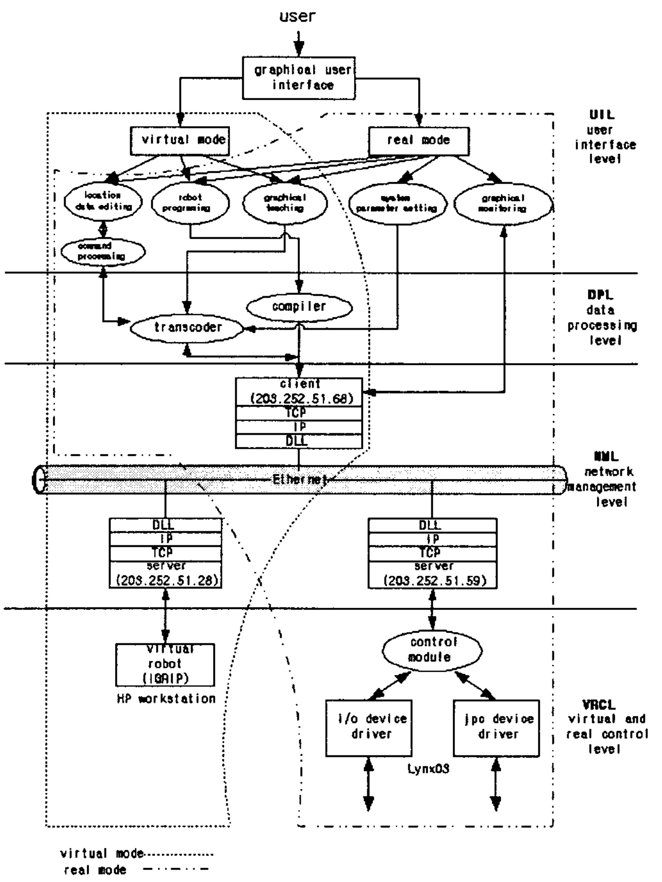

Fig. 2. Details of Software Framework for General GUI.

accepting user inputs to accessing robot's resources, and vice versa, is adapted to offer a successful paradigm in modern software engineering; portability, reusability, maintainability and so on [5].

Software framework

This modular and layered software framework consists of four levels. Each level gives and takes regular data with one another. User interface level is located in the highest level to accept user information and output robot status. We will call this UIL (user interface level). Data accepted from robot's resources to user inputs, vice versa, are transformed to a regular type in the next position. We will name this DPL (data processing level). Also, NML (network management level) located in the third position transfers specific data from/to the lowest level, VRCL (virtual and real control level) have control modules - device drivers, API's, real-time control routines - in real environments. Its intention is to raise a degree of efficiency as run parallel with one another. For example, the operator can compile a robot program for drawing up location data and simultaneously supervise robot status with a set of graphical tools. However, since this robot uses an encoder as a sensor for checking position, this need to return to zero. This means that the user not puts a command into the robot until a servo is turned on and a robot is returned to zero. GUI must consider the point; if the sequence for the robot is not accomplished, GUI has to generate suitable error messages for that situation. It is as follows for details of software framework in Fig. 2.

User interface level: This provides the user with various dialog boxes for input and a graphical monitor for output. Because this part is shown to the user as begin this GUI, the design of this lays emphasis on visual factors as well as functional factors. That is, this part supplies the user with an intuitive and easy environment.

UIL consists of seven windows as shown in Fig. 3, that is, those are windows for setting system parameters, editing location data, putting robot commands, robot programming, a graphical teach, and a graphical monitor. Each window is used for control robots in real and/or virtual environment and has sub-windows to process various data from the user and the robot.

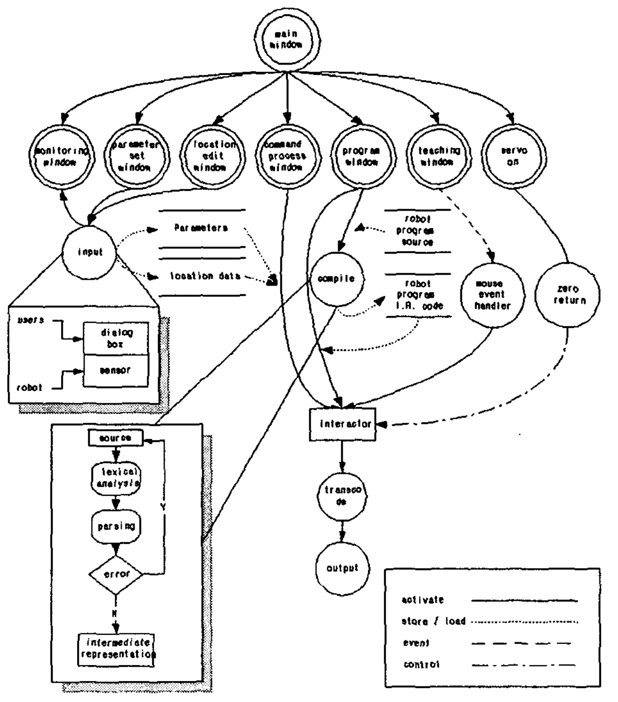

Fig. 3. Interaction between UIL and DPL.

Data processing level: We can consider UIL HCI (human computer interaction), while regard DPL as a kind of data processor. This level can be largely divided into two parts, transcoder and compiler. First, the transcoder transforms raw data from UIL into regular short messages, called packets, appropriate to NML or vice versa. Second, the compiler generates an intermediate code from a source code, and the intermediate code is a kind of protocol used in NML.

One of the most prominent methods for handling robots is to use a robot language. In practice, most robots in a manufacturing plant execute tasks by robot programs. Furthermore, the programming method for handling robots makes up for the weak points in current a set of graphical tools. That is, it supports more detailed instructions for achievement of continuous and/or complex tasks. First of all, development of a robot language translator must go ahead to define the grammar of the robot language. Words that will be used by the compiler have to be defined and grammatical relationship of these established. The compiler consists of two parts, the lexical analyzer and the parser. That looks at the input stream as a collection of basic language elements called tokens. Where, a token is an indivisible lexical unit. This analyzes a sentence; to parse a sentence is to break it up into its component parts in order to analyze it grammatically. Therefore, the source code of a robot program is analyzed and translated into an object code by the compiler, without errors. The object code is the same the output of the transcoder and transferred robot servers through NML (Fig. 3.).

In Fig. 3, Interaction of between UIL and DPL is illustrated. The main window consisted of seven sub-windows forms general UIL. Each sub-window is executed sequentially or concurrently and exchanges data with DPL. Two types of data from the user and the robot are stored in data structures or files and translated into a suitable type by the transcoder and the compiler. The user activates this data in sub-windows and this activated data will wait a sequence - servo on, zero return - in the interactor.

Network management level: In this GUI, data from UIL are arrived at NML via DPL. As pass through DPL, user 's data are transformed into regular packets to transfer from NML to VRCL. Thus, NML is a coupling device of clients and servers. Where, UIL and DPL parts are clients and VRCL part is servers. NML is based on the socket on TCP/IP and connection-oriented service. Details of NML are illustrated in Fig. 4.

In NML, the client API will look at whether the type of packets from a client is reasonable or not and if reasonable, transfer packets into servers and start a timer for waiting a reply. On the other side, the server API will look at the kind of a packet received. If the kind of the packet is not the same that of the packet had been received, the packet is transmitted to VRCL. After that, other clients not send another packets until an acknowledgment receive from a server. That is, during this time, the client is blocked and waits for a reply of the server. Events arriving at the hardware interface and/or the virtual robot cause the notification of the server API. The server API, notified about events from servers, then replies to a waiting client. Because it is send blocked, the client does not consume computation time while waiting for a reply.

The server of the hardware interface has two socket ports. One is used for controlling a robot, and the other for checking the robot status. Thus, while the client transmits a packet to the server, at the same time, the client will not miss any sensor data, as long as its cycle time is smaller than the sensor frequency.

Virtual and real control level: This level implements the logical link to the sensor or actuator. VRCL becomes a target of a client and has mechanisms to access virtual and real resources. It uses mechanisms provided by the real-time operating systems [6]. Also, the level hides the sensor and

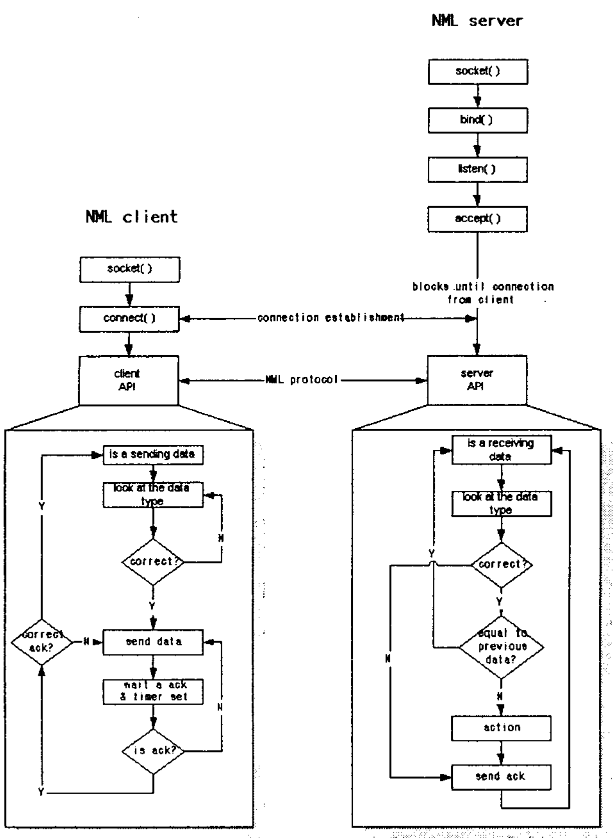

Fig. 4. Internal Structure of NML.

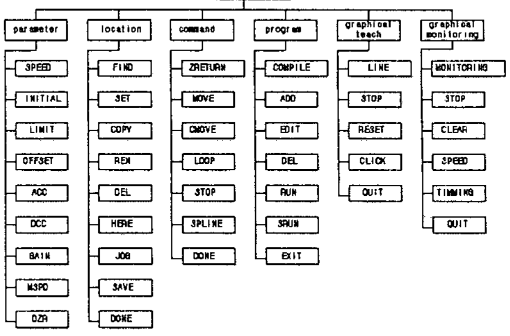

Fig. 5. Menus of GUI.

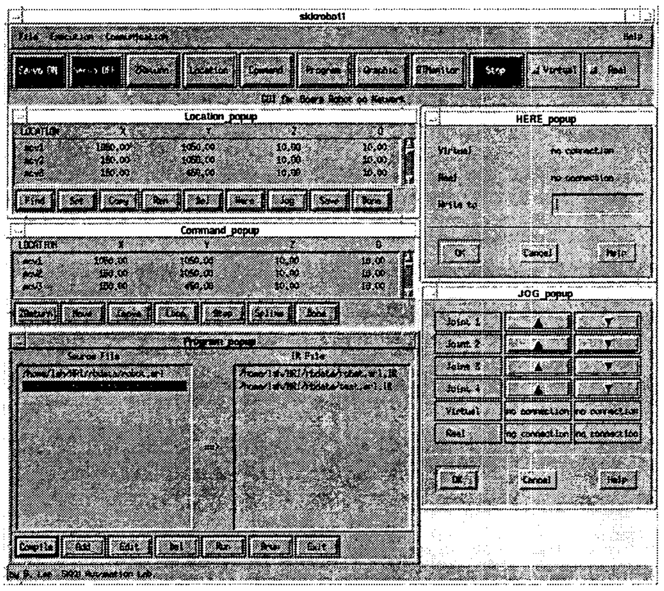

Fig. 6. The beginning picture and some windows.

actuator access mechanisms, e.g., hardware ports address, by using device drivers [7].

3. Graphical User Interface

Appearances and Functions

Fig. 5 shows whole GUI menus for a robot system developed in this study. A pull-down menu is located in the above part of the main window. Whole menus consist of six sub-windows; for setting parameters, editing location data, putting commands, programming, graphical teaching, and graphical monitoring. Each sub-window has one function or another sub-window, which essentially, processes data from the operator. Also, sub-windows except the window for setting parameters and graphical monitoring can interconnect the virtual and/or the real robot through network. Fig. 6 illustrates the beginning picture and some windows. Considering the function of each window is as follows.

Parameter setting window: The operator will be able to set parameter values in this. That is, speed, initialization, limit value, offset value, acceleration, deceleration, gain value, and arm type.

Location data editing window: The operator will be able to create and edit location data with world coordinate or joint coordinate values. Location data created are conveniently managed by using some keys, FIND, SET, COPY, REN, DEL, HERE, and JOG. Where, HERE and JOG keys are able to read the current position of a robot and this location data are stored in a file.

Command window: This window reads a file created in the window for editing location data and displays data in this. The user will manipulate robots by using MOVE key for PTP (point to point) motion or CMOVE key for CP (continuous point) motion with this data and LOOP key will be used for repeated motion.

Program window: In this window, the user will edit robot programs with the grammar defined by the compiler. The robot program is compiled and transmitted to robot servers through network. Where, the user can utilize Table. 1.

Table 1. A robot language

Command words	zr	Zero return
	move	PTP (point to point) motion
	cmove	CP (continuous point) motion
	delay	Delay time
Statements	if else	Decisions
	for	Loops
	stop	Stop executing
	by	Speed
Data type	jloc	Joint coordinate
	cloc	Cartesian coordinate
	int	Integer
	float	Real
Operators	+ - * / ++ --	

Graphical teaching window: The user will click the mouse to move 2D robots drawn on the x-y plane with first axis and second axis and on the z plane with third axis. If so, the user can move virtual and/or real robots to a desired position by only drifting the mouse. At the same time, the 2D robot is moved as its inverse kinematics. Fig. 7 shows the graphical teaching window.

Graphical monitoring window: This enables the user to supervise a robot status because it illustrates the velocity profile of each axis graphically.

Virtual Robot

It is difficult that the user operate a remote robot through network with his eyes close. Thus, to control the remote robot, virtual robot environment is necessary. In the virtual mode, the user can simulate the 3D robot and manipulate more accurately the real robot. Also, as choose both the virtual mode and the real mode, the user can operate the remote robot as reality seeing the virtual robot.

4. Application

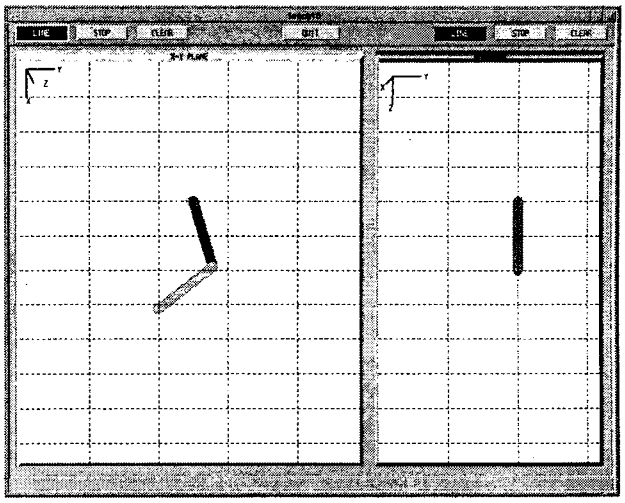
Fig. 7. Graphical teaching window.

In order to test the developed GUI, communication and control architecture the SCARA robot has been used in this paper. Applied systems consist of SCARA Robot (Samsung FARA SM2), servo driver (Samsung SSD) and two joint position controllers. We used PC (Pentium 166Mhz) as the client computer and the server computer for the real robot, LynxOS, UNIX compatible real-time system, as operating systems for these and IGRIP on HP 715/33 as the virtual robot.

Fig. 8 shows the velocity profile of each axis as a robot is being executed by program method.

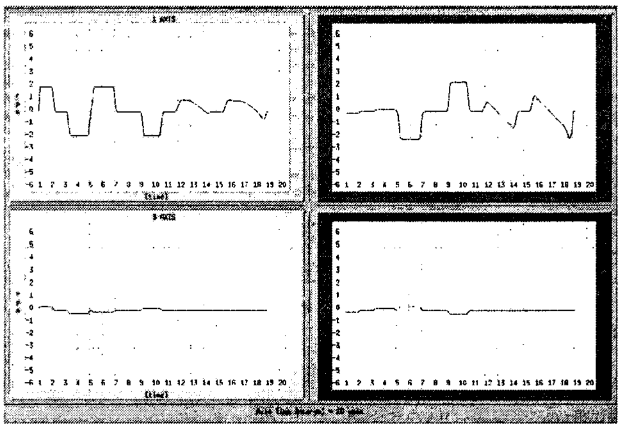
Fig. 8. Graphical monitoring.

5. Conclusions

We have operated and monitored robots on this GUI. It is based on a client-server communication. The introduction of this GUI provides the user with convenient method for controlling robots through network. Thus, in industry, users can reduce much time to be educated so that they can operate robots.

This GUI system has been implemented successfully for our test robot. It is very flexible and will be adapted to a broad variety of networks. Furthermore, the presented GUI system will support adaptability to new hardware and portability to other related systems, e.g., mobile robots.

6. References

[1]. Detlef Zuhlke, Frank Mobius and Christoph Schroder, "Symbols facilitate programming of industrial robots", *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3037-3042, April 1997.

[2]. Matthew W. Gertz, David B. Steward, and Pradeep K. Khosla, "A Software architecture human-machine interface for reconfigurable sensor-based control systems", *Proceedings of 8th IEEE International Symposium on Intelligent Control,* Chicago, Aug. 25-26, 1993.

[3]. Miguel Rodriguez, Alain Codourey, "Graphical User Interface to manipulate objects in the micro world with a high precision robot", *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3031-3036, April 1997.

[4]. Douglas A. Young, *The X Window System Programming and Application With Xt,* Prentice-Hall, Inc, 1994.

[5]. Ian Sommerville, *Software Engineering,* Addison-Wesley, 1996.

[6]. *LynxOS Application Writer's Guide,* Lynx Real-Time Systems Inc., Los Gatos, California, 1995.

[7]. *Device Driver Development Guide,* Lynx Real-Time Systems Inc., Los Gatos, California, 1995.