Design Features and Control Strategies of a Generic Internet Based Telerobotic System

Sudth R. Munasinghe*, Ju-Jang Lee*

Yuji Ishida**, Naruto Egashira***, and Masatoshi Namakura**

* Department of Electrical Engineering and Computer Science, KAIST, Korea

(Tel : +82-42-869-8032; E-mail: rohan@odyssey.kaist.ac.kr, jjlee@ee.kaist.ac.kr)

** Department of Advanced Systems Control Engineering, Saga University, Japan

(Tel : +81-952-288-643; E-mail: {ishida,nakamura}@cntl.ee.saga-u.ac.jp)

*** Department of Control and Information Systems Engineering, Kurume National College of Technology, 830-8555

1-1-1 Kumorino Kurume-City, Fukuoka, Japan (Tel: 81-942-359323; Email: naruto@kurume-nct.ac,jp)

Abstract: This paper describes the design and implementation techniques of a generic Internet based telerobotic system. The purpose is to make precious technical information and hands-on experience of the authors widely available for the research community. The paper is based on the telerobotics system that the authors recently implemented between KAIST and Saga University, Japan. In its current functionality, two control modes can be exercised; high-level supervisory control, and low-level supervisory control.

Keywords: Telerobotics, supervisory control, real-time process, non-real time process

1. INTRODUCTION

Telerobotics is becoming an increasingly important research and application area. Telerboics and teleoperations have been introduced in a wide range of applications recently. Ranging from space and on-orbit [1], teleoperations have found its potential in undersea [2], medical [3], welfare [4], rescue [5], and entertainment [6] applications. Yet, telerobotics research, however, is limited to a very few institutions, and telerobotics is not in the main stream robotics research. One of the most significant reasons behind this apparent scenario is that it needs a very wide range of multi-disciplinary expertise to construct and maintain a telerobotics system. It also seems that the technical details of telerobotics system designs have not been properly documented and widely available for the research community. Therefore, it is truly an exhaustive task to design and implement a telerobotic system.

In this view, the authors are motivated to present a detailed account on design and implementation of a generic Internet-based telerobotic system. The design details may help novice researchers to quickly grasp the key ideas required to implement their own telerobotic systems. The generic telerobotic system described in this paper is based on client-server concept; client is the remote operator, and server is the local controller of the telerobot. Remote operator sends motion commands to the local controller, and the local controller receives those commands, carries out trajectory planning, and finally controls the telerobot.

Two principle control strategies are described in the generic design; high-level supervisory control and low-level supervisory control. In high-level supervisory control, remote operator only sends motion and control parameters to the local controller, and trajectory panning and robot control take place autonomously. On the other hand, in low-level supervisory control, remote operator keeps sending incremental position update commands, while the local controller plans the incremental motion trajectories, and controls the telerobot in a move and wait pattern. High-level supervisory control needs an advanced trajectory planner [7-8] within the local controller, whereas low-level supervisory control can be implemented with a very simple trajectory planner.

Based on the above two control strategies, the authors have designed and implemented an Internet based telerobotic system between Korea Advanced Institute of Science and Technology (KAIST) and Saga University, Japan. Almost all design features and details of this telerobotic system are presented in this paper. The two principle control strategies have been experimented on this telerobotic system. The experimental results are also presented highlighting the inherent features of them. Experimental results verified the successful design and functionality of the constructed telerobotic system. The results also illustrate the merits and demerits of the two control strategies. Both these control strategies are essential in general telerobotic applications, and the results confirm when and where they should be distinctively employed. Directions for further developments and interesting research that could be carried out on the generic telerobotic system are also discussed.

2. SUPERVISORY CONTROL AND SYSTEM DESIGN

The most fundamental and crucial issue in telerobotics is the delay in signal transmission between the remote operator and telerobot controller [9]. In direct telemanipulation, where remote operator is a part of the control loop, the feedback control loop spans both sides of the telerobotic system. At the remote operator side, the feedback is received from the telerobot controller, determines the control action, and sends it across the transmission network. However, when the telerobotic system spans over a long geographical distance, such as in inter-continental, or ground based space telerobotics, the transmission delay can be significantly long, thus, the delayed feedback in direct teleoperation can drive the system unstable. Therefore, very slow speeds are used to maintain the stability.

Supervisory control [9] has overshadowed most direct teleoperations on the fact that most robotic manipulations are relatively simple, and that most general tasks can be planned and executed autonomously by the telerobotic controller alone. For such teleoperations, it is not necessary that the remote operator remain within the control loop. Rather, the control loop can be implemented locally, together with a trajectory planner that plans the motion trajectory autonomously. Then, the remote operator only has to oversee the performance of the telerobot while sending necessary instructions to the trajectory planner.

2.1 High-Level Supervisory Control Strategy

For simple motions and manipulations in static environments, a high-level supervision is appropriate. In high-level supervisory control, remote operator has to send motion and control parameters to the local controller. The rest of the teleoperation is autonomously implemented by the local controller alone. Figure 1 illustrates the telerobotic system implemented between KAIST and Saga University, Japan based on the high-level supervisory control strategy.

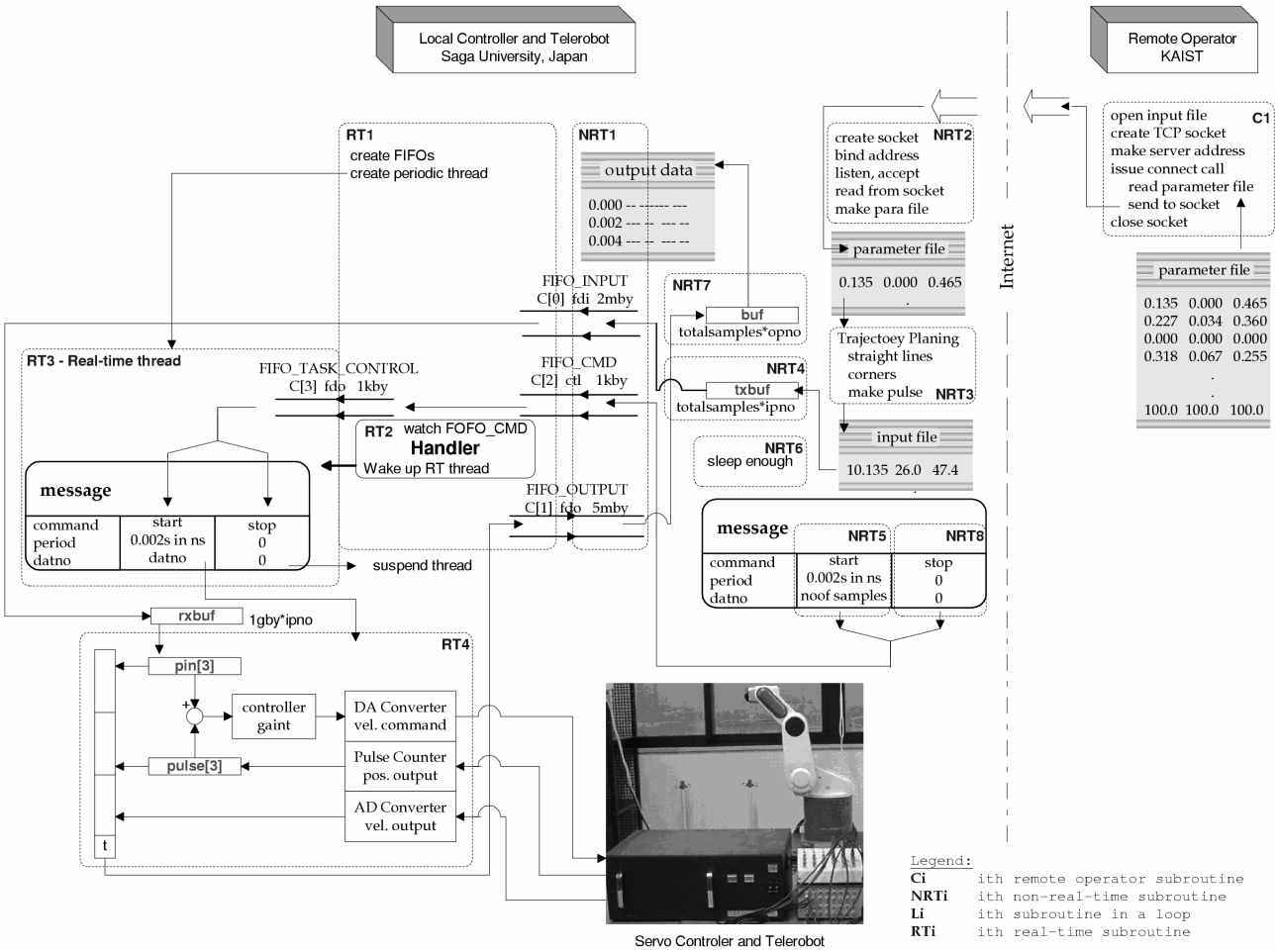

Fig 1. Telerobotic systems implemented between KAIST and Saga University based on high-level supervisory control strategy

Remote Operator:

As illustrated in Fig.1 above, remote operator resides in KAIST, and the local controller and telerobot reside in Saga University, Japan. The two sides communicate through TCP sockets [10] over the Internet. Remote operator reads motion and control parameters from a resident file, and sends those parameters to the local controller in Saga University (**C1**).

Local Controller:

Local controller operates a non real-time process (**NRT1**~**NRT8**). It reads motion and control parameters through a TCP socket, and writes them onto a file (**NRT2**). The entire motion trajectory is planned by an advanced trajectory planner (**NRT3**), which eventually creates the input data file. This pulse file contains the input data sequences of the planned motion of the telerobot. The pulse data is then transferred to a serial buffer whose size is determined by the number of controlled joint of the telerobot, and number of input data samples. The input data is then written to first-in-first-out buffer, FIFO-INPUT (**NRT4**). The message structure is then written with "start" command, together with sampling interval used by the trajectory planner, and number of input samples in the planned trajectory. This message structure is placed in FIFO-CMD (**NRT5**). Starting from this point, the non real-time (**NRT**s) process waits, giving time for the real-time (**RT**) process to carry telerobot control (**NRT6**).

Local controller also operates a real-time process (**RT1**~**RT4**), which is programmed using Pthreads [11]. A handler is created for the FIFO_CMD so that any command written to FIFO_CMD is immediately taken into attention by the handler (**RT2**). When the handler notices a new command in FIFO_CMD, it copies it onto FIFO_TASK_CONTROL, and immediately sends wake-up signal to the real-time periodic thread that runs the control loop of the telerobot (**RT3**). Being woken up by the handler, the real-time periodic thread reads the command from FIFO_TASK_CONTROL to its message structure. If it finds "start" command, then it reads the message structure member variables to know the sampling interval and number of input data samples for the planned motion. The input data is then read from FIFO_INPUT to a serial buffer named rxbuf, which is big enough to hold all input data of the planned trajectory. By this time, real-time periodic thread is ready to carry out the

planed motion of the telerobot. The control loop (**RT4**) is inside the real-time periodic thread, and it operated by executing the feedback control loop with the input data in the serial buffer, rxbuf. The number of control loop operations is already specified in the message structure. In every control loop operation, joint positions of the telerobot are read from the servo controller. A pulse counter provides position feedback whereas a digital-to-analog (DA) converter provides velocity feedback. The outpud data is immediately sent to FIFO_OUTPUT.

The non real-time process then sends "stop" command to FIFO_CMD (**NRT8**). The handler reads it and immediately places it onto FIFO_TASK_CONTROL, and wakes up (or interrupt) the running real-time thread (**RT3**). Once real-time thread sees the "stop" command it suspends the control loop, and the real-time process terminates. The non real-time process continues for a while, and reads the output data from the FIFO_OUTPUT to a serial buffer (**NRT7**), which is long enough to hold the total amount of output data. Output is then written to a data file, following correct row-column configuration.

2.2 Low-Level Supervisory Control Strategy

The low-level supervisory control strategy is illustrated in Fig.2. This design looks similar to the previous high-level supervisory control design, with few major changes. On the remote operator side, reference inputs are read repetitively from a file. For each reference position, it operates a loop (**C2**) within which it segments the individual motions into a set of equi-distance positions, and sends them to the local controller through a TCP socket. The local controller also runs a non real-time loop (**L1~L4**) within which it reads the incremental positions, distance between them, duration for the incremental motion, and number of input samples in each incremental motion (**L1** and **L2**). The input data for the incremental motion of the telerobot is sent to a serial buffer and then placed onto FIFO_INPUT (**L3**). This incremental motion is planned assuming uniform end-effector speed. Then, the "start" command is written onto the message structure, together with the number of input data samples for the incremental motion. The message structure is placed onto the FIFO_CMD, and the non-real time process waits giving time for the real-time process to control the telerobot along the incremental motion.

The real-time process has the handler (**RT2**) and the periodic real-time thread (**RT3**) operating in the same way as described in the high-level supervisory control design. One exception, however, is that the control loop has one additional mode (**RT5)**. This control mode operates during the idling time between two consecutive incremental motions. It uses the last input data as the input for the robot joint controllers, and stabilizes the robot at that point until next incremental position input data appears. However, it does not write output to the FIFO_OUTPUT. Both real-time and non real-time processes terminate essentially the same way as in the previous design.

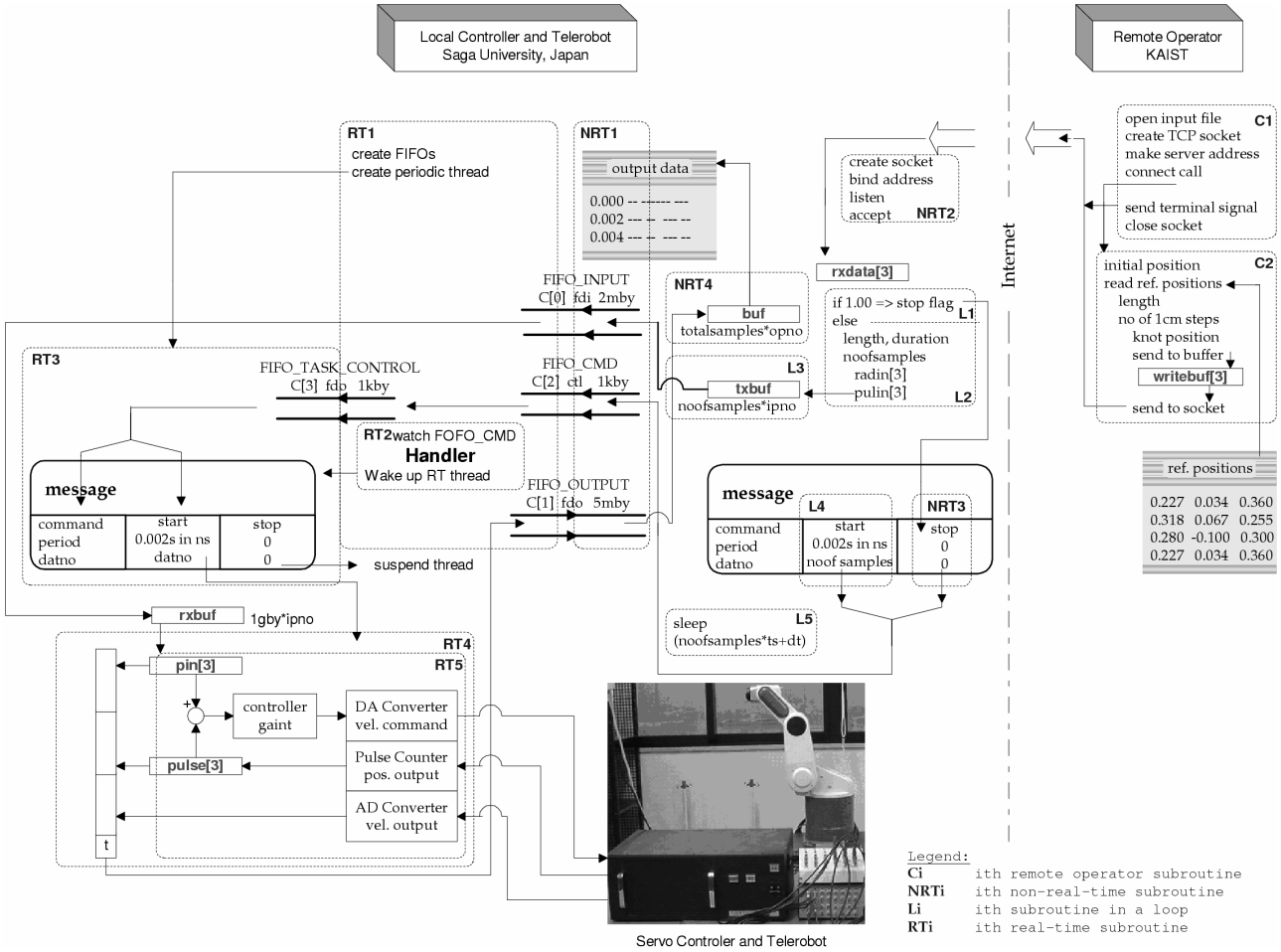

Fig.2 Telerobotic systems implemented between KAIST and Saga University based on low-level supervisory control strategy

2.3 Implementation

The real-time processes were programmed using real-time Linux. Remote operator was implemented on RedHat Linux? 7.3 with kernel version 2.4.20-13.8, whereas local telerobotic controller was implemented on Debian GNU Linux 2.2 (potato) with real-time kernel version 2.2.19-rtl. For Internet security reasons, local controller was not assigned a global IP address. Instead, virtual private network (VPN) server was used in the local controller and enabled access for the remote operator. A simple visual feedback was also implemented with MSN messenger. The visual feedback is not part of the control loop. It only allows remote operator to oversee the motion of the telerobot on his commands.

3. EXPERIMENTS AND RESULTS

3.1 High-level supervisory control

The reference end-effector positions P1(0.227, 0.034, 0.360), P2(0.318, 0.067, 0.255), P3(0.280, -0.100, 0.300), P4=P1, assigned end-effector velocity, v_a=0.3[m/s] were used as motion parameters from the remote operator. The results of the experiment are shown in Fig. 3, where the reference end-effector positions P1~P4 are labeled at he top.

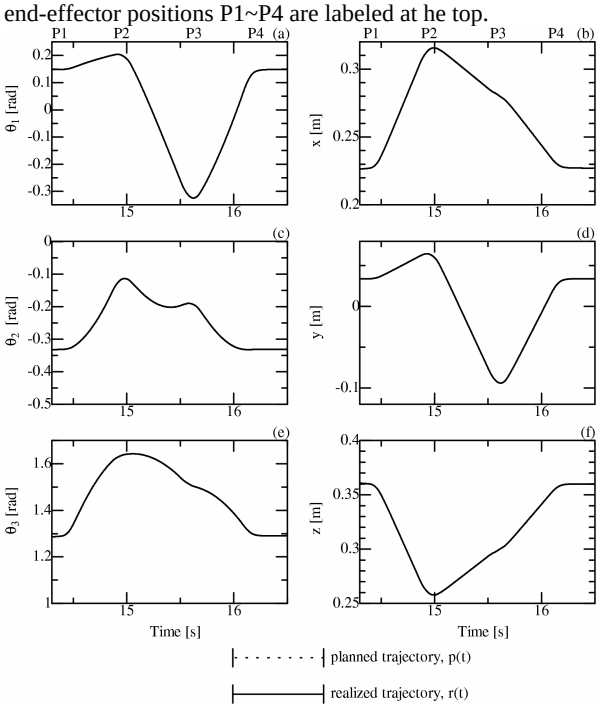

Fig 3. Joint and Cartesian position results of the high-level supervisory control.

End-effector velocity is shown in Fig. 4

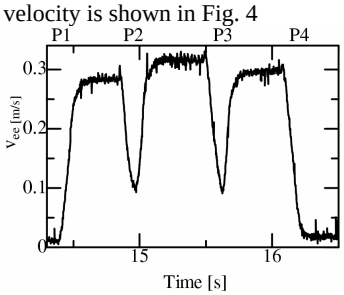

Fig.4 End-effector velocity with high-level supervisory control

The most valued result is the end-effector trajectory in three dimensional Cartesian spaces, as shown in Fig.5

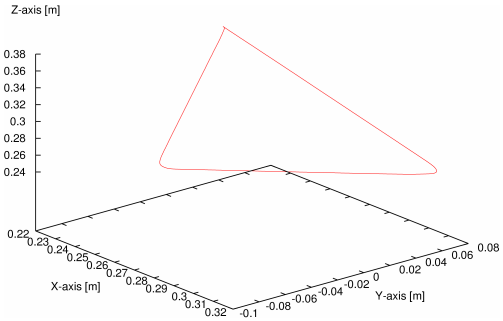

Fig.5 End-effector trajectory performed under high-level supervisory control

3.2 Low-level supervisory control

Under low level supervisory control experiment, following reference trajectory was used: Q1(0.227, 0.034,0.360), Q2(0.318, 0.067, 0.255), Q3(0.280, 0.034, 0.360), and Q4=Q1. Distance of the incremental end-effector motion was set to 1[cm]. The incremental position generator (**C2** in Fig.2) generates 70 increments between Q1~Q2, 69 increments between Q2~Q3, and 72 increments between Q3~Q4. Sending these incremental positions (remote operator side), and control of the telerobot (local controller side) take pace simultaneously. Results of the joint and Cartesian position are shown in Fig.6

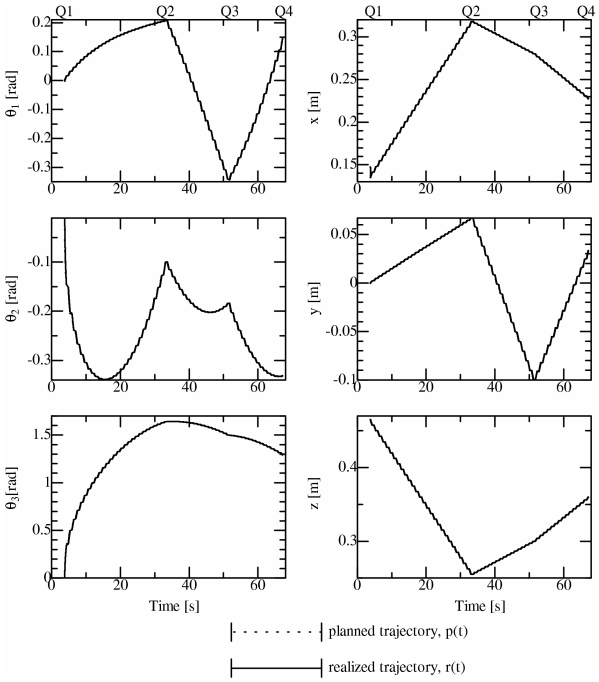

Fig. 6 Joint and Cartesian position results of the low-level supervisory control.

The incremental position movements are elaborated in Fig. 7 by enlarging a small time slice of Fig.7

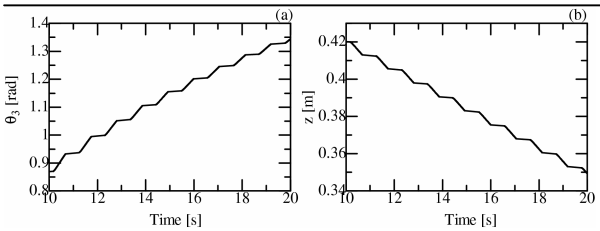

Fig.7 incremental motion in joints and Cartesian space

End-effector velocity is shown in Fig.8

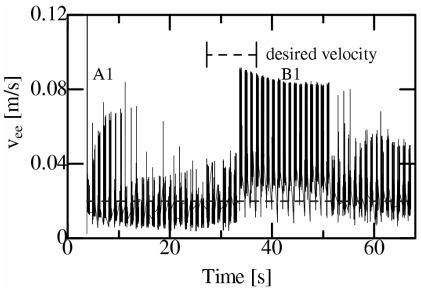

Fig.7 End-effector velocity under low-level supervisory control

The end-effector motion in three dimensional space is shown in Fig.8

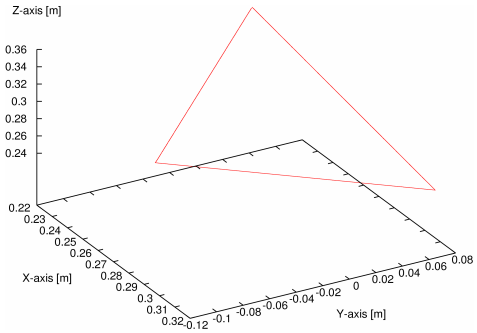

Fig.8 End-effector motion under low-level supervisory control.

3.2 Results Evaluation

The end-effector motion performance in both high-level and low level supervisory control are accurate, and comparable (Fig.5 and Fig.8). However, high-level control shows much better velocity profile (Fig. 4) compared to low-level control (Fig.7). In high-level control, an advanced trajectory [7-8] planner constructs the entire end-effector trajectory so that the resulting motions of joint and the end-effector have become much smoother (Fig. 3) compared to that of low-level control (Fig. 6), where there is no advanced trajectory planner. The move-and-wait nature of the incremental motions under low-level control is visible in Fig. 6, and more elaborated in Fig 7. The waiting time, after every incremental move, is one of the reasons for low-level control operation to take excessively long time, about 65s to complete a comparable physical motion that the high-level control performs in about 16s.

High-level supervision always brings better performance, yet it can only be applied for relatively simple tasks, in static environments. On the contrary, low-level supervisory control can be applied for complex tasks in highly dynamic environments, where other control strategies would simply fail.

4. CONCLUSION AND DISCUSSION

Two basic control strategies; high-level supervisory control and low-level supervisory control, and their corresponding telerobotic system designs have been presented in a significant depth. Amateur telerobotic researchers and engineers would find these designs, details, and results as important guidelines for them to design their own telerobotic systems. The merits and demerits of high-level and low-level control strategies in telerobotics have also been demonstrated.

5. FUTURE DEVELOPMENTS AND RESEARCH

A hand device could be attached at the remote operator side, to generate remote operator commands more human interactive manner. The waiting time in low-level supervisory control has to me minimized so that to make it more continuous in motion. However, it is almost impossible to model this waiting time using standard mathematical procedure. Yet intelligent modeling and prediction techniques [12] would probably bring about a better solution.

REFERENCES

[1] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, “Sensor-based space robotics—ROTEX and its telerobotic features,” *IEEE Tr. Robot. and Automat.*, Vol. 9, No.5, pp. 649-663, 1993.

[2] P. G. Neckes, M. K. Long, “Local-remote telerobotics for underwater vehicles,” *Proc. Symp. on Autonomous Underwater Vehicle Technology*, pp. 11-15, 1992.

[3] A. Rovetta. R. Sala, W. Xia, A. Togno, “Remote control in telerobotic surgery,” *IEEE Tr. Syst., Man and Cyber. Part A*, Vol. 26, No.4, pp. 438-444, 1996.

[4] S. R. Munasinghe and M. Nakamura, “Teleoperation of welfare robotic systems by motion planning considering assigned velocity and acceleration limit,” *Intl. J. of human-friendly welfare robotic systems*, Vol. 3, No.2, pp. 23-31, 2002.

[5] A. Rovetta, “FRIEND robot, space telerobot for rescue and recovery of astronauts,” *IEE/RSJ International Workshop on Intelligent Robots and Systems*, Vol.3, pp. 1663-1668, 1991.

[6] K. Goldberg, “Mercury Project: A feasibility study for internet robots,” *IEEE Robotics and Automation Magazine*, Vol.7, No. 1, pp 35-40, 2000.

[7] S. R. Munasinghe, M. Nakamura, S. Goto, and N. Kyura, “Optimum contouring of industrial robot arms under assigned velocity and torque constaints,” *IEEE Trans. on Syst., Man, and Cyber.*, Vol. 31, No. 2, pp. 159-167, 2001.

[8] S. R. Munasinghe, M. Nakamura, S. Goto, and N. Kyura, “Trajectory planning for industrial robot manipulators considering assigned velocity and allowance under joint acceleration limit,” *Intl. J. of Control, Automation and Systems*, Vol. 1, No.1, pp. 68-75, 2003.

[9] T. B. Sheridan, *"Telerobotics, Automation and Human Supervisory Control*," MIT Press, Cam., MA, 1992.

[10] W. A. Gay, *Linux Socket Programming*, Que Inc, Indianapolis, 2000.

[11] B. Nichols, D. Buttler, and J. P. Farrel, *Pthreads Programming*, O’Reilly & Associates, Inc., Sebastipol, CA, 1996.

[12] B. Kosko, *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence*, Prentice Hall Int., New Delhi, 1996.