AN APPROACH TO TELEROBOT COMPUTING ARCHITECTURE

John Fiala
Ronald Lumia

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Robot Systems Division
Intelligent Controls Group
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

U.S. DEPARTMENT OF COMMERCE
Robert A. Mosbacher, Secretary
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Dr. John W. Lyons, Director
<table>
<thead>
<tr>
<th>DATE DUE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Demco, Inc. 38-290
AN APPROACH TO TELEROBOT COMPUTING ARCHITECTURE

John Fiala
Ronald Lumia

U.S. DEPARTMENT OF COMMERCE
National Institute of Standards and Technology
Robot Systems Division
Intelligent Controls Group
Bldg. 220 Rm. B124
Gaithersburg, MD 20899

June 1990
An Approach to Telerobot Computing Architecture

Intelligent Controls Group
Robot Systems Division
National Institute of Standards and Technology

Scope of the Document

In response to a Goddard Space Flight Center effort to look at the requirements for a computing architecture design appropriate for a telerobot, this document describes the Intelligent Controls Group approach to telerobot computing architectures. It is shown how a seven microprocessor control system can achieve teleoperation with an around-the-loop time of 10 ms. The document focuses on low-level, real-time robotics and does not discuss some issues relevant to space systems such as space-qualification and thermal design.

Keywords: telerobotics, teleoperation, control system architecture, robot control, microprocessor
1. Introduction

This document presents a basic design for a telerobot. The complete robot design is not presented here, however, detailed designs of principal subsystems are presented. The emphasis is on the approaches taken toward software and hardware design of the computing architecture. Example designs are taken from the Servo Level. The motivation for the control system design is completely documented in the references at the end. The bibliography is a complete list of publications from the Robot Systems Division, National Institute of Standards and Technology (NIST), related to the design of telerobot control systems.

2. Software

To facilitate the discussion and simplify the figures, the following discussion on the design of software will concentrate on the Servo Levels of two main subsystems of the telerobot, the hand-controller subsystem of the operator interface and the manipulator subsystem of the robot side. These two subsystems are the main emphasis for projects such as Martin Marietta's FTS and Goddard's Multiple Algorithm Robot Control System (MARCS). Other levels and subsystems will have similar structure, so the example can be applied to other parts of the system as well.

The control system forms two main hierarchies, one for the operator interface and one for the robot. In the operator interface hierarchy, a hand-controller subsystem consists of a Primitive and Servo Level under the E-move Level. Likewise, as shown in Figure 1, a manipulator controller

```
Figure 1. Subsystem Design for Arm and Hand Controller of Telerobot.
```
consists of a Primitive and Servo Level under the Manipulation E-move Level. On the robot side there are also E-move Levels for the Perception and Positioning subsystems. There is a second manipulator subsystem under the Manipulation E-move Level. This subsystem controls the second manipulator arm in a two arm telerobot. Similarly, there is a second hand-controller subsystem in the operator interface. In the discussion, the control subsystem for only one manipulator/hand-controller pair will be presented, the assumption being that a second manipulator/hand-controller pair would have a duplicate control structure.

The Servo Levels depicted in Figure 1 are directed by the corresponding Primitive Levels. However, in teleoperation mode, the two Servo Levels are linked through the global data system. The manipulator is moved based on sensory data generated in the hand-controller subsystem and the hand-controller is activated by sensory data generated in the manipulator subsystem. This teleoperation mode will be the focus of the discussion.

Figure 2 shows a detailed design for the Servo Levels of Figure 1. Figure 2.a represents the manipulator subsystem Servo Level and Figure 2.b represents the hand-controller Servo Level. Each box in Figure 2 is a cyclicly-executing, concurrent software process of the control system. The ovals are global data buffers. System processes communicate only through these global buffers. The particular configuration of boxes shown in Figure 2 is primarily based on the control architecture of the NIST lab system, but includes elements for the implementation of Martin Marietta's control algorithm. This design is also similar to the MARCS system, which is specifically designed to support algorithms like Martin's. (See Appendix 1 for the detailed MARCS design.) Additional boxes might be added for other algorithms, such as the Goddard teleoperation algorithm with adaptive gains, however the basic arrangement and characteristic of software processes depicted in Figure 2 is designed to support a wide range of algorithms without major modification.

Note that not all possible inputs and outputs of processes are shown in the figure. The figure shows all inputs and outputs necessary for the basic teleoperation mode under consideration. Detailed descriptions of the inputs and outputs to the processes in Figure 2 can be found in the references. Interfaces shown are assumed to be complete interfaces as described in the documents even though only a subset of a given interface might be used for the algorithm being discussed. Also, the output from the Servo Level is the desired torque for the joints. It is assumed that either this value is directly proportional to the required motor current, as in the direct-drive case, or that torque loops are provided subsequent to the output.

The data buffers used for communication between subsystems are shown shaded in Figure 2. The two buffers are repeated in both figures for clarity. The buffer $f_m$ contains the force feedback from the robot force sensor related to the Cartesian coordinates of the hand-controller. The buffer $z_o$ contains the Cartesian hand-controller command related to the Cartesian coordinates used for the robot control. The $z_o$ data is the final hand-controller command which includes indexing and scaling.

Each process at the Servo Level in Figure 2 has associated with it a number appearing in the upper right-hand corner of the box. Table 1 gives the execution times for each process executing the appropriate component of the Martin Marietta algorithm. Note that the Martin algorithm is completely implemented by the numbered processes. In fact, some additional functionality (such as gravity compensation) is provided which is not available in Martin's implementation. The execution times are based on Martin Marietta published data. Martin Marietta obtained these times by
Figure 2. a. Manipulator Subsystem Detailed Design.
Figure 2. b. Hand Controller Subsystem Detailed Design.
Table 1. Execution/Communication Times for Figure 2 Processes in Martin Algorithm.

<table>
<thead>
<tr>
<th>Process</th>
<th>Process Function in Martin Algorithm</th>
<th>Execution Time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Acquire joint feedback</td>
<td>0.34</td>
</tr>
<tr>
<td>2</td>
<td>Acquire force/torque sensor data</td>
<td>0.17</td>
</tr>
<tr>
<td>3</td>
<td>Calc. position mod. based on impedance</td>
<td>1.50</td>
</tr>
<tr>
<td>4</td>
<td>N/A</td>
<td>2.10</td>
</tr>
<tr>
<td>5</td>
<td>Calc. inertia decoupling matrix</td>
<td>17.00</td>
</tr>
<tr>
<td>6</td>
<td>N/A</td>
<td>4.70</td>
</tr>
<tr>
<td>7</td>
<td>Forward kinematics on joint position</td>
<td>1.43</td>
</tr>
<tr>
<td>8</td>
<td>Trans. force sensor data to control coord.</td>
<td>0.38</td>
</tr>
<tr>
<td>9</td>
<td>Translate manip. to hand cntlr coord.</td>
<td>0.39</td>
</tr>
<tr>
<td>10</td>
<td>Combine imp. and cmd/ inverse kinematics</td>
<td>1.47</td>
</tr>
<tr>
<td>11</td>
<td>Joint rate limiting</td>
<td>0.27</td>
</tr>
<tr>
<td>12</td>
<td>Joint PD control with inertia decoupling</td>
<td>0.90</td>
</tr>
<tr>
<td>13</td>
<td>N/A</td>
<td>0.25</td>
</tr>
<tr>
<td>14</td>
<td>Force feedback limiting</td>
<td>0.27</td>
</tr>
<tr>
<td>15</td>
<td>Jacobian transpose multiplication</td>
<td>0.57</td>
</tr>
<tr>
<td>16</td>
<td>Translate hand cntlr to manip. coord.</td>
<td>0.78</td>
</tr>
<tr>
<td>17</td>
<td>N/A</td>
<td>1.80</td>
</tr>
<tr>
<td>18</td>
<td>Calc. hand-controller Jacobian</td>
<td>1.32</td>
</tr>
<tr>
<td>19</td>
<td>Forward kinematics on joint position</td>
<td>1.43</td>
</tr>
<tr>
<td>20</td>
<td>Acquire joint feedback</td>
<td>0.34</td>
</tr>
</tbody>
</table>

implementing and timing actual Ada code on the 80386 microprocessor. In calculating the times for Table 1 for each process, Martin Marietta execution times for functions performed in a process were summed and the resulting value was multiplied by 1.2 to allow for a communication overhead of 20%. The result is rounded up to the nearest hundredth of a millisecond. For processes for which execution times are not available, times obtained on Ada code implemented on the 68020 for a seven-degree-of-freedom manipulator control system were used as comparable values. The execution times of Table 1 are used in the next section to show how processes can be distributed to processors for real-time performance when the correct hardware architecture is chosen.

3. Hardware

The problem of telerobot control requires a powerful multiprocessing architecture. This multiprocessing architecture must involve a large number of tightly-coupled processors to perform the
highly centralized aspects of coordinated control. This fact will be demonstrated for the example subsystems of Section 2. First, consider the following desirable features of the centralized part of the computer architecture design for a telerobot.

3.1. Processor coupling via a 32-bit data bus.

The rate that data is transferred between processors in the control system requires that the communication bus support a high communication bandwidth. Since many processors will share the same data bus, megaword-per-second data rates are necessary. Also, the word size for values used in robot applications is mostly 32 bits, since robot control computations are done using 32-bit floating point representations. Current floating point hardware technology is capable of computations with no less than 32 bits. Any smaller representation requires expensive conversions to a larger format. In addition, 32-bit data is the state-of-the-art in microprocessors and multi-microprocessor backplanes.

3.2. Multislot (>20) backplane.

To accommodate the large number of processors and allow for easy system expansion, a multislot backplane is required. Processor and other hardware elements connected by an address and data bus configured as a multislot backplane provides for tight coupling with the added benefit that elements can easily be added or removed. Even a 20-slot backplane may not be big enough depending on the complexity of the system and its degree of autonomy. Multiple multislot backplanes may be desirable.

3.3. Subsystem bus capability.

In addition to a high-bandwidth main bus connecting processors, it is useful to have subsystems buses that allow subgroups of processors to communicate separately from the main bus. For example, the processors performing centralized control of an arm need to communicate more frequently with each other than they do with the processors controlling the other arm, in general. This communication is local to the arm processors and traffic on the main bus can be reduced by giving the arm processors a separate subsystem bus. The arm processors can still use the main bus, but use the subsystem bus for communication within the subsystem. Since the values transferred over the subsystem bus are for the most part 32-bit, the subsystem bus should also have a 32-bit data path.

3.4. Industry standards.

Any computer architecture design should try to use industry standards where possible. Compliance to industry standards reduces development costs, improves reliability, and increases the compatibility with current and future products. This feature, along with items 3.1-3.3, indicates that a good choice for the coupling processors would be an industry standard 32-bit address and data bus, such as Multibus-II or VME bus, configured as a multislot backplane. This type of architecture has been in use for many years and is well tested.

With respect to the distributed part of the control system, i.e., the part that resides in the manipulator arms, it is our understanding that there are two major problems. The first problem is ca-
bling. It is desirable to minimize the number of cables passing through a manipulator. The second problem relates to thermal control within a manipulator. The amount of heat generated within an arm should be kept to a minimum. With these points in mind, here are some features that may be desirable for the distributed part of the hardware architecture.

3.5. Only local control at the joints.

The nature of the control problem for a serial mechanism such as a robot manipulator requires a centralization of control for coordination. An example of this is the inertia decoupling compensation required for FTS. This decoupling can only be done for the manipulator as a whole, not by each joint independently. Any attempt at joint-local decoupling will result in high communication requirements between joints, defeating the whole purpose of the local joint control. Thus, it is important that any control distributed to joint controllers be truly local. For the FTS this means that only joint torque control loops should be done in joint-mounted electronics.

3.6. Digital communication bus in arm.

The large number of sensors (current, temperature, etc.) and the desire to reduce cabling leads to the conclusion that sensory data should be digitally encoded and transmitted over a common link. A simple hardware link, such as a twisted-shielded pair, may be best for this purpose since the cable will need to be quite flexible to minimize cable friction. This need, along with item 3.4, may indicate that 1553 is an appropriate choice. The bandwidth required for this link should not be that great provided the link only transmits sensor feedback and reference signals for local control at the joints. If additional bandwidth is needed, an additional digital bus of the same type can be added to the manipulator.

3.7. Programmable torque loops.

The widely varying thermal conditions in which the FTS will operate, and the inability to test it fully in real space environments, make it likely that the torque loop parameters will need to be modified after the FTS is constructed. In addition, if the FTS is to operate for a number of years on the space station, wear and other aging factors will also require that control parameters change. Thus, the torque loops operating on the joints should be programmable in that all the control parameters could be modified. The most flexible way to do this is to implement digital torque loops on a general microprocessor, although it may be just as easy to implement the torque loops using a microcontroller or programmable gain amplifiers. Not using general microprocessors may be the better approach in minimizing size and heat generated at the joints.


To minimize cabling complexity the motor power for the various joints in a manipulator should all be taken off of a single power bus. This implies that the PWM electronics for providing motor currents should be distributed in the manipulator. The motor command for the PWM’s will be taken from the output of the torque loops.

Considering all of the above points, the telerobot hardware architecture for a manipulator/handcontroller Servo Level pair is obtained as depicted in Figure 3. (For further validation of this basic

Approach to Computing Arch.
Figure 3. Hardware Architecture for Manipulator/Hand Controller Pair.
approach, consider the hardware architecture developed independently for MARCS in Appendix 1.) In this design, all of the general processors reside in a common backplane. These processors communicate with each other over Multibus II using common memory residing on the shared bus. The number of general 386 processors is seven, which is chosen to correspond with the number of microprocessors used by Martin Merietta for a manipulator/hand-controller pair. The processors communicate with device-mounted electronics over the 1553 serial data lines.

The device-mounted electronics include all the electronics necessary to receive the digital torque commands from the 1553 and produce the corresponding actuator motions. In addition, these electronics interface the sensors to the digital bus for sensor feedback. For the manipulator, the device-mounted electronics could be physically located within the links of the manipulator arm. The electronics may include, in addition to the 1553 interface, power electronics, commutation electronics, PWM electronics, torque control loops, and analog i/o hardware. It is assumed that general microprocessors are not used in the device-mounted electronics.

Referring back to Figure 2, all the processes depicted in the figure reside on the seven general-purpose processors in the multislot backplane. Thus, the communication at the "bottom" of the Servo Levels, (joint torques and feedback in Figure 2,) is over the 1553 bus. By restricting 1553 data to be this low-level data only, the ability of the 1553 bus to meet the communication requirements is assured. For example, if seven 16-bit joint torques are sent, seven 32-bit joint positions and six 16-bit force/torque values are received, this requires 27 16-bit words. With the 1553, each 16-bit word is transferred at a cost of 20 bits, for a total of 540 bits. Add a few protocol words and the total transfer is about 600 bits. If this data is transferred every millisecond then the required bus rate would be 600Kbits/s. This rate is well within the capabilities of the 1553 which has a specification for 1 Mbit/s. If an actual 1553 bus only achieved 80% of this (800Kbit/s), the transfer of 600 bits would still take only 0.75 ms.

Now consider the execution times presented in Table 1, and note the following. Processes 4, 5, 6, 17, and 18 can run at a fairly slow rate with respect to the rest of the system and not effect control system performance. These processes need not update their outputs more than about 20 times a second. There are numerous ways to distribute the remaining processes to processors. The system is completely flexible in this respect since all processors are tightly coupled. Processes can be distributed to minimize the around-the-loop time of the teleoperation control, maximize the processing margin, minimize the time between new updates of joint torques, or optimize a number of other factors.

Suppose, as an example, a minimum around-the-loop time for the force-feedback teleoperation control is desired. Note that the around-the-loop time is defined by the process sequence {2, 8, 9, 13, 14, 15, 20, 19, 16, 10, 11, 12}. By grouping processes 4, 5, 6, 17, and 18 on one processor, each of these processes would compute a new output at around 37 Hz. The remaining processes can be distributed to processors as follows.

CPU-1: 4, 5, 6, 17, 18
CPU-2: 1, 2, 8
CPU-3: 9, 13, 14
CPU-4: 15, 20
CPU-5: 19, 16, 10, 11

Approach to Computing Arch.
CPU-6: 12
CPU-7: 7,3

Here, each processor executes the processes in the order listed. Processors 2, 3, 4 and 6 can each execute their processes in less than a millisecond. The reader may verify this by adding the execution times from Table 1. Processor 5 can execute its processes in less than 4 ms, while processor 7 takes less than 3 ms. Thus, all the processors can be synchronized on a one-millisecond boundary so that outputs of processes 1, 2, 8, 9,13,14,15, 20, and 12 get updated every millisecond, outputs of processes 7 and 3 get updated every 3 ms, and the outputs of processes 19, 16, 10, and 11 get updated every 4 ms. The around-the-loop time is determined by the processor sequence {CPU-2, CPU-3, CPU-4, CPU-5, CPU-6}. Adding up the times for this sequence (1+1+1+4+1=8), and making the mild assumption that the additional time-delay for the 1553 and the device-mounted electronics is less than 1 ms per combined torque/feedback transaction, the around-loop-time is 10 ms. This loop has pipelined updates coming every millisecond. In addition, the example achieves a 333 Hz local joint control rate, and a 100 Hz impedance loop rate.

The important point is that the hardware architecture provides many options in terms of allocating computing resources. Other processing resource allocations can easily be made without changing either the hardware or software architectures. For example, the distribution

CPU-1 : 4,5,6,17,18
CPU-2 : 2,8,9,13,14,15,20
CPU-3 : 19,16
CPU-4 : 10,11
CPU-5: 1,12
CPU-6: 7,3

with processors 2, 3, and 4 repeating every 2.5 ms, processor 5 repeating every 1.25 ms, and processor 6 repeating every 3.75 ms, results in an implementation with a 14 ms teleoperation around-the-loop time and a 400 Hz local position loop on the manipulator. And this is with a spare processor left over. It is possible to implement the whole system on just five processors and still achieve good performance by making the distribution

CPU-1 : 4,5,6,17,18
CPU-2 : 1,2,8,9,13,14,15,20
CPU-3 : 19,16
CPU-4 : 10,11,12
CPU-5: 7,3.

The examples show the tradeoffs in performance for the implementation of the Martin Marietta algorithm that can be made for the computing architecture approach of Figures 2 and 3. It should be noted, however, that the Martin algorithm has not been shown to be superior in any way to other approaches. In fact, Martin's use of explicit inverse kinematics in the control loop ensures that the algorithm can not be used for arms with more than six degrees of freedom, or even with six degree-of-freedom arms without simple kinematics. This has been the conclusion of both the NIST and

Approach to Computing Arch.
Goddard labs. Thus, it is important that the control system design for a telerobot be able to run more than just the Martin Marietta algorithm. The system should be easily extensible to run other algorithms and additional levels of the telerobot functional architecture.

4. Extensibility

As discussed in Section 2, the complete telerobot architecture involves many more components than the Servo Levels analyzed in Sections 2 and 3. Some of these components are depicted in the partial telerobot architecture of Figure 4.

Primitive Levels, as depicted in Figure 2, connect the manipulator and hand-controller Servo Levels to the upper levels of the telerobot. These Prim Levels generate trajectories and autonomous reference commands for the manipulator and hand-controller. There must be a Prim and an E-move Level for autonomous manipulator operations such as automatic end effector exchange. Thus, the processes which realize these upper levels must be given computing resources within the system. Note that, in general, resources cannot be taken away from the Servo Level processes since many of the processes must remain in operation when the upper levels are active. (The E-move, Prim, and Servo Levels form a pipeline for hierarchical control.)

With the hardware design of Figure 3, extra slots are available to add processors for the upper levels. Some processors from the seven original ones may also be available to implement the upper levels depending on how processes are allocated to processors at the Servo Level. Extra processors...

![Figure 4. Partial Telerobot Control System.](image-url)
can also be made available to execute other subsystems of the telerobot such as Safety, Positioning and Perception.

With respect to Perception, if the telerobot is to eventually have the capability to analyze images, then it must be possible to add computing resources to the telerobot to perform this function. Generally, specialized hardware is used to perform the computationally expensive function of converting the input intensity image from an iconic (spatially based) image to a symbolic (feature based) image. Additional general support computers process the symbolic image to complete the perception task. A number of specialized hardware systems exist for performing image analysis including PIPE, WARP, PIFEX, and DATACUBE. Such specialized hardware should be easily incorporated into the control system.

With the hardware design of Figure 3, the specialized vision hardware can be given a dedicated processor in multislot backplane. The approach which has been successful in the NIST laboratory has been to use a PIPE interface board along with a general processor to control and configure the PIPE. This allows the processor to use the PIPE like an accelerator to speed up vision computation. The addition of the PIPE has little effect on the performance of the rest of the control system.

5. Conclusion

The design approach presented here represents an accumulation of knowledge from many years of work at NIST. This approach is well-documented in the references which follow. The design presented here supports flexibility in control system performance, a variety of control algorithms, and provides an evolutionary path to additional telerobot functionality.
6. Bibliography

6.1. General


6.2. Software Related


6.3. Hardware Related


Center, Greenbelt, MD, 1988.


Appendix 1.

The Multiple Algorithm Robot Control System (MARCS) is currently under development in Goddard Space Flight Center’s robotics laboratory. This system is designed to run three different teleoperation algorithms, a Martin Marietta-style algorithm, a JPL teleoperation algorithm, and an algorithm developed at Goddard. The following pages are taken from early design efforts in the project and are presented here only to show a consensus of viewpoint among two labs working on telerobot systems development. The first two figures show the Servo Levels for the manipulator subsystem and the hand-controller subsystem. The last figure shows a preliminary hardware architecture to support these two Servo Levels.
Essential MARCS Hardware Components

RRC Slave Arm
- Motors
- Analog Joint Servos
- Resolvers
- Force Torque Sensor
- Intelligent Gripper & Controller Board

HARPS
- User Interface
- Safety System
- Servo Amplifier Boards #1-7

Force Torque Sensor Electronics Instrumentation

1 CPU required in MBI or MBII

RRC Controller Box
- MBI Card Rack
  - "N" CPUs:
    - 6 or 7 ISBC 386s, 20 MHz
    - 4 ISBC 386/s, 33 MHz
    - 2 ISBC 486/s, 25 MHz

  - ISBC ?, ?, MHz
  - ISBC 386, ?, MHz with Kraft Serial Interface Converter
  - CSM

  - iSBC 386/31 SBC, 20 MHz
  - iSBC LNK Board or an iSBC 386/31 with a parallel, or High Speed serial interface

  - Resolver Board D/A Chs 1-4
  - Resolver Board D/A Chs 5-7
  - A/D Board

  - Analog Card Rack
    - Servo Comp Board Chs 1-4
    - Servo Comp Board Chs 5-7
    - Watchdog Board

  - Power Distribution Board

Notes:
1. = Not yet defined
2. Most connections not shown for clarity
In response to a Goddard Space Flight Center effort to look at the requirements for a computing architecture design appropriate for a telerobot, this document describes the Intelligent Controls Group approach to telerobot computing architectures. It is shown how a seven microprocessor control system can achieve teleoperation with an around-the-loop time of 10 ms. The document focuses on low-level, real-time robotics and does not discuss some issues relevant to space systems such as space-qualification and thermal design.