Robotica (2009) volume 27, pp. 403–409. © 2008 Cambridge University Press
doi:10.1017/S0263574708004815 Printed in the United Kingdom

Integration of a plug-and-play desktop robotic system

Jianbo Su* and Yanjun Zhang

Department of Automation, Shanghai Jiao Tong University, Shanghai 200240, China

(Received in Final Form: May 29, 2008. First published online: July 9, 2008)

SUMMARY

This paper describes the development of a desktop robotic system that enables the plug-and-play function through the USB (universal serial bus) port of a personal computer (PC). Thus a new kind of desktop PC peripheral is invented that has programmable manipulability. The robotic system is realized on an internally distributed control structure that facilitates higher system reliability. A PID control algorithm is implemented on a prototype of the proposed system, to demonstrate the system's ability to implement feedback control. Experimental results show the performance and properties of the proposed system.

KEYWORDS: Plug-and-play; USB; Distributed digital control; Robot arm.

1. Introduction

Robot arms have become an indispensable means of automation to increase productivity and flexibility of manufacturing systems. Currently, there are more than 800,000 robot arms working in factories all over the world.[1] In the future, robot arms may act as family members or a part of office facilities. They can be put on desks and serve as a necessary appliance with manipulability. Such robot arms shall be affordable for average people. Practically, it must be easy to set up, e.g., in the plug-and-play way as a peripheral of a PC, and can be used and maintained without specially knowledge in robotics. In addition, it should be of potential to execute complex tasks with the help of other equipments, such as a digital camera. Thus, efficient sensing ability and communications with its host computer are critical for its real-time control.

Although there have been significant progresses in developing robot arms over the past four decades, those robot arms are still only prevalent in industrial areas and not yet common in our daily lives. This is mainly due to the difficulties in achieving convenient setup and low cost. For example, the widely used robot arm PUMA 560 is driven by six DC motors with incremental encoders. It is 13.2 kg in weight with a separate controller box of 36.4 kg, making it difficult to be moved frequently. In addition, the robot controller is realized with a special digital control board and connected to the PC through one cable.[2] And the robot operations and human–robot interactions are complex. So, it is impossible for anyone without special knowledge in robotics to handle it. This is also true for other small industrial robots, like YASKAWA and ADEPT. Moreover, their prices are at least 10 times higher than that of a PC. In 1990s, Lynxmotion Inc. developed a series of simple desktop robot arms, Lynx robot arm, which is driven by several servomotors and controlled by an 8-bit MCU.[3] However, the control module of each joint is fixed in its servo motor, and the control parameters cannot be modified for dynamic environments. So, the accuracy for position control is limited. Later, Intelitek Inc. produced another series of desktop robot arms named Scrobot-ER. Typically, it consists of six DC servomotors and is controlled with a 16-bit MCU in which a PID controller is embedded.[4] However, both these two arms are accessed from PC through serial port, and their data transfer rates are not high enough to enable visual feedback. More seriously, it adopts interrupt request (IRQ) way to communicate with the PC, which is easy to cause conflicts with the robot system.[5]

Recent development in universal serial bus (USB) has revolutionized the connection mode between PC and other systems. As a serial/parallel port, USB interface has become one of the standard I/O interfaces for PC. It supports dynamic attachment and removal, and this type of connection will never cause IRQ conflict on the PC. More attractively, its transfer rates are up to 480 Mbit/s, which is much higher than that of other I/O interfaces.[6] So it can support large and frequent data transmission. So far, the USB interface has been popular in PC peripheral designs, such as the new generation Lego Mindstorms NXT, which integrates USB 2.0 interface for data transmission with PC. However, it is not for real-time robot control since Lego robot is not designed as the PC peripheral.[5] For the robot systems like Lego, PC is more like an interface than a controller.

Another popular robot system is the Sensable Phantom Robot system, which uses the parallel port to connect with the PC.[8] It has the plug-and-play function to PC via IEEE 1394 way. However, average PCs do not support IEEE 1394, which means an additional board is required for communications that increases the system expenses and complexity.

In this paper, we present a desktop robot arm (called Rinar), which has combined advantages of different designs given in refs. [2–4]. By focusing on the universality and low cost, several practical aspects of the design are discussed in detail. Table I gives the comparisons among some available similar robot systems. Specifically, we will emphasize on: (1) flexible data transmission approach; (2) simple but applicable control strategy for motion control; and (3) distributed control structure of the whole manipulator. In particular, our distributed control structure is different from others.[2] As in ref. [2], each controller uses a DSP for motor control,

* Corresponding author. E-mail: jbsu@sjtu.edu.cn

https://doi.org/10.1017/S0263574708004815 Published online by Cambridge University Press

Table I. Comparisons of different robot systems.

Robot arm	Weight (kg)	Cost	Interface	Major applications
Lego NXT	<1	Low	USB, Bluetooth	Education, Small research lab
Sensable Phantom Robot	>4.5	High	IEEE 1394 (Omni version)	Research, 3D Modeling, Medical
Rinar	1.5	Low	USB	Home & Education

communicating with the host computer through a controller area network (CAN) bus. This fashion improves controller bandwidth and facilitates software control after being applied to a PUMA robot arm. However, CAN bus is not easy to handle for this type of intelligent PC peripherals since the control of the robot arm requires the computation power of its host PC as well as other resources to a great extent. Hence, we adopt a simple distributed control structure for our robot arm, which utilizes a microprocessor control unit (MCU) to serve as the control center so as to ease the burden of the host PC, while several independent PID control modules are applied to implement motors' control. In addition, versatility, stability, and cost of system are always taken into consideration when designing the whole system.

The rest of the paper is organized as follows. Section 2 describes the architecture of the proposed system in detail. Section 3 addresses the system attributes on plug-and-play and internal distributed control system. Experiment results are discussed in Section 4, followed by conclusion in Section 5.

2. System Architecture

2.1. Hardware architecture

A desktop robotic manipulator is mainly composed of three parts: robot body, internal distributed control system, and USB interface module (Fig. 1). The USB interface module, which is an intermediary between the PC and the robot arm, consists of a USB interface device and an MCU. This module can translate the data between the USB format and the PC format according to some prescribed data transmission regulations. Also, the MCU connects to the internal distributed control system. This system has several motion control modules for the joint movements, each of which includes a DC motor with incremental encoder, a motor driver, and a motion controller. As a closed-loop control system, each encoder's signals are sent to the motion controller as motors run. The motion controller accomplishes the calculation and sends commands to motors based on the feedback.

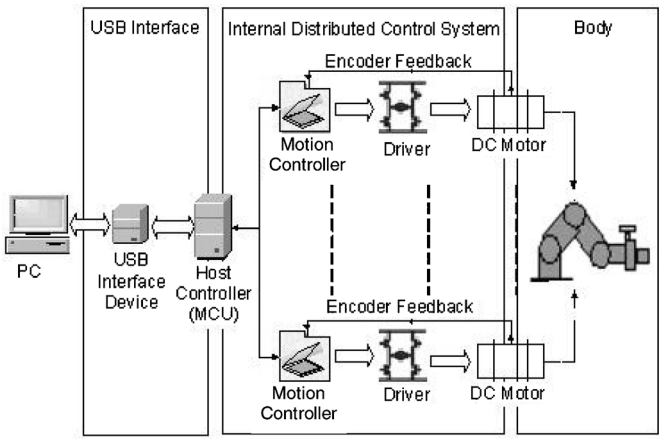


Fig. 1. Hardware structure of the desktop robot.

2.1.1. USB interface module. As a desktop appliance, robot arm's interface with PC is critical to the whole system. The RS232 interface is worthy of noting since it has been the most widespread interface today. Unfortunately, its data transmission speed is too slow for the desktop robot we are developing. Normally, its maximum speed is only 19.2 kbits/s, which means it would take 5 min to read a digitized image of 12 bits with the size of 5.9 million pixels. As a matter of fact, serial transmission technique has now been developed to allow a rate of several megabits per second, such as RS422, Ethernet, and IEEE-1394 (or FireWire, used for the transfer of digital video signals).[9] Transfer speed of those interfaces is ideal for extensive data transmissions. However, few PCs support these interfaces.

USB is specified to be an industry-standard extension to the PC architecture with a focus on PC peripherals that enable consumption and business applications. It is an ideal plug-in connection for all peripheral devices, with a high-speed data transfer rate up to 480 Mbit/s (USB2.0 specification).[10] Taking both connection convenience and data transmission speed into consideration, it is an ideal interface for the desktop robot arm.

With the use of USB, the robot arm can be attached to and removed from a PC dynamically. This feature is significant for a robot arm to be widely accepted as a desktop appliance. By introducing USB into robot arm, not only can the robot arm be controlled by most PCs, but also it would be possible for robot arm to achieve real-time closed-loop control based on sensor information, such as force and visual feedback, thanks to its capability of high-speed data transmission. Hence, performance of the robot arm can be enhanced greatly over that of the traditional ones.[11] Moreover, such a robot manipulator can be mounted to a mobile robot or humanoid robot as a part of its body, which simplifies the mechanical structure design of a more sophisticated robot system. All parts of a robot system can first be designed independently, and then connected with each other, manually or automatically, according to task requirements via USB ports very conveniently.

Therefore, the following properties of USB are essential for the development of proposed desktop robot system:

https://doi.org/10.1017/S0263574708004815 Published online by Cambridge University Press

- Ease-of-use for PC peripheral expansion;
- Full support for real-time audio and video sensing and real-time control;
- Protocol flexibility for mixed-mode isochronous data-transfers and asynchronous messaging;
- Provision of a standard interface capable of quick diffusion into product;
- Integration in commodity device technology;
- Comprehension of various PC configurations;
- Enabling new classes of devices that augment the PC's capability.

There are two types of structure to realize the USB connection for a PC peripheral device. The typical method is to adopt an integrated chip, which includes a processor and a USB interface device. The advantage of this structure is that it can simplify the hardware design and software development. The robot arm we are developing generally need high flexibility and may extent its function in the future. Such type of structure is ideal for this application. Of course, we note that since the processor and the interface device have been built together in this structure, the integrated chip needs to be replaced altogether once either the USB interface device or the processor is updated.

2.1.2. Internal distributed control system. Most robot control schemes have been focused on the centralized control approach. However, robot is now being used for much more complex tasks in more complicated environments, which typically requires massive data exchanges and processing. Hence, the centralized control scheme may not be appropriate any more due to the computational cost and lack of flexibility.[12] It is necessary to balance the computational burden among different processors so as to achieve higher performance of real time control and flexibility of the robotic system. The proposed distributed control architecture is also shown in Fig. 1.

The system processors should have the following functions: data exchanges between the robot system and its host PC, motion control of the robot joints, path planning, and external sensor-based control. We realize these functions in a distributed modularity fashion, allowing the system to be more robust to failures of individual modules. A general embedded distributed control system is composed of a host controller and several control modules with similar functions. Each module takes charge of the control of a joint of the robot, which means every joint of the robot can be controlled and actuated alone. This structure greatly reduces the workload on the host controller. Besides, if one of the modules runs out of order, it does not affect any other modules and the rest of the system may keep on working that leads to higher reliability for the whole system.[13]

For each joint of the robot arm, as shown in Fig. 2, the reference position and speed commands are sent from the main controller to its motion controller. An encoder at each joint motor verifies the speed. Each motor is controlled in the closed-loop form since the motion controller has a direct feedback from the motor.[14]

One of the most important tasks for the joint control of the robot is to configure the controller's parameters. The controller's parameters could be optimized according to the posture of the arm. Obviously, such an optimization process would induce extra burden to the joint controller. Hence, we directly introduce a parameter table stored in MCU. The table contains the optimized parameters of controllers for different joint postures of all motors. When the robot arm is running, control software would select appropriate parameters for each motor according to its rotating angle. Practically, this parameter table is obtained based on a series of experiments.

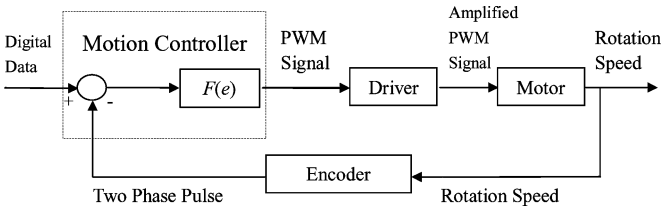


Fig. 2. Control module for each joint of the desktop robot.

2.1.3. Mechanism. The desktop robot should be as simple as possible in its mechanical structure, as well as control mode. Hence, a desktop robot system that has four rotary DOFs, i.e., shoulder, elbow, wrist, and a grabbing tool, is designed (Fig. 3). This configuration is practically sufficient for most of the desktop applications.

All the driving moments of joints come from the DC motors through gear transmission since gearing can provide bigger torque than motor itself.[15] Meanwhile, a plane bearing is used on the waist joint. From Fig. 3, it is easy to see that the workspace of the desktop robot could cover most of the range around the robot.[16]

2.2. Software architecture

2.2.1. Software architecture of MCU. Figure 4 illustrates the software structure in the MCU side. After the start, the USB interface device as well as the chip parameters and device descriptions are initialized. This information will be collected by the host PC's operating system to recognize what type of device is attached when the desktop robot is connected to the PC. Then, the MCU will check all the motion controllers and configure each of them one by one. If any exception occurs during this procedure, system will stop and notice the MCU and thus report to PC. If the initialization process is normal, the system will run in a loop waiting for commands or requests from user. The user can monitor the robot and verify its state when it is running. This architecture ensures the desktop robot to be completely transparent to the end user, and it is important for achieving precise and smooth control while executing a complex task.

Generally, there are two types of data transmitted in this process: command and feedback. Command is sent to the

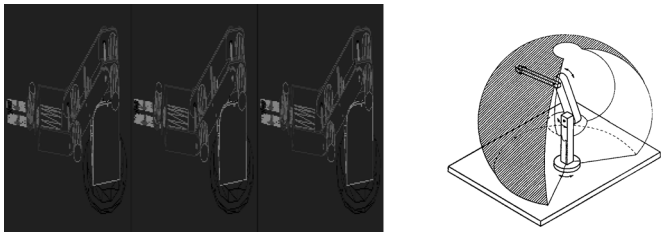


Fig. 3. Mechanism of the desktop robot.

https://doi.org/10.1017/S0263574708004815 Published online by Cambridge University Press

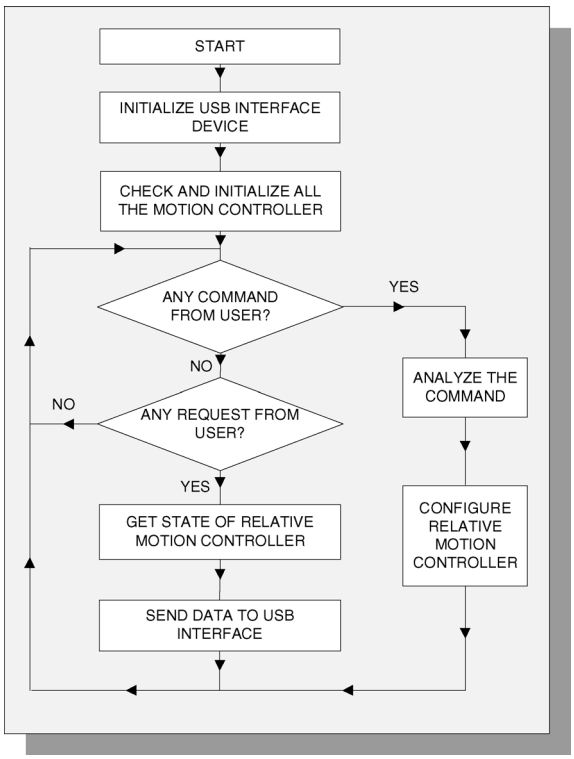


Fig. 4. Software structure of the host controller.

MCU by the PC or other main controllers, and feedback is a response from the MCU to the sender. Taking the feedback data as an example, these data are sent to the USB interface by the MCU. The USB interface converts the data according to the USB transmission protocol. Then the host side (PC or robot) can decode it, after receiving these data, to obtain the understandable feedback.

2.2.2. Control software at the PC side. As we mentioned before, the data would go through several layers before they are transmitted between the robot and the PC. Control software of the desktop robot on the PC side could be developed in any high-level programming language, such as C++ and JAVA, since the application programming interface (API) for the USB device control is definitive once it is installed. The multi-thread programming technology is applied to separate the reading control from the writing control on the USB interface, which can greatly improve the efficiency for data transmission. Here, threads can simply be viewed as independent tasks, and they run simultaneously in one program.

In the PC side, the desktop robot can be controlled as other common PC peripherals. The control panel communicates with the desktop robot through standard APIs of the operating system. So, the control software is independent of hardware. In order to make the control panel more friendly, the statuses of the robot's joints and video feedback from a digital camera (e.g., a webcam) are displayed in the control panel. At the same time, the robot parameters can be changed dynamically according to the environment and the task. The system is thus endowed with the adaptability to dynamical environment. Hence, it is much more flexible than other user interfaces which only have simple task definition.[17]

3. System Attributes

3.1. Plug-and-play feature

From the end user's point of view, PC peripherals with classical I/O interfaces, including serial/parallel ports, etc., do not have the feature of plug-and-play. Commonly, if those peripheral devices are connected to a running PC, the PC's operating system should reset or shutdown for safety. These devices lack flexibility in reconfiguring the system, which are inconvenient to use.[18] Plug-and-play function is more attractive for PC peripheral development.

The desktop robot is connected to a PC through the USB interface. This means it can be safely connected to or extracted from the PC dynamically, even if the PC is running. As it is required by a usual USB device, the driver for the desktop robot is installed on the host PC only at the first-time connection. It is not necessary to install any other additional component when the robot is used again since the robot would be recognized by the operating system automatically. The plug-and-play feature enables the end user to use the desktop robot on PC freely and does not need to care about tedious installation procedures, as required by the average industrial robot manipulators.

3.2. Application of distributed control structure

Distributed control systems have long been successfully applied in automation factories, warehouses, aviation, and other complex systems. Likewise, as the complexity of the robot system increases, the distributed control strategy should be an attractive solution for pursuing a satisfactory performance of it. Research in distributed control on robot so far has mainly been focusing on the multi-robot systems, instead of the internal components of a robot system.[19] In this paper, we explore an internally distributed motion control structure for the desktop robot.

As described above, the basic idea for our internally distributed motion control structure is to divide the system into several independent modules. Each module, i.e., an independent control subsystem, called a node, comprises a DC motor, a motor driver, and a motion controller. All nodes are connected to the MCU controller via an 8-bit data bus. Since the motion controller in each node is responsible for local control, workload on the MCU controller is very low. Hence, both the reliability and the capability of the overall system improve while its apparent complexity decreases. The MCU controller only takes charge of transmitting position commands to the motion controller and dealing with data transmission control in the USB side. This structure allows a tight integration of the high-level planning with the low-level control, which permits real-time easy modifications of the motion trajectory.[20]

4. Experiments

The desktop robot developed here, called Rinar, has four DOFs, which are capable of supporting most of the desktop applications, such as grasping a small object. The prototype of Rinar is shown in Fig. 5. It is 15 cm tall and 45 cm wide. It has four rotary joints, i.e., the gripper, the elbow, the shoulder, and the waist from the finger to the base.

https://doi.org/10.1017/S0263574708004815 Published online by Cambridge University Press

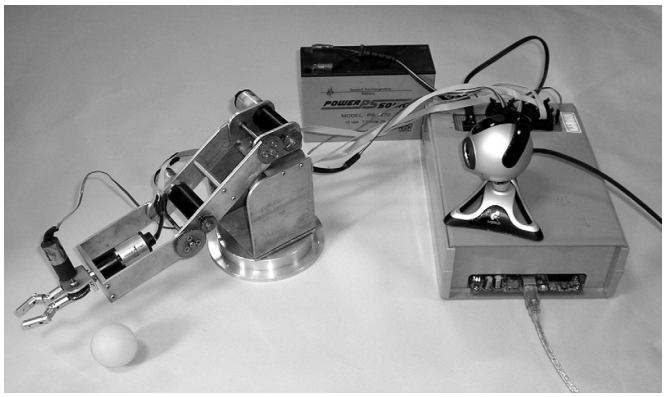


Fig. 5. Prototype of Rinar.

As for the Hi-Speed USB interface device, we choose ISP1581 from Philips Semiconductors (Philips). This chip supports both USB Specification 2.0 and USB Specification 1.1.

For the internally distributed control system, its host controller is an 8-bit micro-controller P89C61 from *Philips Semiconductors*, connecting the joint motion controller modules. By this structure, the system can fulfill speed control, position control, etc. The MCU controller receives control data from the host PC, and sends commands to the motion controllers. In this way, the motion controllers execute control computations, while the MCU controller focuses only on the USB interface control and data analysis.

For the control of the desktop robot, any sophisticated control methods could be employed for better performance, so long as sufficient control expenses could be paid. Here, we take the PID control scheme as an example to show the joint motion control of Rinar. The motion-control processor LM629, from National Semiconductor, is implemented in Rinar. It can perform the desired motion with velocity from 0 to 1,073,741,823/216 counts/sample; i.e., 0 to 16,383 counts/sample, with a resolution of 1/216 counts/sample. The motion controller will send corresponding pulse-width modulation (PWM) signals to the motor driver (L298 from *STMicroelectronics*) for the DC motor. For each motor, there is an encoder with 12 counts/resolution. Also the gearboxes are mounted on motors with the ratios of 231:1 (for the elbow motor) and 370:1 (for other three motors), respectively. Correspondingly, there are 2772 (=231×12) and 4440 (=370×12) counts per resolution, respectively. Whenever the desktop robot conducts a task, the PID controller would get all necessary information, including PID parameters, final position, velocity, and acceleration, for each motor from the MCU. After all the parameters are loaded to the system, the MCU would check the status of those PID controllers with a fixed sampling rate (100 Hz in this experiment). Eventually, all the data would be transferred to the PC through USB with the same rate.

Table II. Response time of Rinar.

Case	Configuration	Time (ms)
1	Elbow joint	21
2	Elbow and shoulder joints	36
3	Elbow, shoulder, and waist joints	52

The joints' response speed of Rinar is first tested. Figure 6. shows sample responses of elbow motor and shoulder motor during the task. Distributed control helps reducing complexity and improving the response speed of the control system. Moreover, by highly independent PID controllers, Rinar can run with a small joint angel error of 0.34°. This joint angle error is quite small in such type of robot arms and can meet accuracy requirement for a desktop robot manipulator.[21]

Table II summarizes the response time of the system in three cases: (1) control the elbow joint only; (2) control the elbow and the shoulder joints simultaneously; and (3) control the elbow, the shoulder, and the waist joints simultaneously.

A preliminary test of the whole robot arm has been carried out to evaluate its plug-and-play feature, the qualitative performance of distributed control system, the facility and flexibility of control panel, and the system response time.

We connect Rinar to the PC via the USB interface. The PC's operating system (e.g., Windows 2000) first finds it and asks for the driver for this device. After being installed, it is recognized by the operating system every time it is connected to the PC again. This is what we called plug-and-play function, as expected.

Then, Rinar is controlled as a general plug-and-play PC peripheral. A user control panel is developed for Rinar in

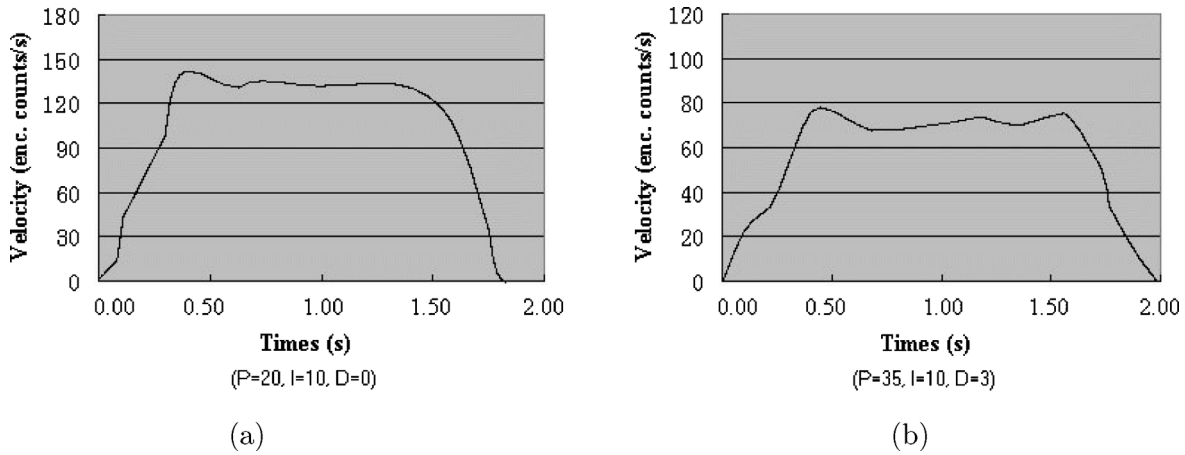


Fig. 6. (a) Step response of the elbow motor. (b) Step response of the shoulder motor.

https://doi.org/10.1017/S0263574708004815 Published online by Cambridge University Press

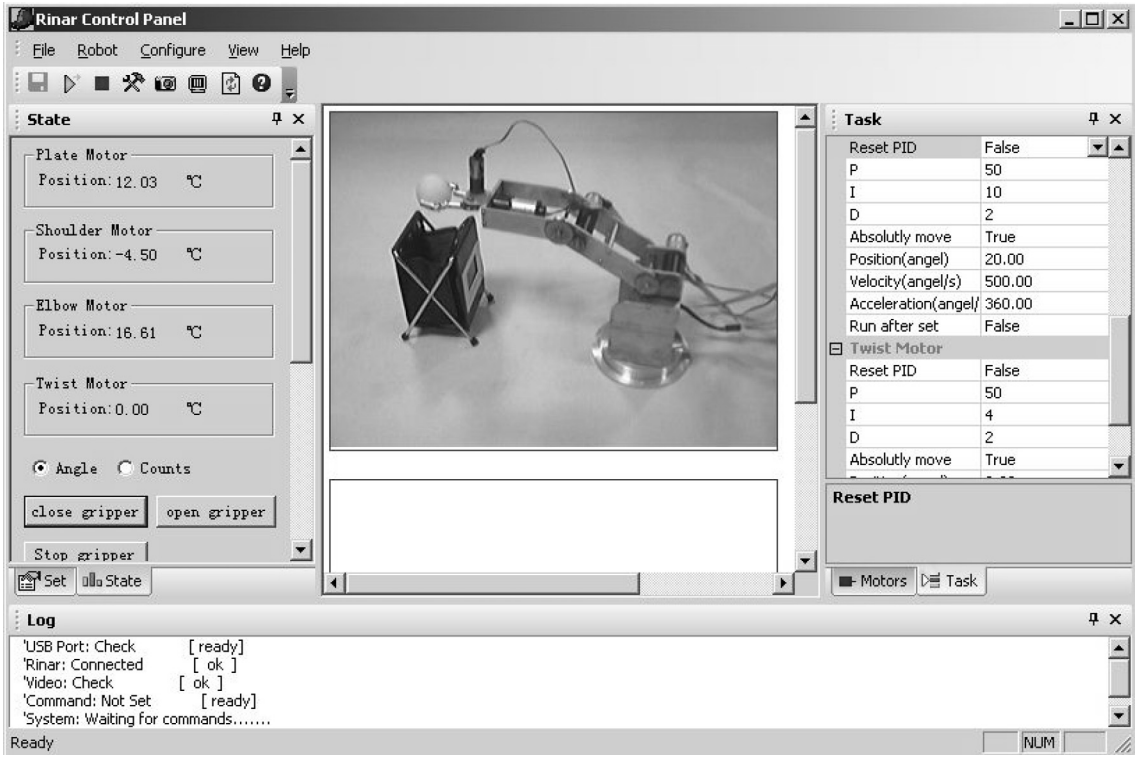


Fig. 7. User control panel of Rinar.

PC as a human–machine interface, as shown in Fig. 7. There are four display areas in the control panel. The left part displays the status of Rinar. Visual feedback is shown in the middle of the control panel. The system calibration model, control parameters, and the task are configured at the right part, while the task log information is shown at the bottom area. The user can monitor the running situation of Rinar and change the parameters very easily, if necessary, during the task execution.

The prescribed task here is to drive Rinar to pick up a ping-pong ball from the ground and put it into a basket. This task is a typical visual-servoing task, for which the

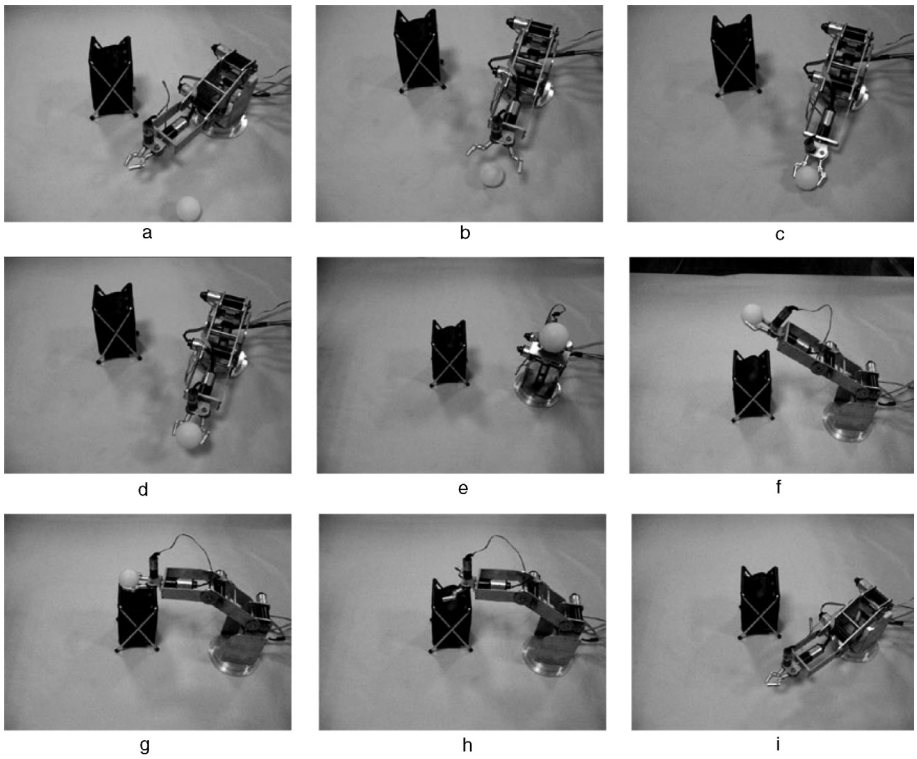


Fig. 8. A typical task executed by Rinar.

https://doi.org/10.1017/S0263574708004815 Published online by Cambridge University Press

visual-servoing strategy proposed in ref. [22] is adopted and the task is executed by Rinar automatically. The process is shown in Fig. 8. The ball is recognized and located by an external camera, and then by the host PC and Rinar [Fig. 8(a)]. Hence, Rinar can move its gripper to the position above the ball [Fig. 8(b)]. The gripper opens and moves down to catch the ball [Fig. 8(b) and (c)], and then moves to the position above the basket [Fig. 8(d)–(f)]. It drops the ball at a proper location above the basket [Fig. 8(g) and (h)]. After the task is finished, the gripper returns to the starting position [Fig. 8(i)]. We successfully conducted the same task eight times continuously. The motion precision and repeatability precision are high enough for the typical desktop task, which exhibits satisfactory performance of the proposed desktop robot system.

5. Conclusions

This paper presents a general approach to developing a desktop robot arm that has the plug-and-play property to the PC. The robot integrates an internally distributed control system that could be operated via the USB interface. A prototype system, Rinar, is developed as an example, which is shown to be able to responsd quickly and perform smooth motion control.

The essential property of the desktop robot reported in this paper is the plug-and-play feature to the PC via the USB manner. Some other plug-and-play manners, or even the wireless manners, such as the infrared, bluetooth, and 802.11 b, are also under consideration. Moreover, an internal digital camera is being added to Rinar that can take advantage of the capabilities of the USB interface. With improved performance and sensibility, this plug-and-play robot arm has potentials for widespread use in home and office environment.

Acknowledgment

The work was supported in part by the National Natural Science Foundation of China under grant 60675041.

References

1. International Federation of Robotics, World Robotics 2003, http://www.ifr.org.
2. G. F. Wyeth, J. Kennedy and J. Lillywhite, "Distributed Digital Control of a Robot Arm," *Proceedings of the Australaian Conference on Robotics and Automation (ACRA 2000)*, Melbourne (2000) pp. 217–222.
3. J. Mikkelsen, "A Machine Vision System Controlling a Lynx Arm Robot Along a Path," Technical Report, University of Cape Town, South Africa (Oct. 28, 1998).
4. Intelitek, SCORBOT-ER 5plus Robotic Arm (Jan. 2004), http://www. intelitek.com/.
5. Universal Serial Bus Specification Revision 2.0 (Sept. 2003), http://www. usb.org/.
6. T. Kouyama, H. Nano, C. Kon and N. Shimizu, "The Design of a USB Device Controller IYOYOYO," *Proceedings of the Asia and South Pacific Design Automation Conference*, Katakyashu, Japan (Jan. 2003) pp. 573–574.
7. Lego Company, "Mindstorms?NXT?Kit", http://mindstorms.lego.com/, (Sept. 2003).
8. Sensable Technologies, "Sensable FreeForm Modeling and Modeling Plus Systems Introduction", http://www.sensable.com/products-freeform-systems.htm, (Feb. 2004).
9. Cho Kyoung-Youn and Choi Kwang-ho, "A Novel Architecture of Home Gateway for Efficient Packet Process," *Proceedings of IEEE Workshop on Knowledge Media Networking*, Kyoto, Japan (Jul. 2002) pp. 63–67.
10. J. Axelson, "USB Complete-Everything You Need To Develop Custom USB Peripherals," Lakeview Research, Madison, (1999) pp. 14–16.
11. M. Morita and S. Uchikado, "New Design of Visual Servoing Using an Uncalibrated Camera and a Calibrated Robot," *Proceedings of the IEEE 28th Annual Conference of the Industrial Electronics Society*, Sevilla, Spain, Vol. 3 (Nov. 2002) pp. 1954– 1959.
12. Z. Zheng, "Application of Distributed Control Unit (DCU) in Machine Control Area," *Proceedings of IEEE International Conference on Electronics, Circuits and Systems*, Lisboa, Portugal, Vol. 1 (Sep. 1998) pp. 249–252.
13. P. Purkayashta, S. Sood and K. Padmanabhan, "Computer Aided Engineering of Distributed Control Systems," *Proceedings of IEEE/IAS International Conference on Industrial Automation and Control*, Taipei, Taiwan (1995) pp. 441–444.
14. T. J. Tarn, "Effect of motor dynamics on nonlinear feedback robot arm control," *IEEE Trans. Rob. Automat.* **7**(1) (Feb. 1991) pp. 114–122.
15. R. Parkin, *The Basis for New Industrial Development* (Comput. Mech. Publications, Southampton, UK, 1994) pp. 521–526.
16. L. Sciavicco and B. Siciliano, *Modelling and Control of Robot Manipulators* (Springer Verlag, New York, 2000) pp. 9–10.
17. D. Kushida, M. Nakamura, S. Goto and N. Kyura, "Force-Free Control of Articulated Robot Arm Considering Velocity Along Assigned Locus," *Proceedings of the 41st SICE Annual Conference*, Osaka, Japan, Vol. 4 (2002) pp. 2167–2171.
18. C. J. N. Coelho Jr., A. O. Fernandes, M. P. de Oliveira and A. I. Tavares, "An Embedded Converter From RS232 to Universal Serial Bus," *Proceedings of 14th Symposium on Integrated Circuits and Systems Design*, Pirenopolis, Brazil (Sep. 2001) pp. 91–96.
19. Y. Wei, T. R. Vaughan, G. S. Sukhatme, J. Heidemann, D. Estrin and M. J. Mataric, "Evaluating Control Strategies for Wireless Networked Robots Using an Integrated Robot and Network Simulation", *Proceedings of the IEEE International Conference on Robotics and Automation*, Seoul, South Korea, Vol. 3 (2001) pp. 2941–2947.
20. A. Chella, V. Di Gesu, G. Gerardi, I. Infantino, D. Intravaia, B. Lenzitti, G. Lo Bosco, A. Messina, R. Pirrone and P. Storniolo, "DAISY: A Distributed Architecture for Intelligent System", *Proceedings of the 4th IEEE International Workshop on Computer Architecture for Machine Perception*, Boston, USA (Oct. 1997) pp. 42–50.
21. T. J. Tarn, "Performance comparison of four manipulator servo schemes," *IEEE Control Syst. Mag.* **13**(1), 22–29 (Feb. 1993).
22. J. Su, W. Qiu, H. Ma and P. Woo, "Calibration-free robotic eye–hand coordination based on an auto disturbance-rejection controller," *IEEE Trans. Rob.* **20**(5), 899–907 (Oct. 2004).

https://doi.org/10.1017/S0263574708004815 Published online by Cambridge University Press