January 1990

A Distributed System for Robot Manipulator Control, NSF Grant ECS-11879 Fourth Report

Richard P. Paul
*University of Pennsylvania*

Peter Corke
*University of Pennsylvania*

Janez Funda
*University of Pennsylvania*

Gaylord Holder
*University of Pennsylvania*

Hiroaki Kobayashi
*University of Pennsylvania*

See next page for additional authors

Follow this and additional works at: [http://repository.upenn.edu/cis_reports](http://repository.upenn.edu/cis_reports)

Recommended Citation


This paper is posted at ScholarlyCommons. [http://repository.upenn.edu/cis_reports/818](http://repository.upenn.edu/cis_reports/818)

For more information, please contact repository@pobox.upenn.edu.
A Distributed System for Robot Manipulator Control, NSF Grant ECS-11879 Fourth Report

Abstract
This is the fourth annual report representing our last year's work under the current grant. This work was directed to the development of a distributed computer architecture to function as a force and motion server to a robot system. In the course of this work we developed a compliant contact sensor to provide for transitions between position and force control; developed an end-effector capable of securing a stable grasp on an object and a theory of grasping; developed and built a controller which minimizes control delays; explored a parallel kinematics algorithms for the controller; developed a consistent approach to the definition of motion both in joint coordinates and in Cartesian coordinates; developed a symbolic simplification software package to generate the dynamics equations of a manipulator such that the calculations may be split between background and foreground.

Comments

Author(s)
Richard P. Paul, Peter Corke, Janez Funda, Gaylord Holder, Hiroaki Kobayashi, Yangsheng Xu, and Yehong Zhang

This technical report is available at ScholarlyCommons: http://repository.upenn.edu/cis_reports/818
A Distributed System for Robot Manipulator Control
NSF Grant ECS-11879
Fourth Report

Richard P. Paul
Peter Corke  Janez Funda  Gaylord Holder
Hiroaki Kobayashi  Yangsheng Xu
Yehong Zhang

The University of Pennsylvania
Moore School
Philadelphia PA 19104

ABSTRACT

This is the fourth annual report representing our last years work under the current grant. This work was directed to the development of a distributed computer architecture to function as a force and motion server to a robot system. In the course of this work we developed a compliant contact sensor to provide for transitions between position and force control; developed an end-effector capable of securing a stable grasp on an object and a theory of grasping; developed and built a controller which minimizes control delays; explored a parallel kinematics algorithms for the controller; developed a consistent approach to the definition of motion both in joint coordinates and in Cartesian coordinates; developed a symbolic simplification software package to generate the dynamics equations of a manipulator such that the calculations may be split between background and foreground.
Contents

1 PAST RESEARCH ......................................................... 1

2 CURRENT RESEARCH .................................................. 2
  2.1 Array Processor Software Tools ................................ 2
  2.2 Quaternions ....................................................... 3
  2.3 Instrumented Passive Compliant Wrist Manipulator Control . 5
  2.4 Redundant Wrist Design .......................................... 6
  2.5 Redundant Manipulator Control ................................ 6
  2.6 Visual Servoing .................................................... 7
  2.7 Stability of Systems with Random Communication Rates .... 7
  References ............................................................. 8

3 DOCUMENTATION ...................................................... 8

A APPENDICES ............................................................ 13
  A.1 A General Device Driver for Ultrix: dev/bus .................. 14
  A.2 A New Approach to Laboratory Motor Control: MMCS ........ 23
  A.3 Simulator for the Systolic Array Processor .................... 74
  A.4 Implementation of Faddeev’s Algorithm ......................... 96
  A.5 Compliant Wrist Design ........................................... 124
  A.6 Hybrid Control in the Presence of Passive Compliance ...... 146
  A.7 Decoupling Mechanisms for Robot Manipulators ............... 163
  A.8 Nonlinear Control of a Robot with a Passive Compliant Wrist 182
  A.9 The Hamilton Wrist ................................................. 194
  A.10 Control of a Seven Joint Manipulator ......................... 218
  A.11 Visual Servoing of Robot Manipulators ....................... 235
  A.12 Stability of Robot Systems with Random Sample Rates ..... 249
1 PAST RESEARCH

Research during the prior years of this grant related to the problems of multi-sensor control of robots, sensor fusion, and grasp planning. A distributed computing architecture was proposed in which sensors and actuation controllers run on separate processors coupled together by a network and supervised by a coordinator. The coordinator used Bayesian techniques to cluster sensor observations and to provide a robust estimate of environment state.

Problems of grasp planning were considered together with the design of a new three fingered hand of medium complexity. A contact sensor was developed which was to provide contact detection information, compliance during contact, and relative end-effector displacement.

Actuation was handled by a special purpose, concurrent processor which provided for both force and motion control. The aim of this processor was to remove the computational limitations on manipulator performance. This system involves delays of the order of 5 milliseconds between changes in Cartesian coordinates and a response at the manipulator actuator level. A number of software and hardware tools were developed in the course of this work. Algorithms were carefully studied in order to reduce the real time complexity of manipulator control.

Documentation relating to this work was as follows:

- the integration, coordination, and control of multi-sensor systems [1] [2] [3] [4] [5] [6];
- grasping [7] [8] [9, 10, 11] [12]; a four joint wrist design [13] [12];
- the initial development of the distributed force and motion server [14] [15] [16, 17][18];
- research on general manipulation and dynamics [19] [20] [21] [22] [23] [23, 24, 25]; the development of a passively compliant instrumented wrist [26, 27].
2 CURRENT RESEARCH

Research during the current year of this grant, in the absence of the Hughes Systolic Array Processor, was in the areas of software and hardware development, representations of displacement, the use of a passive, instrumented, compliant wrist, redundant wrist design and control, visual servoing, and finally an analysis of stability of control systems with random rates.

We have developed a systems software package to map devices into user memory space for direct, simple robot manipulator control. We have also developed a hierarchical software motor control package allowing the separation of control from specific hardware configurations. With regard to the systolic array processor we have developed an assembler and simulator for the processor. We have also written the Faddeev algorithm and checked it with the simulator. While the systolic array processor is ideal for matrix operations we have also investigated the use of quaternions and line based coordinates showing that they are computationally equivalent to their matrix equivalents. We have moved ahead on the investigation of the use of a compliant wrist. This device, located at the end of the manipulator, provides for a low pass filter on environmental interactions. It provides for stable compliance control and for hybrid control. Two devices have now been built. We have built a four joint wrist and developed control algorithms to avoid kinematic singularities which severely limit the usefulness of manipulators. Based on a dedicated visual processor we have achieved video rate visual servoing of a manipulator. Finally we investigated the stability of systems subject to random control rates.

2.1 Array Processor Software Tools

Without the right tools, experiments and hence applied research doesn’t advance very much. One of the major tools developed during this grant has been that of /dev/bus. The development of /dev/bus gave user processes on MicroVAX IIs the ability to access and control hardware on the system bus. See Appendix A.1

Sun Micro Systems, Masscomp, and IBM all supply a version of Unix which allows a user process direct access to hardware on the system bus. However, the Unix for MicroVAX IIs supplied by Digital Equipment Co. and which runs our robot control software, does not. Thus, interfacing new
equipment necessitated recompiling the operating system, slowing projects and experiments.

With the implementation of /dev/bus, robot control programs now have fast access to frame buffers, parallel communications, Analog-to-Data Converters. We no longer have to bring an entire machine down while the software interface to new hardware is being coded. This has greatly simplified the designed of robot control experiments.

Many projects within the GRASP laboratory involve motion control via electric servo motors, for example robots, hands, camera mounts and tables. To date each project has been based on a unique hardware/software approach with associated development problems and delays. MMCS[28] see Appendix A.2 is a new modular, and host independent, motor control system for laboratory use.

One additional development is a 2 axis motor interface board that can be plugged into an IBM PC bus. Also developed is an adaptor that connects an IBM PC bus to a SUN workstation, allowing IBM PC cards to appear in the SUN's memory space. Device driver software provides a powerful and general way to control the closed loop performance of each axis, under application software control.

Preliminary experiments indicate that a SUN workstation can perform 6 axis servo control at rates exceeding 500Hz whilst still providing a Unix windowed environment for program development and execution.

While we are still waiting for the Hughes Processor we have developed both a symbolic assembler, see Computer and Information Science Department Report MS-CIS-88-39 or GRASP Lab Report 143, and a simulator, see Appendix A.3.

Much of robotics computing involves matrix operations and this generally becomes the computation bottleneck in problem solving. Faddeev's algorithm provides not only a method general enough to deal with a large variety of matrix operations but also simple enough for systolic implementation and forms the basic algorithm that the Hughes Processor will execute. See Appendix A.4.

2.2 Quaternions

Three-dimensional modeling of rotations and translations in robot kinematics is most commonly performed using homogeneous transforms. We inves-
tigated an alternate approach, employing quaternion/vector pairs as spatial operators, and compared it with homogeneous transforms in terms of computational efficiency and storage economy. The two formalisms were compared in terms of the computational speed in performing a spatial transformation of a point vector, composition of two such operator, computing the corresponding inverse transformation, and normalizing the rotational part of the operator. Both sequential and parallel implementations of the corresponding algorithms were considered. The results of our analysis suggest that the two formalisms are virtually equivalent for the case of non-normalizing sequential applications, i.e., the case where computations are carried out on a single-processor machine and the rotational operators are not normalized prior to being used in the computations. We also observed that the non-normalizing algorithms based on homogeneous transforms parallelize slightly better than their quaternion/vector counterparts. If the cost of normalizing the rotational operator is included in the total cost, however, the quaternion/vector pair approach yields much more efficient implementations on both single and multi-processor systems. This is due to the fact that a quaternion can be normalized with a minimal amount of computational expense, whereas normalization of a rotational matrix requires substantially more effort. In summary, it is our conclusion that quaternion/vector pairs are as efficient, more compact, and mathematically more elegant than their matrix counterparts.

We further presented a robust algorithm for converting rotational matrices into equivalent unit quaternions, which exhibits stability in the neighborhood of rotational singularities. As a demonstration of power and elegance of quaternion algebra, we also developed an efficient quaternion-based inverse kinematics solution for the Puma 560 robot arm, see Computer and Information Science Department Report MS-CIS-88-06 or GRASP Lab Report 133.

Further investigations of available mathematical models of spatial transformations led us to consider the set of Plücker coordinate based screw transformations. We performed a similar analysis as above of the four most commonly used line-oriented formalisms for representing and effecting spatial screw displacements of rigid bodies. The formalisms analyzed were: dual 3x3 orthogonal matrix, dual unit quaternion, dual unitary 2x2 matrix, and dual Pauli spin operators. The analysis and comparison was again based on the computational efficiency in performing common operations needed in kinematic analysis of multi-linked spatial mechanisms, i.e., transformation of
a line in space, and composition of two such successive transformations. We found that the dual unit quaternion representation offers the most compact and most efficient screw transformation formalism, but that line-oriented methods in general are not well suited for efficient kinematic computations. The mathematical redundancy inherent in Plücker coordinate based screw operators makes them computationally less attractive than the corresponding point-oriented formalisms, mentioned above. See Computer and Information Science Department Report MS-CIS-88-83 or GRASP Lab Report 159.

2.3 Instrumented Passive Compliant Wrist Manipulator Control

Most industrial robots today are utilized to perform tasks in which the end-effectors are in contact with the environment. It is becoming increasingly clear that robots require a more sophisticated compliant motion. A new compliant motion methodology combining passive compliance and active control together has been developed.

The compliant wrist which is installed between the end-effector and the robot was developed in the GRASP lab. The wrist consists of two portions: a passive compliance element and a sensing mechanism [29], [30] [31]. The passive compliance provides an adaptation for assembly operations and manufacturing processes so that the positioning tolerances are relaxed and the high forces normally produced in jamming or wedging are reduced. The sensing information is used two ways. In position control, the sensed information is utilized to compensate deflection of the wrist, due to the load or external forces, so as to increase apparent stiffness of the manipulator wrist system. In force control, the wrist sensor is used as a force sensor by which means the manipulator is driven in the same direction as the sensed force and the desired contact force is maintained. See Appendix A.5. Various experiments in force and position control demonstrated its applicability [31] [32]. Transition as the robot makes contact with or breaks away from the surface is accommodated by the method presented [32]. Based on these algorithms, a new hybrid control strategy has been derived. See Appendix A.6. As its application, a sinusoid surface tracking experiment has been performed. The effects of various conditions, such as digital filter, force gain, position compensation gain, desired contact force, environmental characteristics, and passive
damping in the device, on the system performance have been analyzed and demonstrated by experiments [32].

A systematic approach to design of a decoupling compliance mechanism has been studied and the results are useful not only for a passive compliance design but also for parallel manipulators compliant control [33] [34]. See Appendix A.7. For a complete dynamic control of a robot with a compliant wrist, we have built a dynamics model for the entire system including link elasticity, actuator characteristics, and compliant wrist and applied the nonlinear feedback control theory to the system, and thus a complete dynamic hybrid control has been studied [35]. See Appendix A.8.

2.4 Redundant Wrist Design

When a manipulator loses one or more kinematic degrees-of-freedom, there are directions in which the manipulator cannot apply forces and/or moments — these configurations have been termed configuration singularities. Configuration singularities are inherent in the kinematic structure and the type of joint-actuators used in the manipulator. For example, when revolute and prismatic joints are used, it can be shown that singularities always exist. As Fisher [1984] has shown, the configuration singularities in the traditional 3-R spherical wrist can be avoided only when an additional joint is added to the wrist.

The screw theory of mechanics [Ball, 1900] provides a framework for analyzing configuration singularities in a compact manner. In this paper, we study the kinematics of a serial chain 4-R spherical wrist from the viewpoint of screw theory. Fisher's original solution for the 4-R spherical wrist is developed further by integrating the geometry of the wrist into the control scheme. See Appendix A.9

2.5 Redundant Manipulator Control

Most redundancy control methods require the computation of the pseudoinverse of the Jacobian matrix. The large computation load makes the real time control of a redundant manipulator impossible. This paper presents a redundancy control algorithm for avoiding singularities of the 6 DOF manipulator. It is based on the partitioning of the Jacobian matrix. The formulation is computationally simple since it only requires the inversion of a 6x6 matrix.
The control is applied to a seven degree of freedom robot manipulator with a 4 joint spherical wrist built at the GRASP Lab. Two types of simulation results are presented. The first plots trajectories of the manipulator and the second is a video tape using graphical packages. Both simulation results confirm that the method can be used successfully to solve the singularity problem of the 6 DOF manipulator. See Appendix A.10.

2.6 Visual Servoing

Some experiments in real-time sensor based robot control were conducted in which the robot position loop was closed visually[36]. Using newly available binary image, and existing grey-level processing hardware in conjunction with two workstation computers, the robot position loop was closed at video field rates, 60Hz. The control strategy is very different to the usual approach using explicit trajectory generation, and more closely resembles a feedback control system. see Appendix A.11.

2.7 Stability of Systems with Random Communication Rates

Robot systems are now based on distributed computer networks with sensing on one or more computers, sensor fusion of another, and actuation on yet another computer. Communications is accomplished by means of a shared network. Such a network involves random delays. The work reported here relates to the stability of such systems under various sampling rate distributions. See Appendix A.12.
3 DOCUMENTATION


[19] R. P. Paul and Hong Zhang. Robot motion trajectory specification and
generation. In Hideo Hanafusa and Hirochika Inoue, editors, *Robotics
Research: The Second International Symposium*, pages 373–380, MIT

manipulators with spherical wrists based on the homogeneous transfor-
5(2), 1986. Special Issue on Kinematics.

[21] Alberto Izaguirre and R. P. Paul. Computation of the inertia and gravi-
tational coefficients of the dynamic equations of the robots. In *Proceed-
ings of the IEEE International Conference on Robotics and Automation*,

dynamics equations of the robot manipulators using a LISP program.
In *Proceedings of the IEEE International Conference on Robotics and

[23] Alberto Izaguirre, Minoru Hashimoto, Richard P. Paul, and Vincent
Hayward. Identification of the parameters of the dynamic equations of
robot manipulators. In *IEEE International Workshop on Robotics: Trends,

Hayward. A new computational structure for real time dynamics. In
*Identification of parameters in dynamics*, S.I.C.E. Conference, Hiroshima,
JAPAN, July 1987.

Hayward. A new computational structure for real-time dynamics. *The
International Journal of Robotics Research*, 1988. accepted for publica-
tion.

[26] Yangsheng Xu and Richard P. Paul. On position compensation and
force control stability of a robot with a compliant wrist. In *Proceed-
ings of the IEEE International Conference on Robotics and Automation*,


A.1 A General Device Driver for Ultrix: dev/bus
A General Device Driver for Ultrix

or

Leave the Driver to /dev/bus

Gaylord Holder
General Robotics and Active Sensory Perception Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389
(holder@grasp.cis.upenn.edu)

Abstract

New hardware is the bane and the boon of the research laboratory: the boon because it brings new power, new capabilities, and new solutions; the bane because it means someone has to sit down and write the interface code for the board. /dev/bus attempts to simplify the process on the MicroVAX IIIs running Ultrix 2.0 by allowing user processes direct access to the board's control status registers and Q-Bus memory. Unlike similar drivers for Suns, RTs and Masscomp, /dev/bus provides a means of establishing a user function as an interrupt handler. The delays and variability of the interrupt delivery are analyzed. Problems with the implementation are also described.

Introduction

Inspired by device drivers for the IBM RT and Sun which allowed user processes direct access to the bus, /dev/bus brings this capability to Ultrix 2.0 on the MicroVAX. After opening /dev/bus, a process can request access to both the Q-Bus I/O space and the Q-Bus memory. In addition, the driver can signal a process when an interrupt comes in on a vector.

The General Robotics and Active Sensory Perception Laboratory (GRASP) at the University of Pennsylvania has been using /dev/bus for the past two years to provide user processes with access to:

- frame buffers.
- parallel I/O.
- analog to digital data acquisition.
- real-time clock for timing analysis.

/dev/bus has proved to be reliable, easy to use and versatile.

---

1 This work was funded in part by National Science Foundation grant number DMC-8512838. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author and do not necessarily reflect the views of the National Science Foundation. The author wishes to thank Robert King and Filip Fuma for their help and support in this work.

2 The MicroVAX II uses a private bus for system memory which /dev/bus doesn’t touch. Memory on the Q-Bus is usually provided by hardware such as frame buffers. This is the Q-Bus memory that /dev/bus provides access to.
The rest of this paper describes the user interface and implementation of /dev/bus.

**User interface**

After the user process opens /dev/bus, it may request a pointer to the Q-Bus I/O space, Q-Bus memory, or install an interrupt handler by calling the functions: `bus_getio()`, `bus_getqmem()`, and `bus_sethand()` respectively.

**bus_getio()**

The function `bus_getio()` is used to obtain a pointer the Q-Bus I/O space. The bus address of a device is added to the pointer to reference the device's CSRs (Control Status Registers). NULL is returned on error.

```c
  caddr_t bus_getio(fd);
  int fd;
```

The CSRs for the DRV-11C 16-bit parallel communications board from Digital Equipment Corporation consist of a control word, output buffer, and input buffer. A structure similar to the following may be used to interface with the DRV-11C:

```c
typedef struct {
    u_short csr;        /* Configuration word */
    u_short obuf;       /* Output buffer */
    u_short ibuf;       /* Input buffer */
} drv_11c;
```

If the DRV-11C lives at DRV_ADDR on the Q-Bus, it may be accessed by:

```c
  drv_11c *drv_p;     /* Pointer to DRV-11C's CSRs */
  
  drv_p = bus_getio(fd);
  drv_p = (drv_11c*)((u_int)(drv_p) + DRV_ADDR);
```

where `fd` is file descriptor returned from a call to open "/dev/bus". If `bus_getio()` should fail, it returns NULL.

**bus_getqmem()**

The function `bus_getqmem()` is used to reference memory on the MicroVAX's Q-Bus such a memory in a frame-buffer. A pointer to the Q-Bus memory is obtained by passing the /dev/bus file descriptor, the beginning address of the desired Q-Bus memory, and the size of the memory in bytes to `bus_getqmem()`.

```c
  caddr_t bus_getqmem(fd, qbus_addr, nbytes);
  int fd;
  caddr_t qbus_addr;
  int nbytes;
```

The argument, `nbytes`, must be a multiple of 512. NULL is returned on error.
The Data Translation 2651 is a frame-grabber with two 512 by 512 by 8 bit frame buffers.

```c
typedef struct {
    u_char fba[512][512]; /* Frame buffer A */
    u_char fbb[512][512]; /* Frame buffer B */
} dt_fb;
```

Given that it is at DTQMEMADDR, the following code can be used to access the DT2651's memory.

```c
dt_fb *fbp; /* Ptr to DT2651 frame buffer memory */
fbp = bus_getqmem(fd, DTQMEMADDR, sizeof(dt_fb));
```

Again, `fd` is the `/dev/bus` file descriptor.

**bus_sethand()**

A user process may attach a function to an interrupt vector with `bus_sethand()`. The `/dev/bus` descriptor, pointer to the function, and interrupt vector are all passed to `bus_sethand()`.

```c
int bus_sethand(fd, fn, vec)
    int fd;
    int (*fn)();
    int vec;
```

`bus_sethand()` returns -1 on error.

After a interrupt hander has been installed with `bus_sethand()`, `/dev/bus` will send a SIGINT signal to the user process whenever an interrupt is asserted. `bus_sethand()` takes care of calling `signal()` so `(*fn)()` will be executed with the driver delivers the signal.

The DRV-11C can be configured to interrupt the CPU whenever a word is sent and/or received. A simple(-minded) scheme to count the number of times a device at the vector VEC_ADDR interrupts is shown below.

```c
int interrupt_counter = 0;

interrupt_handler()
{
    ++interrupt_counter;
}

main()
{
.
.
.
    bus_sethand(fd, interrupt_handler, VEC_ADDR);
}
Because of the time required to process signals, this is not a highly reliable scheme if the
device can interrupt more frequently than 500 times a second.

Implementation

This section provides a brief outline of the /dev/bus driver.

Initialization

The first thing the driver does when /dev/bus is opened is to lock the process in memory. This
keeps the kernel from trying to do something stupid like swap frame buffer memory a
process is accessing out to disk. In addition, open() allocates a data structure the driver
will use to keep track of Q-Bus memory the process is accessing, the process id, and
interrupt vectors the process is handling, so the process's state can be restored when it exits
or closes /dev/bus.

Q-Bus Access

bus_getio() and bus_getqmem() work in pretty much the same way. The major
difference is that bus_getio() returns a pointer to the Q-Bus I/O space, addresses
0x20000000 to 0x20001FFF, while bus_getqmem() works with the address from
0x30000000 to 0x303FFFFF.³ A single common driver routine, bus_qmem(), is
called by both bus_getio() and bus_getqmem(). It takes a pointer to the process
structure, the beginning address of the the Q-Bus memory to be mapped in, and the number
of bytes to be mapped.

bus_qmem() calls expand() to add PTEs (Page Table Entries) to the user process. The
original value of the PTEs are saved so they can be restored later. The new PTEs are
changed to reference the appropriate Q-Bus page frames and the pointer to the Q-Bus
memory is returned to the user process. The only funny business about this whole thing is
that it requires the swap space associated with the process also be expanded, otherwise, the
operating system panics, thinking it somehow grew a process without remembering to
adjust the swap space.

Interrupt Handling

When a user process establishes a signal handler, the /dev/bus driver records the vector in
the data structure that it allocated the process when /dev/bus was opened. A small change
to the assembler routine _stray in locore.s allows /dev/bus to pass device interrupts
to interested processes.

When a device for which the operating system is not configured interrupts, _stray picks
off the interrupt vector and interrupt priority. _stray was changed to call the /dev/bus

³ MicroVAX Handbook, pp 5-34 to 5-36, Digital Equipment Corporation, Nauhua, N.H.
function, `bus_sigintr()` with the interrupt vector as an argument. This function runs though the /dev/bus data structures looking for a process which wants to handle it. If none is found, the `bus_sigintr()` returns and _stray logs the stray interrupt. If a process has established an interrupt handler for the vector, `bus_sigintr()` calls `psignal()` to deliver a SIGINT signal to the process.

**Interrupt Latency**

One of the original uses planned for /dev/bus was to provide a real time capability for user programs. The idea was that a device could interrupt the user process which would do its thing. This would be great for robot control since all the code would be running in user space on a single machine. Standard debuggers could be used to make sure the code worked and life would be just peachy. The big question was how long it took the interrupt to wend its way through Ultrix and kick in the user's interrupt handler and how variable the times were. An experiment was set up to learn how long it took until the kernel got a-hold of the interrupt, and then how long it took to pass to the user process.

A KWV-11 real time clock board was set to run at 2 MHz. It was then set to wait 2 milliseconds and then interrupt. The time between the interrupt and resetting the KWV-11 was recorded. The interrupt routine would set the KWV-11 to interrupt in 2 milliseconds to give the interrupt handler to return and restores the process's normal context. A total of 10 trials of 10,000 samples were run with the interrupt handler in the kernel and from the user process. The trials were run on a MicroVAX II with 5 Mb of memory, a quiet network, standard user priority and a version of Ultrix 2.0 modified for /dev/bus.

As figure 1 shows, the interrupt latency for the user process averages to about 0.7 milliseconds while the kernel latency, as shown in figure 2, is about 0.12 milliseconds. The MicroVAX's 10 millisecond clock shows up in figure 2 quiet clearly.

![Interrupt Latency](image)

**Figure 1: Interrupt to User Process Latency**

The kernel interrupt is being called lowest hardware priority. However, the only thing that it do to bring in the user process and still clear the interrupt stack (remember, there isn't a clean way to get back from the user process context to the kernel interrupt stack) is to set up an AST (Asynchronous System Trap), which is essentially what Ultrix's signaling mechanism does, or call `psignal()` as /dev/bus already was.

-5-
From these, and other, experiments clearly showed that the variability of the interrupt latency is coming after the setting up the AST. Other hardware, kernel housekeeping, network traffic and what not was being handled at the expense of the interrupt handler's reliability and responsiveness.

Although the latency from interrupt to user process was less than 7 milliseconds 90% of the time, see figure 3, the enormous variability destroyed any hope of using /dev/bus for real time work.

Clean Up

The user interface doesn't provide any means of removing an interrupt handler or restoring the process's original memory map, but the kernel must. When a process closes /dev/bus, or it exits, the kernel function `bus_clear_uinfo()` is called to restore any PTEs that /dev/bus used. Any interrupt handlers the process had installed are also removed.

Other Issues
So far, with the exception of the modification of _stray, the implementation of /dev/bus has been completely in the driver module. Unfortunately, there are a few more patches that must be made to get everything working.

More Cleaning Up

The biggest problem /dev/bus has been getting the operating system to call bus_clear_uinfo() at the right time. During the initial design of the driver, it had been assumed that driver's close function would be called before any of the process's memory was de-allocated -- or at least there was a single procedure where processes were dismantled. It was a great surprise to find not only is the process's memory returned to the system's memory pool long before the devices are closed, but also that it is done from several routines.

In the current implementation, bus_clear_uinfo() is called from vmemfree() in ./sys/vm_mem.c.

Protecting the User from Him/Herself

One of the other modules which had to be modified was machdep.c. Without this change, a user process could request memory that wasn't really there, for example the memory address to bus_getqmem() could be wrong. When the process went to use the pointer, it would reference non-existent memory, and the system would panic. A check was added to see if the current process was using /dev/bus. If so, it was assumed to be responsible for problem. This way, a process making an invalid reference exited with SIGBUS rather than taking the whole system with it.

Memory Allocation

One of the bigger surprises was that the sbrk() keeps a local copy of the size of the process in the global assembler variable curbrk. When a process called bus_getio() or bus_getqmem() the size of a process changed without sbrk updating curbrk. Thus, the next time the process decided to print something with printf() (which calls malloc() which can in turn call sbrk()) sbrk() decides to set the process size based on the value in curbrk which is usually a good deal smaller than the real size of the process. The kernel then tries to free the newly allocated PTEs which point to the Q-Bus and panics when it realizes that someone has been tampering with its processes.

To fix this problem, bus_getio() and bus_getqmem() call a routine fix_curbrk() with the number of bytes to add to curbrk. This way sbrk()'s notion of the process's size matches reality.

Problems

There are still some problems.

/dev/bus should provide some support for multi-user access. At the moment, no checking is done to see if the Q-Bus memory or interrupt vector requested by one process is already being used by another.
/dev/bus also ignores problems that might crop up if a process using /dev/bus decides to spawn a child. Since both the interrupt and illegal memory reference features look up a process based on the recorded process id, `fork()` can be a nasty problem. An associated problem has to do with debugging. While a process using /dev/bus can still be run under any of the standard debuggers, care must be exercised when looking a variable values. If the /dev/bus process has a pointer out to Q-Bus memory, it is fine to look at the value of the pointer -- but looking at the contents of what the pointer references sends the whole system into an uproar. This is because the process being run under the debugger had its PTEs mangled by /dev/bus, while the debugger didn't.

The only board with DMA used on the GRASP laboratory's MicroVAXen is the Ethernet Interface. This is not one to use to debug the DMA facilities of /dev/bus, so the DMA support isn't.

Finally, /dev/bus still occasionally will crash the machine. For the most part, the machines are as stable as any other Unix box. But once and a while, especially if there is a system is to be demonstrated, the system will panic in `vrelvm()` or something and away it goes. Sigh.

**Conclusion**

/dev/bus was originally designed so that user processes could interface with new hardware without the operating system overhead or writing a new device driver. On the whole, that goal has been met. New devices can be installed and a rough set of interface libraries written in only a few days. The best part is that usually, the system only has to be rebooted for the physical installation of the device.

While /dev/bus doesn't make device drivers redundant, it does give the system's programmer a chance to work with the device before having to plunge into the kernel. This pulls more of the development work out of the kernel and shortens the time need to write the device driver. Best of all, it gives knowledgeable users the tools needed to write their own interface software.
A.2 A New Approach to Laboratory Motor Control: MMCS
A New Approach to Laboratory Motor Control

MMCS

The Modular Motor Control System

Peter I. Corke\textsuperscript{1}
pic@grasp.cis.upenn.edu

Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104

January 13, 1989

\textsuperscript{1}Research Scientist, CSIRO Division of Manufacturing Technology, Melbourne, Australia.
Abstract

Many projects within the GRASP laboratory involve motion control via electric servo motors, for example robots, hands, camera mounts and tables. To date each project has been based on a unique hardware/software approach.

This document discusses the development of a new modular, and host independent, motor control system, MMCS, for laboratory use. The background to the project and the development of the concept is traced.

An important hardware component developed is a 2 axis control motor control board that can be plugged into an IBM PC bus or connected via an adaptor to a high performance workstation computer.

To eliminate the need for detailed understanding of the hardware components, an abstract controller model is proposed. Software implementing this model has been developed in a device driver for the Unix operating system. However for those who need or wish to program at the hardware level, the manual describes in detail the various custom hardware components of the system.
# Contents

1 Introduction .......................................................... 1  
   1.1 Overview .......................................................... 1  
   1.2 Background and Motivation ....................................... 2  
   1.3 System overview .................................................. 4  
      1.3.1 Control Processor and Software ......................... 4  
      1.3.2 Motor interface ............................................. 6  
   1.4 Acknowledgements ................................................ 7  

2 Application model of the motor controller ....................... 8  
   2.1 Compensator ....................................................... 9  
      2.1.1 The general transfer function ............................ 9  
      2.1.2 PID implementation ....................................... 10  
      2.1.3 PD implementation ......................................... 10  
      2.1.4 PI implementation ......................................... 10  
   2.2 Control options .................................................. 11  
      2.2.1 Velocity servo .............................................. 11  
      2.2.2 Torque servo ............................................... 11  
      2.2.3 Coulomb friction compensation ........................... 11  
      2.2.4 Feedback source ........................................... 11  
      2.2.5 Setpoint source ............................................ 12  
   2.3 The Unix device driver .......................................... 12  
      2.3.1 Configuring the servo ..................................... 12  
      2.3.2 Choice of parameters ...................................... 14  
      2.3.3 Accessing servo state ..................................... 15  
      2.3.4 Error handling ............................................. 16  
      2.3.5 Other device driver functions ............................ 16  
      2.3.6 Code example ............................................... 18  
   2.4 Accessing hardware directly ................................... 18  
   2.5 Control synthesis ............................................... 18  

3 mcTool ................................................................. 24
4 Host adaptor

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1 Introduction</td>
<td>27</td>
</tr>
<tr>
<td>4.2 Adaptors in general</td>
<td>27</td>
</tr>
<tr>
<td>4.2.1 Generic specification for adaptors</td>
<td>27</td>
</tr>
<tr>
<td>4.2.2 PCbus signals redefined</td>
<td>28</td>
</tr>
<tr>
<td>4.3 In particular: VMEbus adaptor</td>
<td>28</td>
</tr>
<tr>
<td>4.3.1 VME memory Map</td>
<td>28</td>
</tr>
<tr>
<td>4.3.2 PC bus access</td>
<td>28</td>
</tr>
<tr>
<td>4.3.3 Adaptor Control Registers</td>
<td>31</td>
</tr>
<tr>
<td>4.3.4 The Servo Clock</td>
<td>31</td>
</tr>
<tr>
<td>4.3.5 Panic signal</td>
<td>32</td>
</tr>
<tr>
<td>4.3.6 Interrupts</td>
<td>32</td>
</tr>
<tr>
<td>4.3.7 LED indicators</td>
<td>32</td>
</tr>
<tr>
<td>4.3.8 Miscellaneous Notes</td>
<td>32</td>
</tr>
</tbody>
</table>

5 The Mark I motor interface card

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Servo board specification</td>
<td>35</td>
</tr>
<tr>
<td>5.2 Design aims</td>
<td>35</td>
</tr>
<tr>
<td>5.3 Description</td>
<td>36</td>
</tr>
<tr>
<td>5.3.1 Memory map</td>
<td>36</td>
</tr>
<tr>
<td>5.3.2 The latch signal</td>
<td>38</td>
</tr>
<tr>
<td>5.3.3 D/A double buffering</td>
<td>38</td>
</tr>
<tr>
<td>5.3.4 Calibration</td>
<td>39</td>
</tr>
<tr>
<td>5.3.5 Diagnostics</td>
<td>39</td>
</tr>
<tr>
<td>5.3.6 Panic signal</td>
<td>39</td>
</tr>
<tr>
<td>5.4 Board details</td>
<td>40</td>
</tr>
<tr>
<td>5.4.1 Switches</td>
<td>40</td>
</tr>
<tr>
<td>5.4.2 LED indicators</td>
<td>41</td>
</tr>
<tr>
<td>5.4.3 Configuration</td>
<td>41</td>
</tr>
<tr>
<td>5.4.4 Pinouts</td>
<td>41</td>
</tr>
</tbody>
</table>
List of Figures

1.1  Notional controller structure .................................................. 5
2.1  Motor controller block diagram .................................................. 9
2.2  MMCS code example ................................................................. 19
2.3  MMCS code example ................................................................. 20
2.4  MMCS code example ................................................................. 21
2.5  SunOS code example for direct hardware access ............................ 22
4.1  VME Host adaptor memory map (byte addresses shown) ............... 29
4.2  PC bus I/O space address formation ............................................ 30
5.1  Servo board memory map .......................................................... 37
5.2  Motor interface board layout ....................................................... 40
Chapter 1

Introduction

1.1 Overview

This first chapter discusses the motivation for a new modular, and host independent, motor control system for laboratory use. The background to the project and the development of the concept is traced. A number of possible solutions are proposed and discussed, leading to a general description of the system that has been implemented.

Chapter 2 describes in detail an abstract programmer's model of the axis controller. Details of the servo interface hardware are hidden, allowing the applications programmer to concentrate on higher level control. The software implements position, velocity or torque control, selectable per axis, and the closed loop dynamics may be modified by a digital compensation network.

Chapter 3 describes an interactive graphical tool that allows a user to configure the axis controller, perform diagnostics and perform joint level motions.

The last two chapters are not essential reading for casual programmers, but are essential for those programming at the hardware level.

Chapter 4 describes the function performed by the host bus adaptor and also the axis controller bus, which is the same as IBM/PC bus. Details such as redefinition of some signal lines\textsuperscript{1}, and the addressing conventions used are covered. It describes in detail the hardware implementation of the VME host to PC bus adaptor that was built.

Chapter 5 describes in detail the hardware and programming details for the Mark I servo interface board.

\textsuperscript{1}It's not as bad as it sounds
1.2 Background and Motivation

Many projects within the GRASP lab. involve motion control via electric servo motors, for example robots, hands, camera mounts and tables. Each project has been based on a unique hardware/software approach. In the last few years the approaches have included

- VAL-II control language receiving commands over a serial line from a host computer
- RCCL (Hayward and Paul)
- RFMS multiprocessor (Zhang and Paul)

The first approach is limited by communications speed, and is not suitable for real-time sensor based control. The RFMS controller has proved in practice to be very difficult to program, and does not seem to have realized the full potential of its parallel hardware architecture.

RCCL is a very general robot programming environment and is capable of real-time sensor based control, as has been demonstrated by various projects within the lab. It does have the drawback that it is tightly coupled to the VAX architecture and Unimate robots and their controllers.

RCCL provides the programmer with a particular model of the robot and its environment. This model, based on kinematic position equations and cartesian representation using homogeneous transforms, is very powerful, however there are many applications to which it is not well suited. It is at this point that the inherent inflexibility of RCCL becomes a problem, and the application programmer’s effort goes increasingly into outwitting and thwarting RCCL “features”.

Based on discussions with robot users in the laboratory the following points were made

1. Robot control hardware. It was considered that the best platform for robot control would be a powerful single processor system like a workstation. A single thread machine is inherently easier to program, and a workstation provides an integrated environment for program development and high speed execution. To allow a workstation to perform robot control an interface is required to the robot’s electronic subsystems.

2. Robot interface. The RCCL controllers use a relatively high level interface to the Puma robot. The Unimate controller boxes provide position servo capability, A/D\(^2\) and D/A\(^3\) converters etc. A functionally more general interface was designed for the RFMS project, but the interface was physically limited to use within the RFMS(board size, connectors etc).
Professor Paul commissioned a final year project to build a general purpose 6 axis interface for the MicroVAX Qbus, but this was never finished and there is some doubt as to whether MicroVAXs and Qbus are the hardware platform to use in the future.

The author suggested a more general solution, based on the technology developed for the RFMS. The axis controller would be modular, thus allowing it to be expanded easily to cope with changing requirements, for example 7 axis robot, robot + hand, or two cooperating robots. Most importantly the axis controllers would be independent of the host processor bus, whether it be Multibus, VMEbus or Qbus. A simple electronic adaptor would connect the axis controller bus to the host bus, and would represent a relatively small fraction of the total system complexity, thus allowing easy migration to new host computing platforms. It was decided that the axis controller bus should be the IBM-PC bus, due to the variety of compatible products in the marketplace.

3. Robot control software. Based on experience with RCCL and CSIRO's ARCL robot controller[5] it has been decided to redesign the robot control software so as to be very modular, as opposed to the "monolithic" structure of RCCL. The structure looks like comprising a number of simple interfaces and functional blocks, implemented as libraries, and on which the applications programmer can build. The detailed work would be tackled by Gaylord Holder as a Master’s project. A number of considerations in the design are:

- at the lowest level it must be able to interface with the existing RCI interface to Unimate controllers, as well as the new MMCS hardware.
- at the highest level it must provide a similar level of functionality to the RCCL programming environment, since this is one (despite its limitations) with which many workers are familiar. Within this new programming environments different programming tools will hopefully spring up and eventually replace RCCL.

To restate this, a new motion controller should

- be based on a fast single thread processor
- contain a host independent and modular motor interface
- be accessible via a small and modular software library

The remainder of this document is concerned with the first two points only.
1.3 System overview

This section provides an overview of the hardware and software components of MMCS.

1.3.1 Control Processor and Software

The control processor has two main computations to perform

- High rate servo control loops for the motors
- Slower rate trajectory generation

In the existing Unimate controller the servo control loop functions are performed per axis by an 8 bit 6503 microprocessor, see Figure 1.1. To achieve high joint stiffness and dynamic performance a sample rate approaching 1kHz is desired. There is no way that a process running under Unix (on a 1988 vintage workstation) can achieve this order of response, thus the alternatives are to

1. use a separate processor to perform the high speed servo calculations. The software could run on “bare-metal” or under a real-time operating system.

2. perform the servo computation at interrupt level in a Unix device driver.

The first approach offers the most flexibility, and there are a number of possibilities for a separate processor including

- 680x0 VME CPU cards manufactured from many sources. These processor boards can plug into the VME backplane and communicate with the host via shared memory. Having the same instruction set as the host eliminates the need for software cross development tools. Communications and support software could be developed, or an off-the-shelf package such as VxWorks could be utilized.

- Bell Labs JIFFE[3] processor, which can be plugged into a SUN workstation, and has full software support including a C compiler, and host communication facilities.

The second approach is lower in cost but not as flexible. With an attached processor the user can code up an experimental servo algorithm, download then run it. However, a servo loop in the kernel means that the user would have to rewrite the driver, link a new kernel and boot it. Debugging tools exist for the kernel but they are primitive. More seriously kernel code cannot use floating point arithmetic. Such an approach, under the Xenix operating system, has been described[4].

A variation on this theme is the RCI package written by John Lloyd[9] in which the kernel interrupt handler invokes a user process function in kernel
Figure 1.1: Notional controller structure
mode. This provides run time linkage of user code into the kernel, but debugging remains difficult and the user code must obey some strict guidelines.

The approach taken in this project is to build hardware consistent with both approaches, but the first implementation will be use servo loops embedded in a device driver. The driver implements a very general servo loop capable of being configured for position, velocity or torque mode operation. Any application that wishes to can bypass the servo loops and specify motor currents directly (torque mode). In this case the control algorithm is running in a user process and its scheduling cannot be guaranteed, with possibly serious consequences for stable and smooth control. This is unavoidable when working under Unix.

Comparison of the two approaches in Figure 1.1 shows that the functionality of the six 6503 servo cards in the Unimate controller has been shifted to the host computer. Simulations of the servo software for 6-axis computation time was done for a number of processors and the results are summarized below

<table>
<thead>
<tr>
<th>Processor</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sun 3/160</td>
<td>0.43</td>
</tr>
<tr>
<td>MicroVAX 3500</td>
<td>0.25</td>
</tr>
<tr>
<td>MicroVAX II</td>
<td>0.70</td>
</tr>
</tbody>
</table>

This indicates that Unix device driver based servo loops can provide satisfactory sample rates on most of the GRASP laboratory machines. The simulation does not take into account effects such as adaptor hardware access time, interrupt latency, or service overhead time.

1.3.2 Motor interface

The motor interface was designed with the following design aims

- make it as host bus independent as possible
- make it as modular as possible
- use as much of the proven iSBX design[7] as possible
- control up to 16 axes

To achieve this the hardware has been partitioned into three components

1. Motor interface hardware that provides current drive signal to the motor and processes signals from sensors regarding the motor's state.
2. Axis controller bus into which motor interface cards are plugged.
3. Host adaptor to connect motor interfaces to a host computer that will perform the servo computations.
The motor interface is the electronics that connects the motor to the axis controller bus. It provides an analog drive signal to the motor, and measures shaft angle via an incremental encoder, as well as application specific quantities via a general purpose analog input.

Control of a system with up to 16 axes introduces a number of problems such as timing skew between sampling the first and last axis. If the system is connected to a Unix host computer we cannot rely on software, even at driver level to initiate sampling since interrupt latencies can vary by up to 100 μsec. Thus it was seen to be essential that sampling is controlled by a hardware clock, and the host is notified by interrupt so that it can read and process the state information. The hardware clock signal, SCLOCK, is common to all motor interface cards.

Since the host is also interrupted by the SCLOCK, by the time the interrupt handler routine is entered all state information is available to be read. This means that A/D conversion time is overlapped with the interrupt service overhead for maximum efficiency.

Safety considerations indicate that every motor interface should have the ability to indicate its readiness for operation or an error condition. This signal, referred to as PANIC, is also common to all interface cards, anyone of which can assert the line to indicate a system failure. The type of failure can be determined by host software polling all cards.

The axis controller bus needs to be an ordinary computer bus with address, data and control signals, but it also needs to have the SCLOCK and PANIC signals. It was initially decided to define and use a custom bus for this purpose, but later the decision to use the IBM PC bus was made. The PC bus is nearly ideal in that it is simple to interface to, and a wide range of peripheral cards is available for it. The two special signals could have been implemented by a separate ribbon cable linking all boards, but this is not failsafe in that it is possible for cards to be not connected. Instead two signals from the IBM PC bus were “redefined” for these purposes and is discussed further in Chapter 4.

1.4 Acknowledgements

Professor Richard Paul provided the impetus and support for the project.

Dave Feldman did the detailed design of the servo board and design and construction of the VME/PC bus adaptor. Filip Fuma and Mat Donham designed and built the iSBX cards used in the RFMS.
Chapter 2

Application model of the motor controller

A general model for servo motor control is proposed. It is capable of performing most commonly required functions such as closed-loop position or velocity control. If this functionality is not required the application can also directly specify motor current demands. The motor controller referred to here comprises

- The motor interface hardware described in detail in Chapter 5.
- Servo software in the kernel of the host computer

Since future interface boards may incorporate more control functionality, an abstract user model allows the software/hardware balance to be changed without applications being recoded.

Figure 2.1 shows the proposed control model. A number of switches S1...S5 control various operating modes of the controller. Feedback can come from one of two sources (S4)

- An incremental shaft angle encoder
- An A/D converter

A derivative block may be switched into the feedback path (S5) which will cause a velocity servo function to be implemented. The setpoint signal may come from either (S1)

- The application program via a write() system call, which is represented by u in Figure 2.1.
- The A/D converter

S3 switches in an optional Coulomb friction compensation block, while S2 bypasses the feedback control and allows the setpoint to control motor current directly.
2.1 Compensator

A compensator is included in the forward path to allow users to tailor the dynamic response of the closed-loop system. Typically for a DC electric servo motor, the transfer function is

\[ \frac{\Theta}{I} = \frac{k}{s(Js + B)} \]

where \( \theta \) is motor shaft angle, \( i \) is motor current, \( k \) is the motor torque constant, \( J \) is the motor inertia comprising self and reflected load inertia, and \( B \) is viscous friction.

To provide position control, feedback is required, and to achieve good dynamic performance and disturbance rejection some compensation is required.

2.1.1 The general transfer function

The controller implements a unity gain negative feedback loop on position, with a general discrete transfer function compensator as shown in Figure 1. The transfer function is

\[ D(z) = \frac{a_2z^{-2} + a_1z^{-1} + a_0}{b_2z^{-2} + b_1z^{-1} + b_0} \]
where the coefficients $a_i$ and $b_i$ are programmable by the user. The DC gain is given by $\sum a_i / \sum b_i$. All coefficients and quantities are 32 bit signed integers, so care must be given to the scaling of the integer coefficients.

Many design methodologies may be used to synthesize the compensator coefficients\[6\]. If a continuous time transfer function is synthesized, perhaps using any of the standard forms discussed below, techniques such as bilinear transform, Z-transform or pole/zero mapping\[6\] may be used to generate equivalent discrete time transfer functions. Details of some commonly used control strategies are given below. An example of synthesis for a PID control law is given in Section 2.5.

2.1.2 PID implementation

The classical continuous time PID controller has a transfer function of

$$u = Pe + D \frac{d}{dt} e + I \int e dt$$

where $e$ is the error, demanded minus measured plant output. This may be Laplace transformed to

$$\frac{U}{E} = P + Ds + \frac{I}{s}$$

from which it is clear that the transfer function has a pole at the origin, $s = 0$, and a complex pair of zeros affected by the parameters $P$, $I$, and $D$.

2.1.3 PD implementation

The transfer function of a PD controller is

$$\frac{U}{E} = P + Ds$$

which has a zero at $s = -P/D$.

2.1.4 PI implementation

The transfer function of a PI controller is

$$\frac{U}{E} = P + \frac{I}{s}$$

which has a zero at $s = -I/P$, and a pole at the origin.
2.2 Control options

2.2.1 Velocity servo

A velocity servo loop may be implemented by switching in a differentiator (S5) to the position feedback path. The differentiator is implemented by a three point derivative

\[ \frac{dy}{dt} \approx \frac{3y_t - 4y_{t-1} + y_{t-2}}{2} \]

to yield a smoother velocity estimate. Note that the velocity units (see S4) are either encoder counts, or transformed A/D units, per sample interval.

2.2.2 Torque servo

For torque control, the compensation computation is completely switched out (S2), and the user specified value is used directly as motor current demand.

2.2.3 Coulomb friction compensation

Coulomb friction is a non-linear effect, in which an approximately constant torque opposes the motor's torque. The friction torque is not necessarily the same for each direction of rotation, and varies with joint loading, and will thus be somewhat configuration dependant. A optional Coulomb friction feedforward function may be enabled (S3) to compensate for this non-linear effect. The compensator implements the control law

\[ i_m = \begin{cases} 
    i + i_{cp} & \text{if } \dot{\theta} > 0 \\
    i - i_{cn} & \text{if } \dot{\theta} < 0 
\end{cases} \]

where \( i \) is the output of switch S2.

If the velocity is zero, and a non-zero current is specified the sign of the current demand (from the digital compensator) is used, since that indicates the direction of desired motion.

\( i_{cp} \) and \( i_{cn} \) are the currents required to overcome the Coulomb friction torques in the positive and negative rotational directions respectively.

2.2.4 Feedback source

The feedback signal (S4) may come from either the shaft angle incremental encoder \( \theta \), or from the A/D converter associated with the axis. The raw data from the A/D converter is processed with a simple linear law

\[ \phi = aADC + b \]

that provides scaling and offset before it is used as the feedback signal.
For example, opening the feedback path (S5), and selecting demand from the A/D (S1), the servo will implement a programmable digital filter between A/D and D/A. Application software could also log the raw or filtered signal.

2.2.5 Setpoint source

The setpoint, or demand signal may come from one of two sources (S1) as shown in Figure 2.1. Normally it would be supplied by the user's application program to the device driver via a write() system call. However it may be selected to come from the processed A/D signal, $\phi$.

2.3 The Unix device driver

A SunOS device driver (/dev/mc) has been written to implement the application model of the controller, and is described in this section. The mc device driver does not support the many individual features of specific motor interfaces. To access these capabilities it is probably more effective to map to the device hardware from Unix, and directly manipulate control registers as described in section 2.4.

2.3.1 Configuring the servo

Every axis has a parameter structure which describes the mode of operation to the mc device driver.

```c
#include <sys/mcdef.h>

/*
 * Per joint parameter structure
 */
struct mc_param {
    int which;
    int a2, a1, a0, b2, b1, b0; /* compensator coefficients */
    int ic_pos, ic_neg; /* feedforward constants */
    int mode, clkdivisor;
    int ilim, ilimax; /* current limit */
    int plo, phi; /* position limits */
    int adc_a, adc_b; /* adc conversion law */
    int ipole; /* current filter pole */
};
```

The units for `ic_pos`, `ic_neg` and `ilim` are D/A converter units. `plo` and `phi` are in the units of whatever feedback source is selected. `clkdivisor` is the number
of hardware clock ticks between servo computations for this axis. That is, each
axis may be servoed at a sub-multiple of the hardware clock rate.
Possible values for mode are

- **MD.OFF**: Not servoed
- **MD.POS**: Position control mode
- **MD.VEL**: Velocity control mode
- **MD.TORQ**: Torque (current) control mode
- **MD.TEST1**: Generate triangle waveform

Additional values may be or'd with the mode word, such as

- **COULCOMP**: Enable the Coulomb friction feedforward (S3)
- **ADFB**: Feedback comes from $\phi$ not $\theta$ (S4)
- **ZEROFB**: Zero feedback, that is, open-loop operation (S5)
- **ADDMD**: Demand comes from $\phi$ not the computer (S1)
- **ADOFF**: Torque offset comes from $\phi$ not the computer
- **POSCHK**: Check position limits on feedback signal
- **SOFTERR**: Don't shut motors down when error is detected

Note that not all switch combinations are useful, and this is not checked.

The servo parameters for an axis are set or examined using an ioctl() system
call on the mc device. To retrieve parameters from an axis the which element
must first be set to indicate which axis the parameters are required for.

```c
struct mc_param par;
par.which = axis;
ioctl(mcf, HGETPARAH, &par);
```

$mcf$ is the file descriptor for the motor control device, /dev/mc0. To set
parameters the parameter structure should be initialized by the user's program,
and the which element set to indicate which axis the parameters are destined
for.

```c
struct mc_param par;
par.which = 3;
par.mode = MD_VEL;
ioctl(mcf, HSETPARAH, &par);
```

Only one axis may be initialized per ioctl(). All parameters are initialized
as shown in Table 2.1 when the device is opened.

Also at open time the driver scans the axis controller bus looking for motor
interface cards from axis 0 through axis 15. When the device is closed, all motor
currents are set to zero and the PANIC signal asserted.
<table>
<thead>
<tr>
<th>Structure element</th>
<th>Initial value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a0</td>
<td>1</td>
</tr>
<tr>
<td>a1</td>
<td>0</td>
</tr>
<tr>
<td>a2</td>
<td>0</td>
</tr>
<tr>
<td>b0</td>
<td>1</td>
</tr>
<tr>
<td>b1</td>
<td>0</td>
</tr>
<tr>
<td>b2</td>
<td>0</td>
</tr>
<tr>
<td>ic_pos</td>
<td>0</td>
</tr>
<tr>
<td>ic_neg</td>
<td>0</td>
</tr>
<tr>
<td>ilim</td>
<td>1/2 maximum current</td>
</tr>
<tr>
<td>ilimmax</td>
<td>0</td>
</tr>
<tr>
<td>adc_a</td>
<td>1</td>
</tr>
<tr>
<td>adc_b</td>
<td>0</td>
</tr>
<tr>
<td>mode</td>
<td>MD_OFF</td>
</tr>
<tr>
<td>clkdivisor</td>
<td>1</td>
</tr>
<tr>
<td>ipole</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 2.1: Initial parameter values

2.3.2 Choice of parameters

It may appear that there is an overwhelming number of parameters to set before anything can be done. This is true, and unavoidable with the general approach taken. For the Unimate controller many of these issues are handled by the 6503 microprocessors and the code they execute from EPROM. The Unimate servos have been tuned for good performance with the motors and mechanical systems used. For MMCS these parameters must be determined by the user, there is no alternative. The interactive tool mctool can let a user adjust controller parameters to obtain good performance.

The controller parameters are very dependent upon the type of motor, the power amplifier, and mechanical drive train. Parameters that work well for one motor may cause instability with another.

However some practical hints are in order:

- Determine which direction the encoders change when a positive torque is applied to the motor. If the encoders increase in a negative direction then the DC gain of the compensator must be negative, and both Coulomb friction compensation parameters must be negative.

- For position mode control PD control is appropriate, for velocity mode PI control is appropriate.

- The maximum current should be left at some low value (default is half maximum) until the controller parameters and application are well-behaved.
2.3.3 Accessing servo state

The driver maintains state information for each axis.

/*
 * Per joint state and status
 */
struct mc_state {
    int adcval;    /* latest A/D value (processed) */
    int encnow;    /* latest encoder value */
    int inow;      /* last current command issued */
    int error;     /* latest error value */
    int fbnow;     /* latest value of feedback quantity */
    int vel;       /* latest velocity estimate d/dt {fbnow} */
};

A read() system call on the mc device returns a vector of mc_state structures, which may be used by the user as required. The state variables are

adcval The instantaneous value of the A/D after processing via the linear law.

encnow The instantaneous value of the incremental encoder counter.

inow The instantaneous or filtered motor current demand, is related to the torque needed to maintain the position or velocity demand set. It will be related to disturbance forces such as gravity or robot/object interactions. If the parameter ipole is zero inow is the instantaneous current. If non-zero, then the motor current demand is filtered by a unity gain first order digital filter whose pole is at ipole/MC_FSCALE, and the value of inow should be divided by MC_FSCALE to convert from fixed point filter arithmetic to real value.

error The instantaneous error between the feedback quantity and the demand.

fbnow The instantaneous value of the feedback quantity, which will always be the same as either encnow or adcval.

vel The current plant output velocity estimate, in units per sample period.

State information is always available once the mc device is open. It is updated at every SCLOCK, the sample interval is set by the MSETINTERVAL ioctl() call.
MERRPLO Low position limit crossed
MERRPHI High position limit crossed
MERRILIM Sustained current overload
MERRPANIC Hardware panic detected

Table 2.2: Device error codes

2.3.4 Error handling

The MMCS subsystem can generate a number of error conditions. The reason for the error can be found by using the MLASTERR ioctl() call.

```c
int errcode;

ioctl(mifd, MLASTERR, &errcode);
```

The low eight bits of `errcode` are the axis that caused the error, while the high order bits may be one of the codes shown in Table 2.2. The bit MERR_POSERR, if set, indicates that a position error, high or low occurred.

On all error conditions the application process is notified by a signal SIGUSR1. If the operating mode of the axis is OR'd with SOFTERR then no further action is taken by MMCS, and the user's signal handler is responsible for dealing the condition. If SOFTERR is not set the robot shutdown by giving a zero current demand to all motors, and activating the brakes.

The driver always checks for sustained torque overdrive of the motor. If the current demand exceeds the parameter `ilim` for more than `ilimmax` samples the MMCS is shutdown. Motor current is always clipped to the maximum value allowed by the D/A converter. If `ilimmax` is zero, then motor currents are clipped to `ilim` and no error condition is generated.

If `mode` has the POSCHK bit set then the feedback quantity, from encoder or A/D is checked against the limit parameters `phi` and `plo`. The error condition is generated only when the limits are crossed, not continuously while the limit exists.

A hardware panic is initiated by one of the motor interface cards, or the hand held panic button. The failsafe nature of the system design means that panic will also be asserted if connections such as that between host and MMCS, or MMCS and panic button are broken. The axis number bitfield in the error code is meaningless for this condition.

2.3.5 Other device driver functions

Functions, not already discussed, that can be controlled via ioctl() calls are given in Table 2.3.
<table>
<thead>
<tr>
<th>Request</th>
<th>Argument</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MGETNUMCARDS</td>
<td>int</td>
<td>Return the number of motor interface cards present in the axis controller bus.</td>
</tr>
<tr>
<td>MSETNUMJOINTS</td>
<td>int</td>
<td>Specify the number of axes that will be controlled by the servo code. Returns EINVAL if this number is greater than that supplied by the motor interface cards present.</td>
</tr>
<tr>
<td>MGETPARAM</td>
<td>struct mc_param</td>
<td>Return the parameter structure for the axis specified by the which element of the passed parameter (read/write argument). Will return EINVAL if which is greater than the number of axes, or if bo is equal to zero.</td>
</tr>
<tr>
<td>MSETPARAM</td>
<td>struct mc_param</td>
<td>Set the parameter structure for the axis specified by the which element of the structure. Will return EINVAL if which is greater than the number of axes.</td>
</tr>
<tr>
<td>MSTOP</td>
<td></td>
<td>Stop all axes, set all motor torques to zero, joint control modes to MD.OFF, remove enable status, and activate brakes.</td>
</tr>
<tr>
<td>MENABLE</td>
<td></td>
<td>Check that all boards are operating, and allow all D/A’s to be written.</td>
</tr>
<tr>
<td>MSETVERBOSITY</td>
<td>int</td>
<td>If M.VERBOSE bit is set then the driver prints additional diagnostic information during operation. If M.ERRORPRINT bit is set then information is only printed during error situations.</td>
</tr>
<tr>
<td>MGETVERBOSITY</td>
<td>int</td>
<td>Returns the verbosity flag.</td>
</tr>
<tr>
<td>MGETLASTERR</td>
<td>int</td>
<td>Returns the error code for the last error that happened. The lower 8 bits specify the axis, the higher bits specify the error type, see 2.2.</td>
</tr>
<tr>
<td>MSETINTERVAL</td>
<td>unsigned int</td>
<td>Set the hardware timer interval in μsec. Return EINVAL if timer is incapable of meeting the interval requested. If the time interval is greater than the heartbeat timeconstant in the motor interface board PANIC will be asserted by hardware, see 5.3.6.</td>
</tr>
<tr>
<td>MGETINTERVAL</td>
<td>unsigned int</td>
<td>Get the hardware timer interval in μsec.</td>
</tr>
<tr>
<td>MSETENC</td>
<td>unsigned int</td>
<td>Set the hardware encoder register to given value. Lower 8 bit specify axis, next 16 bits specify value.</td>
</tr>
<tr>
<td>MSETLED</td>
<td>int</td>
<td>Control axis indicator LEDs, each bit in the argument controls one LED associated with the axis. Axis is specified by lower 8 bits, LEDs by bits 8...</td>
</tr>
<tr>
<td>Request</td>
<td>Argument</td>
<td>Comments</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>MDIAGMODE</td>
<td>int</td>
<td>Specify the diagnostic operating mode. If bit M_ANDIAG is set then analog loopback is enabled, if bit M_ENDIAG is set then encoder loopback is enabled. EINVAL is returned if the board cannot perform the specified diagnostic.</td>
</tr>
<tr>
<td>MSETDAC</td>
<td>int</td>
<td>The lower 8 bits of the argument specify which axis D/A is set to the value specified by the next higher 12 bits.</td>
</tr>
<tr>
<td>MDACMODE</td>
<td>int</td>
<td>If argument is non-zero then DAC double buffer mode is enabled for all axes, otherwise single buffered mode is enabled.</td>
</tr>
<tr>
<td>MGETSTATS</td>
<td>struct mc_stats</td>
<td>Returns a structure of statistics gathered by the driver about interrupts, such as total, those missed, overrun etc.</td>
</tr>
<tr>
<td>MZEROSTATS</td>
<td></td>
<td>Zero the driver's statistics structure.</td>
</tr>
</tbody>
</table>

Table 2.3: ioctl calls for the mc device

2.3.6 Code example

Figure 2.4 is a code fragment that illustrates the important steps in controlling motors via the mc device driver.

2.4 Accessing hardware directly

This approach to interfacing, mapping device hardware registers to a Unix process, is very specific to the flavor of Unix being used. Some likely approaches are Ultrix via the /dev/bus device driver, or SunOS via the /dev/vme* device drivers. User's following this path should be very familiar with the material in Chapters 4, 5 and the appendices. A SunOS code fragment is given in Figure 2.5.

For more details consult the Unix manual entries for valloc(2) and mmap(2). If an access is made to a address at which no device resides, the VMEbus times out and a SIGSEGV (segmentation violation) signal is delivered to the user process. A SIGBUS (bus error) signal can be delivered if the device hardware messes up the VME cycle.

Note that accesses to devices via mapped memory cause “non-priviliged” address modifiers to be issued while accesses from a device driver cause “priviliged” address modifiers[1].
/*
Example program showing use of MMCS system for 2 axis control in
velocity mode with random trajectories.
*
* pic 1/89
*/

#include <stdio.h>
#include <sys/file.h>
#include "/usr/sys/sundev/mcdei.h"
#include <signal.h>

int fd,
naxis,
verbose,
intval = 2;
int setp[16];
char *pname;

#define VAL(b) (atoi(&b[1]))

main(ac, av)
int ac;
char **av;
{
    struct mc_stats stats;
    struct mc_state state[16];
    struct mc_param param;
    int i;
    void mmcserr();

    pname = av[0];
    /*
    if (ac == 1) {
        usage:  fprintf(stderr, "Usage: %s %\n", av[0]);
        exit(1);
    }*/
    while (--ac > 0 && **++av == '"') {
        register char *p = *av;

        while (++p != '\0')
            switch (*p) {
                case 'v': verbose++; break;
                case 't': intval = VAL(p); break;
            }
    }

Figure 2.2: MMCS code example
if (verbose) {
  i = M_VERBOSE;
  ioctl(fd, MSETVERBOSITY, &i);
}

if (ioctl(fd, MGETNUMCARDS, &i) < 0)
  perror("ioctl:");
printf("%d cards in system\n", i);

naxis = i*2;
if (ioctl(fd, MSETNUMJOINTS, &naxis) < 0)
  perror("ioctl:");

intval *= 1000;
if (ioctl(fd, MSETINTERVAL, &intval) < 0)
  perror("ioctl:");
  
/*
 * zero the encoders
 */
for (i=0; i<naxis; i++) {
  setp[i] = 0;
  ioctl(fd, MSETENC, &i);

/*
 * initialize the parameters
 */
param.which = 0;
ioctl(fd, MGETPARAM, &param);
param.a0 = 50;
param.b0 = -1;
param.mode = MD_VEL | SOFTERR | POSCHK;
param.plo = -6000;
param.phi = 6000;
param.ilimmax = 0;
}

Figure 2.3: MMGS code example
for (i=0; i<naxis; i++) {
    param.which = i;
    ioctl(fd, MSETPARAM, &param);
}

if (ioctl(fd, MENABLE) < 0) {
    fprintf(stderr, "cant enable\n");
    exit(3);
}

for (i=0; i<naxis; i++)
    setpt[i] = veloc();
write(fd, setp, naxis * sizeof(int));

/*
 * The program now waits, all control is done in a signal
 * handler invoked when an axis exceeds its position limits.
 */
for (; ;)
    sigpause();

void
mmscser(int)
{
    int      errcode;
    int      axis, code;

    ioctl(fd, MGETLASTERR, &errcode);
    axis = errcode & 0xff;
    code = errcode & "0xff;
    printf("%s limit on axis %d\n", (code == MERR_PLO) ? "low" : "high", axis
    );
    /*
     * choose random velocity in opposite direction for error axis
     */
    setp[axis] = -veloc() * abs(setp[axis]) / setp[axis];
write(fd, setp, naxis * sizeof(int));

veloc()
{
    return random() % 20 + 10;
}

Figure 2.4: MMCS code example
```c
#include <signal.h>
#include <sys/file.h>
#include <sys/mman.h>
#include <sys/types.h>

int fd, /* file descriptor for the bus device */
    len, /* length of memory window to map */
    off, /* base of memory window */
    buserr();
caddr_t caddr_t

if (len < getpagesize()) /* round len up to a page size */
    len = getpagesize();

fd = open("/dev/vme16", O_RDWR); /* open the bus device */
if (fd < 0)
    perror("open");

addr = valloc(len); /* allocate virtual memory */
if (addr == NULL)
    perror("valloc");

/*
 * map bus memory window into user's virtual memory
 */
if (mmap(addr, len, PROT_READ|PROT_WRITE, MAP_SHARED, fd, off) < 0)
    perror("mmap");

signal(SIGBUS, buserr); /* set up signal handlers */
signal(SIGSEGV, buserr);

buserr()
{
    printf("BUS ERROR\n");
    longjmp(env, 1);
}
```

Figure 2.5: SunOS code example for direct hardware access
2.5 Control synthesis

There are a number of methods of transforming a continuous time system to a discrete time system, such as

- Pole/zero mapping
- Bilinear transformation
- Bilinear transformation with frequency prewarping
- Z-transform
- Z-transform with zero-order hold

Each of these techniques has a different effect on characteristics of the system such as DC gain, frequency response, overshoot etc. Standard control systems texts[8][9] can provide more details.

One simple approach is the bilinear transform

\[ s = \frac{2}{T} \frac{z - 1}{z + 1} \]

where \( T \) is the sampling interval. The Laplace transform expression for a general PID controller is

\[ \frac{u}{e} = \frac{Ps + Ds^2 + I}{s} \]

where \( P \), \( D \) and \( I \) are the proportional, derivative and integral gains respectively. Substituting yields

\[ z^{-2}(T/2 + 2D/T - P) + z^{-1}(I - 4D/T) + (P + 2D/T + IT/2) \]

\[ -z^{-2} + 1 \]

which is in the form implemented by the controller's compensator. The coefficients should be scaled, so that all are greater than 1, since all arithmetic is performed in 32 bit fixed point.
Chapter 3

mcTool

mcTool is an interactive graphical tool that runs under the SunView window environment. The Sunview Programmer's Guide provides some details on the conventions of this interface.

When invoked mc tool opens the motor control device, /dev/mc0, and determines the number of motor interface cards. A panel is then built for every axis present. The topmost panel provides global controls for the device, in particular sample interval. The enable button must be pressed to commence servo operation once parameters for the various axes have been set. The diagmode switch controls analog and encoder loop back modes of the motor interface board, see Chapter 5.

mc tool reads status information from the device driver five times per second and updates the displayed values for encoder count, A/D and motor current.

Many parameters of the device driver may be altered, by typing in new numeric values or adjusting control sliders. Operating modes such as Coulomb friction compensation, LED indicator, feedback source, or servo mode may be controlled by switches.

The SetUpSource switch allows the driver setpoint to come from either

- the SetUppoint slider
- the A/D converter

- a square wave whose amplitude is controlled by the Amplitude slider which replaces the SetUppoint slider in this mode.

The Mode switch controls the actual servo operating mode. Its values can be any of

- Off this axis not servoed in which case the SetUppoint slider is not
- Pos this axis is in position servo mode and the labels of the SetUppoint slider reflect this. displayed.
Vel this axis is in velocity servo mode and the labels of the Setpoint slider reflect this.

Torq this axis is in torq servo mode and the labels of the Setpoint slider reflect this.

Test1 this axis is in MD_TEST1 mode which outputs a sawtooth waveform of amplitude equal to the present current limit, Ilim.

Diag this axis is not servoed by any adjustment of the Setpoint slider causes that value to be output to the D/A. This is useful for testing the D/A's or setting a specific output voltage.

The PID gain sliders have units of %, that is the displayed value divided by 100 since the sliders can only have integer values. As the sliders are adjusted the appropriate compensator coefficients are computed and modified in the driver.

Some command line switches for mc_tool are

-tinterval Set the initial value of the interval time to the specified number of milliseconds.

-u Don't update the encoder, A/D and motor current state information.

-z Zero all incremental encoder registers.

Since many processes can open the mc device at the same time, mc_tool can be used as a monitor of the motor state while other processes are controlling it. In this situation mc_tool should leave all axes in the
Chapter 4

Host adaptor

4.1 Introduction

A fundamental component of the proposed modular motor control system (MMCS) is the interface between the host bus and the axis controller bus. The first such interface contracted is for VME bus host machines. This chapter describes the hardware details necessary for application program development.

4.2 Adaptors in general

Physically an adaptor consists of two cards linked by a ribbon cable. One board plugs into the host bus, and the other plugs into the axis controller bus.

For future host bus adaptors it is likely that the existing axis controller side of the adaptor could be used, necessitating only a new host bus card.

4.2.1 Generic specification for adaptors

The adaptor provides five main functions

1. A mapping of memory accesses on the host bus to PC bus I/O cycles on the axis controller bus.

2. A programmable source of clock pulses, SCLOCK, used to synchronize the latching of state measurements in all motor interface cards.

3. A facility to interrupt the host processor on two conditions, clock pulse SCLOCK and control bus detected panic condition PANIC.

4. A general purpose clock signal (around 8MHz) must be supplied to the PC bus for use by motor interface boards.

27
5. Safety features such as interfacing to the handheld panic button, and control of mechanism brakes if present.

4.2.2 PCbus signals redefined

Two PC bus signals have been redefined for this application. From a purist's standpoint this is bad design, but is unavoidable unless a separate cable was run linking all motor interface cards, or we adopted a different bus altogether. Some care was taken in choosing which lines to use for these special purposes.

1. SCLOCK signal, used to control latching on all motor interface cards. This signal uses one of the PC bus DMA request lines DRQ1...DRQ3. It is jumper selectable on the adaptor and motor interface cards, and should be set such that all cards use the same line, and that the line does not conflict with other PC bus devices.

2. PANIC signal, used to indicate that one of the motor interface cards has detected an error condition. This line is a wired or, that is, any board can pull it high to assert the PANIC condition, which is then available to all other boards.

4.3 In particular: VMEbus adaptor

Physically the adaptor is two cards linked by an 8 foot, 50way ribbon cable. One board plugs into the VME host's backplane, and the other plugs into the control (PC) bus. The VME end is based upon a Logical Design Group VME-5100D VMEbus prototype card with only 10 extra chips. The PC end is based on a bare wirewrap prototype module.

4.3.1 VME memory Map

The adaptor occupies 2kbytes of VME A16 address space, layed out as shown in Figure 4.1. It comprises two regions, one that is mapped to PC bus requests, and another that is adaptor control registers. The current configurable base address of this segment is 0x6000.

The adaptor only responds to 8(O) data, that is byte data at odd addresses. Short (16 bit) requests to even addresses (odd address-1) will transfer the bottom byte of the short to the adaptor, the top byte is ignored on write, and 0xff on read.

4.3.2 PC bus access

When A10 is 0, all accesses to the adaptor are passed straight through to the PC slave bus. These accesses become I/O space requests on the PC bus and occupy the odd addresses 0x001 through 0x3FF on the VME bus.
Figure 4.1: VME Host adaptor memory map (byte addresses shown)
Figure 4.2: PC bus I/O space address formation
The PC bus allows for 16 bit I/O space addresses. IBM however define only unique values or ranges for the least significant 9 bits of the I/O address. The address range for prototype card, which we use for motor interface cards, is 0x300 to 0x31f. However this is not enough address bits to support upto 16 interface cards, each with upto 32 bytes of registers. We thus use address bits 10...13 to perform the board select function, and bits 0 through 4 for register select.

The address mapping between VME and PC I/O space is shown diagramatically in Figure 4.2. VME address bits A01...A05 are mapped to PC address bits A00...A04. VME address bits A06...A09 are mapped to PC address bits A10...A13. PC address bits A05...A09 are set via the control register. PC address bits A14 and A15 are fixed at 0.

For the motor interface cards, VME address bits A01...A05 will specify a device register byte on an interface card. VME address bits A06...A09 specify which controller card is being selected on the PC bus. Thus, a bank of up to sixteen controller cards is mapped as a contiguous space of addresses on the VME bus.

### 4.3.3 Adaptor Control Registers

When VME address bit A10 is 1, an adaptor register is selected. These registers include the interrupt vector register, the PC control/status register (PCCSR), and the 8254 Programmable Interval Timer (PIT).

When address bit A06 is a 1 on the VME bus, the vector register is selected. This puts the vector register address at 0x0041. A single byte may be written to this register to supply a vector number used during interrupt service on the VME bus. This register may not be read.

When address bit A06 is 0, one of the other registers is selected. When VME addresses A05...A03 are (respectively in binary) 000, the PIT is selected. In this case, A02 and A01 specify the register on the PIT. Thus, the PIT registers are: timer 0, 0x401; timer 1, 0x403; timer 2, 0x405; control register, 0x407.

When A06 is low and A05...A03 are 001 (binary), the PCCSR is selected. Thus, its address is 0x409. The bitfields of this register are shown in Table 4.3.3.

### 4.3.4 The Servo Clock

The servo clock, SCLOCK, is the output of timer 1 on the 8254. It is clocked by timer 0, which is in turn clocked by the 8 MHz system clock which is derived from the VME bus. The system clock may be configured down to 4, 2, or 1 MHz, and is also passed to the PC bus. Timers 0...2 are permanently gated on.

Since timers 1 and 2 are cascaded, allowing a count of 32 bits to be made at 8 MHz, a very wide range of servo loop times is achievable. SCLOCK clocks an edge triggered latch which can cause a VME bus interrupt request. The servo
clock is also passed to the PC bus on the DRQ 3 line, for use by the motor interface cards.

The SCLOCK signal is also fed to timer 2 which may be used to by driver software to check that interrupts are not being missed, that is at each interrupt timer 2 is only 1 different from its value at the last interrupt.

4.3.5 Panic signal

The panic signal from the PC bus is logically combined with the status of the handheld emergency stop button, and used to generate the system PANIC signal. The rising edge of this signal clocks an edge triggered latch which can cause a VME bus interrupt request.

4.3.6 Interrupts

Interrupts are controlled by the PC control register (PCCSR). When bit 7 of the PCCSR is low, the two interrupt latches are forced to clear. They will remain cleared until bit 7 is set high. In this state, a low-to-high transition of SCLOCK will set the servo clock interrupt latch, and a low-to-high transition on PANIC sets the panic interrupt latch. They will remain high until cleared by clearing bit 7. Both latches are cleared together, so an interrupt service routine must poll both interrupts and service the appropriate ones before clearing bit 7.

The status of the latches is read back from bits 5 and 6. To enable the latches to interrupt the VME bus, a 1 must be written into bit 5 or 6. If interrupts are used, don’t forget to set the vector latch, see section 4.3.3.

Currently the VME-5100D is set to interrupt at VME level 5, which is at the same level as the clock and Unix scheduler for a Sun3[2]. To alter the level both jumpers J5 and J7 on the VME-5100D must be altered. Be sure to remove the interrupt acknowledge (IACK) jumper from the backplane for the slot containing the VME-5100D. A “spurious level 7 interrupt” message on the Sun console indicates that the IACK daisy chain is incorrectly jumpered or that J5 and J7 on the VME-5100D are inconsistent.

4.3.7 LED indicators

The adaptor card has two LED indicators. The green LED is the SCLOCK signal, while the red LED is the status of the user interrupt request to the VME-5100D interrupter logic.

4.3.8 Miscellaneous Notes

VME bus resets is passed to the PC bus. The reset signal also resets the PCCSR to 0x00 (interrupts off).

32
The PC side of the adaptor may be powered down while the system is running without affecting the VME side. To disconnect the VME side from a SUN 3, the SUN must be shut down, powered off, and the VME-5100D board and 3U-2U adapter must be removed from the SUN backplane, and the IACK jumper installed.
<table>
<thead>
<tr>
<th>Bit</th>
<th>Write operation</th>
<th>Read operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-0</td>
<td>the PC bus address bits A05 through A09</td>
<td>what was written to those bits</td>
</tr>
<tr>
<td>5</td>
<td>when set, enables interrupts from SCLOCK</td>
<td>Bit 5 is 1 if an SCLOCK interrupt has been latched (but not necessarily passed to the VME bus)</td>
</tr>
<tr>
<td>6</td>
<td>when set, enables panic interrupts from the PC bus</td>
<td>Bit 6 is 1 if a panic interrupt has been latched (also, not necessarily passed to the VME bus)</td>
</tr>
<tr>
<td>7</td>
<td>controls the clearing of the two interrupt latches. When it is 0, the latches get cleared. When it is 1, the latches may get set</td>
<td>returns what was written to it.</td>
</tr>
</tbody>
</table>

Table 4.1: PCCSR bitfields
Chapter 5

The Mark I motor interface card

The motor interface is the electronics that connects the motor to the axis controller bus. It provides an analog drive signal to the motor, and measures shaft angle via an incremental encoders, as well as application specific quantities via a general purpose analog input.

This chapter provides hardware specifications and details to those needing to program the device hardware directly.

5.1 Servo board specification

5.2 Design aims

One of the most important design features is to allow for the non-deterministic timing of the host computer, targeted in the first instance to be a SUN workstation running Unix. The use of an additional CPU running a real-time operating system would provide better performance, albeit at greater cost.

At the short sample times required (less than 10ms) a Unix process cannot be reliably scheduled to respond. However a device driver working at high hardware priority will respond within 100us, but will not be deterministic due to interrupts being locked out in critical regions of the Unix kernel. Thus, it was not desirable for the data sampling to be controlled directly by the host computer.

Instead, sampling is controlled by an axis controller bus signal SCLOCK, and at each sample all A/D converters and encoder registers are latched, and the host notified by interrupt that new data has arrived. The host device driver can then read the robot state and compute new setpoints to be written to the
D/A converters. State is thus sampled at a fixed intervals with zero timing error. Additionally the D/A converter updates can occur in a double buffered mode in which the new values are not output until the next sample time.

5.3 Description

Each board provides all the functionality required to control two servo axes, referred to here as joint 1 and joint 2. Each controller consists of an A/D, a D/A, and an incremental encoder interface (IEI).

- AD574 12 bit analog to digital converter. This may be used to input application specific signals such as joint angle, torque, current etc.
- AD667 12 bit digital to analog converter for specifying motor current or torque. The device has the ability to double buffer the digital commands.
- HCT2000 incremental encoder interface (IEI) to decode quadrature signals from incremental encoders. This device has 16 bit resolution, and can be read or written at anytime. It also has the facility for the count to be latched by an external signal, which we use for calibration purposes.
- i8255 to provide control of all board operating modes and to provide status information. The 8255 device provides 24 programmable input or output lines. The usage in this interface is summarized in Table 5.1.

The D/As contain two cascaded 12 bit latches. The D/A must be accessed twice to load a new value into the first latch, once for the low byte, and once for the high nibble. D/A data is 12 bit offset binary. The loading of the second latch, and hence the D/A output, is controlled by port B bit 3 of the 8255 device to occur either when the high nibble is written, or on the edge of the SCLOCK signal.

The A/Ds convert all 12 bits on the edge of the SCLOCK. The 12 bit offset binary values must be read in two byte accesses. The status of the conversion can be read via port C bit 4 of the 8255, which when set, indicates both converters are done.

Data sheets on these devices are provided in an Appendix.

5.3.1 Memory map

The memory map of the motor interface card given in Figure 5.1 shows the addresses as seen from the PC bus. All of the addresses given are relative to the board's base address, and are five bits only.

The board resides at address 0x300 to 0x31F in 9 bit (IBM standard) PC I/O space. However this is not enough address bits to support upto 16 interface cards, each with upto 32 bytes of registers. We thus use address bits 10...13 to
Figure 5.1: Servo board memory map
peform the board select function, and bits 0 through 3 for register select. Thus, 0x0300 is the base of board 0, and 0x3F00 is the base of board fifteen. The board address is set by a 4 bit DIP switch on the card, when a switch is on, or closed, it corresponds to a 0 bit.

The 8255 programmable peripheral interface (PPI) is at addresses 0x00 through 0x03.

The HCT2000 for joint 1 is at addresses 0x04 through 0x05, while that for joint 2 is at addresses 0x06 through address 0x07. The most significant byte occupies the lower address. The HCT2000 may be written to initialize the 16 bit up/down counter. It may be read at anytime to give the instantaneous counter value, however the first read after a latch command will yield the counter value at the time of the latch. The operating mode of the HCT2000 is set via port A of the 8255, and would normally be mode 5, although a number of other modes such as frequency and interval measurement may be useful. The HCT2000’s two byte registers should always be accessed in the same order after reset, see the applications note for more details.

The D/A’s are at addresses 0x08 through 0x0B. These locations are write-only. Location 0x08 is the low byte of D/A 1, and 0x09 is the high nibble of D/A 1. Similarly, 0x0A and 0x0B write to D/A 2.

The A/Ds are at addresses 0x0C through 0x0F. These locations are read-only. Location 0x0C returns the low byte of A/D 1. Location 0x0D returns the low byte of A/D 2. Location 0x0E returns the high nibbles of both A/Ds. The high nibble of A/D 1 will reside on bits 0 through 3 of this byte, and A/D 2 will use bits 4 through 7.

5.3.2 The latch signal

The latch signal is used to start the conversion of the A/Ds, enable the output latch on the D/As, and cause the IEs to latch their counts.

This latch signal may come from either the rising edge of SCLOCK (a backplane signal from the host adaptor) or be generated via addressing (see Figure 5.1). To enable latch on the rising edge of SCLOCK port B bit 2 of the 8255 must be asserted.

Any access to location 0x10 will cause a latch signal to the A/Ds and IEs, just as the SCLOCK signal does. Any access to 0x12 will do the same, except that it will cause all boards in the controller to simultaneously latch. Jumper E021...E023 (see Section 5.4.3) control whether or not the D/A responds to the software latch signals.

5.3.3 D/A double buffering

The D/A converters are capable of working in normal or double buffered mode. In normal mode, the D/A analog output reflects exactly what is written to it. In double buffered mode, the analog output changes to the last written value
<table>
<thead>
<tr>
<th>Bit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-2</td>
<td>Control the operating mode for both IEIs (HCT2000 chips).</td>
</tr>
<tr>
<td>3</td>
<td>Diagnostic LED 1.</td>
</tr>
<tr>
<td>4</td>
<td>Diagnostic LED 2.</td>
</tr>
<tr>
<td>5</td>
<td>Artificial encoder A signal for IEIs.</td>
</tr>
<tr>
<td>6</td>
<td>Artificial encoder B signal for IEIs.</td>
</tr>
<tr>
<td>7</td>
<td>Artificial encoder I signal for IEIs.</td>
</tr>
</tbody>
</table>

Port B: (OUTPUT)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>When 0, the artificial encoder signals are passed to the HCT2000s. When 1, encoder signals from the joints are passed through.</td>
</tr>
<tr>
<td>1</td>
<td>When 0, the output of the D/As is looped back to the A/Ds. When 1, the output of the D/As is switched to the joints.</td>
</tr>
<tr>
<td>2</td>
<td>This bit must be a 1 for the external servo clock on the PC bus to affect this board.</td>
</tr>
<tr>
<td>3</td>
<td>This bit controls the output mode of the D/As. When 0, writing to the high address of a D/A causes its output to be updated. When 1, the internal servo clock, conditioned by Bit 2, causes both D/A outputs to be updated.</td>
</tr>
<tr>
<td>4</td>
<td>When 0, the index latches (see Port C, bits 1 and 3) are cleared. When 1, the index latches get set on the next index pulse. Also, when this bit is a 1, index pulses cause the IEIs to latch.</td>
</tr>
<tr>
<td>5-7</td>
<td>Unassigned</td>
</tr>
</tbody>
</table>

Port C: (INPUT)

<table>
<thead>
<tr>
<th>Bit</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This returns the status of the heartbeat for joint 1. A 1 indicates that all is well. The heartbeat gets retriggered when the low byte of the D/A gets written to.</td>
</tr>
<tr>
<td>1</td>
<td>This is a 1 if the index for joint 1 has been latched. The index gets latched when the index latch enable is set to 1 (Port B, bit 4) and a high level is seen on the index input.</td>
</tr>
<tr>
<td>2</td>
<td>The same as Bit 0, but for joint 2.</td>
</tr>
<tr>
<td>3</td>
<td>The same as Bit 1, but for joint 2.</td>
</tr>
<tr>
<td>4</td>
<td>This signal is high when the A/Ds are not converting. A low to high transition on this signal means that good data is in the data latches for the A/Ds.</td>
</tr>
</tbody>
</table>

Table 5.1: PPI bits
synchronously with the latch signal. Thus guarantees a fixed one sample time delay between reading state and providing a new setpoint value.

5.3.4 Calibration

Calibration of incremental encoders is important for control of many manipulators. Frequently the procedure involves driving the axis so that the encoder’s zero index is detected, and measuring the shaft angle with some low resolution absolute transducer (such as a potentiometer), and then computing the correct value for the encoder register. In RFMS the axes were driven, the zero index status polled, and the motor stopped when the index is detected. It was found necessary to drive the motors very slowly else the index would be missed.

The new interface provides a calibration mode, in which the motor can be driven at any speed, and when the index pulse is detected, the instantaneous encoder value is latched and the host notified. To enable this mode port B bit 4 of the 8255 should be asserted. Port C bits 1 and 3 indicate that the IEIs for joints 1 and 2 respectively have been latched. The IEIs can then be read to determine the encoder count at which the index pulse occurred, another encoder read to obtain the current encoder counts, and determination of motor potentiometer voltages provides all the information needed to determine the absolute motor angle.

Note that subsequent index pulses are not locked out, and will also latch the encoder count. Detection of an index pulse does not stop the motor, this must still be done by host software upon detection of the index latched status.

5.3.5 Diagnostics

The board has a number of diagnostic test facilities built in. Firstly the input of the A/D converters can be “looped back” to the D/A outputs, which allows testing of the A/D, D/A devices and the analog signal paths. The loopback is done via a relay, so some short time should be allowed for the relay contacts to close. This mode is enabled by setting 8255 port B bit 1.

Secondly, synthetic encoder signals A, B and I can be fed into the IEIs to test their operation. The synthetic signals are the same for both axes, and come from 8255 port A bits 5...7. This mode is enabled by clearing 8255 port B bit 0.

5.3.6 Panic signal

One signal on the backplane is the active high PANIC line, that may be asserted by any board that detects a failure. The only failure detected by this interface card is a failure to regularly update the D/A converters with new setpoint values. The D/A write signal drives a oneshot which has a 50ms timeconstant. The outputs of the oneshot (one per axis) are called heartbeats (since they indicates
something is going on), and can be checked by onboard LED indicators and by software via port C bits 0 and 2 of the 8255.

Note that the "panic" condition will not go away until all the D/A converters have been written to initially. Note also that for SCLOCK intervals longer than the heartbeat timeconstant PANIC will be asserted regularly, it was not considered that servoing at less than 20Hz would be useful.

5.4 Board details

Details of the motor interface board layout are given in Figure 5.2.

5.4.1 Switches

There are two switches on the card, one per axis. Their purpose is to take an axis "out" of the controller without having to remove a card, or half a card. In the enable position everything works as described. When disabled, the analog current drive for the axis is switched from the D/A output to ground, and the heartbeat signal is ignored, thus panic won't be caused when that axis's D/A is not updated.
5.4.2 LED indicators

<table>
<thead>
<tr>
<th>LED</th>
<th>Color</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>encB1</td>
<td>Green</td>
<td>Encoder B signal for joint 1.</td>
</tr>
<tr>
<td>encI1</td>
<td>Green</td>
<td>Encoder index signal for joint 1.</td>
</tr>
<tr>
<td>encA2</td>
<td>Green</td>
<td>Encoder A signal for joint 1.</td>
</tr>
<tr>
<td>encI2</td>
<td>Green</td>
<td>Encoder index signal for joint 1.</td>
</tr>
<tr>
<td>h/beat1</td>
<td>Green</td>
<td>Heartbeat signal for joint 1.</td>
</tr>
<tr>
<td>h/beat2</td>
<td>Green</td>
<td>Heartbeat signal for joint 2.</td>
</tr>
<tr>
<td>gp1</td>
<td>Red</td>
<td>General purpose (software controllable) indicator for joint 1.</td>
</tr>
<tr>
<td>gp2</td>
<td>Red</td>
<td>General purpose (software controllable) indicator for joint 2.</td>
</tr>
<tr>
<td>power</td>
<td>Red</td>
<td>Board 5V supply is OK.</td>
</tr>
</tbody>
</table>

Table 5.2: Motor interface LED indicators

5.4.3 Configuration

There are four groups of jumpers, as described in by Table ??

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>E001</td>
<td>E002</td>
<td>A/D1 10V range</td>
</tr>
<tr>
<td>E001</td>
<td>E003</td>
<td>A/D1 20V range</td>
</tr>
<tr>
<td>E004</td>
<td>E005</td>
<td>A/D2 10V range</td>
</tr>
<tr>
<td>E004</td>
<td>E006</td>
<td>A/D2 20V range</td>
</tr>
<tr>
<td>E007</td>
<td>E008</td>
<td>If closed D/A1 10V span</td>
</tr>
<tr>
<td>E009</td>
<td>E010</td>
<td>If closed D/A2 10V span</td>
</tr>
<tr>
<td>E023</td>
<td>E021</td>
<td>D/A does not respond to software latch</td>
</tr>
<tr>
<td>E022</td>
<td></td>
<td>D/A responds to software latch</td>
</tr>
<tr>
<td>E020</td>
<td>E017</td>
<td>Sample clock is DRQ1</td>
</tr>
<tr>
<td>E020</td>
<td>E018</td>
<td>Sample clock is DRQ2</td>
</tr>
<tr>
<td>E020</td>
<td>E0189</td>
<td>Sample clock is DRQ3</td>
</tr>
<tr>
<td>E011</td>
<td>E012</td>
<td>Panic is IRQ5</td>
</tr>
<tr>
<td>E011</td>
<td>E013</td>
<td>Panic is IRQ4</td>
</tr>
<tr>
<td>E011</td>
<td>E014</td>
<td>Panic is IRQ5</td>
</tr>
<tr>
<td>E011</td>
<td>E015</td>
<td>Panic is IRQ6</td>
</tr>
<tr>
<td>E011</td>
<td>E016</td>
<td>Panic is IRQ7</td>
</tr>
</tbody>
</table>

Table 5.3: Mark 1 configuration jumpers

They are used to set voltage scaling for the A/D and D/A converters as well as to select which PC bus lines are used for the SCLOCK and PANIC signals.
5.4.4 Pinouts

For each motor there is a 16 way ribbon cable connector on the card. This connector is compatible with the lines 1...16 on the 26 way connectors and cables used in the RFMS, as long as pin 1 of MMCS cables mates with pin 1 of RFMS cables. Note that all even numbered lines are ground.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Encoder A signal</td>
</tr>
<tr>
<td>3</td>
<td>Encoder B signal</td>
</tr>
<tr>
<td>5</td>
<td>Encoder index signal</td>
</tr>
<tr>
<td>7</td>
<td>Analog input</td>
</tr>
<tr>
<td>9</td>
<td>D/A analog output</td>
</tr>
</tbody>
</table>

Table 5.4: MMCS motor cable pinouts
Bibliography


A.3 Simulator for the Systolic Array Processor
Symbolic Simulator/Debugger
for the
Systolic/Cellular Array Processor

Janez Funda

University of Pennsylvania
Department of Computer and Information Science
Philadelphia, PA 19104

October 15, 1988
1 Introduction

This document describes an implementation of a symbolic simulator/debugger for the Systolic/Cellular Array Processor (SCAP), which is currently being built at Hughes Research Laboratories. The SCAP system is a parallel computer with 256 identical processing elements (PEs) connected using a mesh interconnection network in a $16 \times 16$ grid. Each PE features a two-bus internal architecture, with seven functional units, and four I/O ports used to communicate with its four neighboring PEs. All functional units operate on 32-bit fixed point data. The reader is referred to [Przytula,88] for a detailed description of SCAP's architecture, data representation format, and machine level operation of the system.

The construction of a symbolic simulator/debugger for the SCAP system was motivated by the need to run, test and debug some assembly language programs that were being developed in the General Robotics and Active Sensory Perception (GRASP) Laboratory here at the University of Pennsylvania. The Laboratory is expecting a delivery of the actual computer some time this year. However, we felt that availability of a simulator/debugger prior to the actual delivery of the system would increase the utilization of the hardware, once installed.

The scope of the simulator/debugger project included (1) an implementation of a simulator, that corresponds as closely as possible to the operation of the actual hardware, (2) a loader module to load user data into memory prior to program execution, and (3) an easy-to-use basic user interface, allowing the user to trace the execution of the program, single-step through the code, examine all relevant portions of the system, and make references to the symbolic information in the source program (e.g., symbolic labels, processor mask names, etc). The current implementation meets all aspects of the above objectives. It does not, however, strive for a sophisticated visual interface with the user. Therefore, no graphics front end interaction has been implemented. This would be a useful extension to the system, should the simulator/debugger be used more than currently anticipated.

The SCAP simulator/debugger (hereforth referred to as sim) described in this document is designed to operate in conjunction with the corresponding SCAP assembler (ass), developed by Hartholtz [Hartholtz,88]. The reader is encouraged to refer to [Hartholtz,88] for a description of the SCAP assembly language syntax, facilities, and
Simulating the execution of a given SCAP assembly language program is therefore a two stage process. The program is first assembled using `ass`, producing a file containing non-relocatable absolute object code and the symbol table information [Hartholz,88]. This file plus a file containing the data to be operated on are then passed as arguments to `sim`. The details of the data file format as well as user interaction with the system are described below.

This document is intended as a brief user's guide for `sim`. To illustrate some of `sim`'s features, I have added a couple of appendices at the end of this document, showing an actual interaction with the system. Appendix A shows an assembly language source program and the corresponding data file, while Appendix B contains the transcript of a `sim` session, performing a simulation of the program on the given data. Note that the program has been written to exhibit a wide variety of instructions for testing and demonstrational purposes, and is *not* intended to compute any particular useful function.

## 2 Description of the System

### 2.1 General

**NAME**

`sim` — SCAP symbolic simulator/debugger

**SYNOPSIS**

```
  sim [ -w ] codefile datafile
```

**DESCRIPTION**

`sim` is a symbolic simulator/debugger for the Systolic/Cellular Array Processor currently being designed and built at Hughes Research Laboratories.

`codefile` is a binary file containing non-relocatable object code, produced by running `ass` on a SCAP assembly language source program. `codefile` also contains symbol information about the original source program.

`datafile` is an ascii text file containing the data to be manipulated by the program stored in `codefile`. See Section 2.3 for a detailed description of the `datafile` format.
The following option is recognized by sim:

-w Enable the reporting of run-time warnings. Most of the warnings, which are suppressed by default, are concerned with alerting the user to the potentially undesirable side effects of inter-processor I/O (see [Przytula,88]).

sim operates by reading codefile and datafile and initializing its internal data structures corresponding to the Instruction Memory (also referred to as Program Memory) and Data Memory accordingly. Information about the symbols defined in the original source program is also obtained from codefile and appropriate internal symbol tables are constructed. sim then enters a prompt-execute loop where the user is repeatedly prompted for commands, which are in turn parsed, interpreted, and executed by the simulator. Various self-explanatory error and warning diagnostics at both the command and execution levels are issued when appropriate.

2.2 User interface in sim

Upon invoking sim, the simulator displays a message indicating that loading has been completed and prints the size (in machine instructions) of the loaded program. It then displays the prompt 'sim>' and awaits a user command. The following commands are recognized by sim:

? or help

Print a summary of all available commands.

run or cont

Run the program or resume execution after a breakpoint. Execution continues until one of the following conditions becomes true:

- a fatal run-time error has occurred in one of the active processors (e.g., multiplication overflow)
- a breakpoint has been encountered, or
- the end of the program has been reached.

step \[ j \]
Execute the next \( j \) instructions. If the specification of \( j \) is absent, a step of one instruction is assumed. Termination conditions are equivalent to the ones in the case of run or cont above.

**pc**

Show the contents of the program counter (pc). The program counter is displayed in hexadecimal, octal, and decimal formats.

**readaddr**

Show the contents of the read-address counter. This value will be used as the source address during the next memory read (i.e., shift north/south with memory read).

**writeaddr**

Show the contents of write-address counter. This value will be used as the destination address during the next memory write (i.e., shift north/south with memory write).

**pfifo**

Show the contents of the Program Queue (pfifo). The list of Program Memory addresses comprising the queue are displayed along with the status of the queue (empty/full/ok).

**wfifo**

Show the contents of the Write Queue (wfifo). The list of Data Memory addresses comprising the queue are displayed along with the status of the queue (empty/full/ok).

**rfifo**

Show the contents of the Read Queue (rfifo). The list of Data Memory addresses comprising the queue are displayed along with the status of the queue (empty/full/ok).

**proc \( i \) \( j \) [ \( f \) ]**
Show the contents of the processor in row $i$ and column $j$. The default format is to display the contents of the processor’s registers and ports as real values (format = %19.16f). If the -$f$ option is specified, then the registers and ports are printed as the hexadecimal equivalent of the 32 bit fixed point internal representation of data (format = %8x). The first format lends itself to a more natural inspection by the user, but contains small roundoff errors incurred by the conversion. Conversely, the hexadecimal format corresponds exactly to the internal data representation, but is difficult to interpret. See Section 2.5 for a detailed description of the processor display format.

**proglen**

Display the length of the program. The length is given in the number of machine instructions and is printed in hexadecimal, octal, and decimal format.

**memory $i$ [ $j$ ] [ $f$ ]**

Display rows $i$ through $j$ of Data Memory. If the specification of $j$ is absent, only the row $i$ is displayed. Each row is preceded by a decimal address of the row. The default format is to print the 16 elements of the row as real values (format = %13.10f), 4 per line. If the -$f$ option is specified, the elements are printed as hexadecimal equivalents of the internal fixed point data representation (format = %8x), 8 per line. In either format, printing of identical rows is suppressed.

**instr $i$ [ $j$ ]**

Display rows $i$ through $j$ of Program Memory. If the specification of $j$ is absent, only the instruction at address $i$ is displayed. Instructions are disassembled and printed in symbolic notation, closely resembling that of the assembly language syntax. Some departures were necessary because certain assembly language instructions translate into more than one machine instruction (e.g., inter-processor I/O instructions). See Section 2.4 for more detail.

**mask label**

Display the mask with the symbolic name label. A simple graphical representation of the processor array is printed and the enabled processors are
marked with "x".

**sysfld code**

Decode the hexadecimal system field code code. The names of the eight system signals are given and the asserted signals are marked with "X".

**dump**

Produce a system dump in file "core.dump". Data Memory, Program Memory, Read, Write, and Program Queues (fifo's), as well as the contents of all 256 processors are written out to a text file for later inspection. Default printing formats are used for the Data and Program Memory.

**break [ label ]**

Set a breakpoint at the symbolic label label. Only one breakpoint can be set at a given label. A total of 20 breakpoints can be active in the system at any given time. If no label argument is given, all currently active breakpoints are listed.

**unbreak label**

Remove a breakpoint at the symbolic label label.

**exit** or **quit**

Exit the simulator.

### 2.3 Data file format

All data in the SCAP system is organized into named data queues of fixed size. Each queue is composed of a non-zero number of rows of data. The system distinguishes between three different kinds of data queues — *ascending*, *descending*, and *constant* queues [Przytula,88]. Ascending queues grow towards higher Data Memory addresses, descending queues grow toward lower Data Memory addresses, and constant queues are comprised of a single row of data.

The simulator obtains the information about the sizes and types of all data queues defined in the original source program from the object code symbol tables. However,
in order for the loader (i.e., the first stage of the simulator) to be able to correctly load data into Data Memory, the user must list the data queues in the data file (datafile) in the same order in which the corresponding type/size definitions appear in the original source program. Each queue must be listed as a sequence of real values in the range \((-2.0, +2.0)\) and it is the user's responsibility to ensure that the sizes of the given data queues agree with those specified in the queue definitions in the source program. All queues should be listed from head to tail, regardless of the type of the queue. The loader phase of the simulator will account for a particular type of a data queue and load the data in the correct memory locations.

In the absence of any information about the exact placement of the queues in the Data Memory (only the sizes of the queues are known), the simulator assumes that data is to be loaded into Data Memory starting at address 0. Therefore, all input data is loaded into the top portion of the memory and extends as far down (towards higher addresses) as necessary. This convention may change in response to a possible future change in the way data queues are declared in the source program — if the syntax of \texttt{ass} is modified to allow the programmer to explicitly specify where in memory particular queues should be located, then the simulator's loader (as well as the actual loader) can be modified to account for that.

2.4 Disassembly conventions

In this section I will briefly present the mnemonics used in the symbolic disassembly, employed by \texttt{sim} to display instructions stored in the Program Memory. A typical disassembled instruction appears as follows:

\begin{verbatim}
0024 R/C OF0FF0F0 0FF NOP CSUM2A NOP NOP CSUM2A NOP NOP B7 ADD2 NOP B7 ADD2
\end{verbatim}

where the first four fields (from left to right) correspond to

1. the decimal value of the program counter associated with the instruction,

2. mask type indicator (R/C = row/column; D = diagonal),
3. hexadecimal encoding of the mask (see [Hartholz,88]) — use mask command (Section 2.2) to decode,

4. hexadecimal encoding of the system field bits (see [Przytula,88]) — use sysfld command (Section 2.2) to decode

The remaining fields encode both phases of the actual instruction. The top line corresponds to Phase 1 and the second line to Phase 2 of the instruction cycle. Moreover (as is suggested by the header), the first three of these fields refer to the external processors (column 1 of the processor array), whereas the second triple of fields refers to the internal processors (columns 2 through 16).

As mentioned above, the mnemonics used in disassembly have been chosen to correspond as closely as possible to the ones used in the assembly language. An important exception are the mnemonics for inter-processor I/O, because a single line of assembly language code expands into two or three (depending on the type of I/O) machine level instructions (see [Przytula,88]). There are a few other minor exceptions. However, all of the additional mnemonics, not present at the assembly language level, are easy to identify and interpret, and a detailed description of the differences will therefore be omitted in this document. Instead, a complete list of the mnemonics, as used by sim, is given below. For convenience, the mnemonics have been grouped into functional categories.

1. registers: (src or dst)
   - AB0 ... AB7 : registers accessible from both buses
   - A0 ... A7 : registers accessible from bus A
   - B0 ... B7 : registers accessible from bus B

2. I/O ports: (src or dst)
   - E:N : to/from East port over bus A, North port over bus B
   - N:E : to/from North port over bus A, East port over bus B
   - S:W : to/from South port over bus A, West port over bus B
   - W:S : to/from West port over bus A, South port over bus B

3. functional unit output registers: (src only)
SUM1A : sum from Adder1 accessible from bus A
CSUM1B : 1's compl. of sum from Adder1 accessible from bus B
SUM2B : sum from Adder2 accessible from bus B
CSUM2A : 1's compl. of sum from Adder2 accessible from bus B
QUOTA : quotient of the Divider unit accessible from bus A
HIGHA : maximum output of Sorter unit accessible from bus A
LOWB : minimum output of Sorter unit accessible from bus B
SFTA&B : outputs of the Shifter unit (accessed together only)

4. Arithmetic operation codes:
ADD2 : addition in Adder2
ADDD : addition in both Adder1 and Adder2
MULTF1 : multiplication in Multiplier1, first stage
MULTS1 : multiplication in Multiplier1, second stage
MULTF2 : multiplication in Multiplier2, first stage
MULTS2 : multiplication in Multiplier2, second stage
DIVS : division with inputs coming from the Shifter unit
DIV : division with inputs coming from the buses
SORT : sort/comparison in the Sorter unit
SHIFT : shift/normalization in the Shifter unit

5. I/O operation codes:
SWL_IN : receive low-order word from South and West neighbors
SWH_IN : receive high-order word from South and West neighbors
NEL_IN : receive low-order word from North and East neighbors
NEH_IN : receive high-order word from North and East neighbors
SWL_OUT : send low-order word to South and West neighbors
SWH_OUT : send high-order word to South and West neighbors
NEL_OUT : send low-order word to North and East neighbors
NEH_OUT : send high-order word to North and East neighbors
SWH&NEL : receive high-order word from South and West neighbors, and send low-order word to North and East neighbors
NEH&SWL : receive high-order word from North and East neighbors, and send low-order word to South and West neighbors
2.5 Processor display format

As mentioned in Section 2.2, the contents of a processor’s I/O ports and/or general purpose registers can be displayed in two different formats — fixed point format (unsigned long) and floating point format (double). An example of a floating point format processor information display (taken from Appendix B) is shown below.

```
          N.in = 0.0199999995529652  N.out = 0.0000000000000000
          W.in = 0.0000000000000000  E.in = -0.0000000009313226
          W.out = -0.0000000009313226  E.out = 0.0000000000000000

          S.in = 0.0000000000000000  S.out = -0.029999993294477

          ABO: 0.0199999995529652
          A1: 0.0000000000000000
          A2: 0.0000000000000000
          A3: 0.0799999982118606
          A4: 0.0000000000000000
          A5: 0.0000000000000000
          A6: 0.0000000000000000
          A7: 0.0000000000000000

          Adder1: (dyn. clock = -1)  Multiplier1: (dyn. clock = -1)
          SUM1A = 0.0399999991059303  PAR_PROD & CARY: inaccessible
          CSUM1B = -0.0400000000372529  ready = y (mul_clock = -1)

          Adder2: (dyn. clock = 4)  Multiplier2: (dyn. clock = 4)
          CSUM2A = -0.0032000001519918  PAR_PROD & CARY: inaccessible
          SUM2B = 0.0031999992206693  ready = y (mul_clock = -1)

          Divider: (dyn. clock = -1)  Shifter: (dyn. clock = -1)
          QUOTA = 0.0000000000000000  SHIFTA = 1.2799999713897700
          ready = y (div_clock = -1)  SHIFTB = 0.6399999856948850

          Sorter: (dyn. clock = -1)
          HIGHA = 0.0000000000000000  column ...................... 0016
          LOWB = 0.0000000000000000  enabled? ..................... yes
```

The topmost box gives the contents of the processor’s I/O port registers. The values of both input and output registers (32 bits) for each of the processor’s four ports are shown (see [Przytula,88] for a discussion of PE architecture). Following the listing
of the general purpose register contents, the template shows the current status of the seven functional units of the given processor. All functional unit output registers (their names appear capitalized in the template) are dynamic registers, which means that the information they store becomes unreliable after 5 clock cycles. Therefore, each functional unit has an associated dynamic clock, which shows the number of cycles remaining until the expiration of the register contents. The value of \(-1\) denotes an expired clock.

Because division and multiplication are multi-cycle operations, two additional global system clocks are maintained by \texttt{sim} — \texttt{mul.clock} and \texttt{div.clock}. Following the loading of the operands, the SCAP system requires 4 cycles to perform a multiplication, and 9 cycles to perform a division. Therefore, the two clocks are set to 4 and 9, respectively, in the cycle in which the corresponding operation was initiated. When the operation has completed, \textit{i.e.}, when the corresponding clock has expired, the results become available in the dynamic output registers, the \texttt{ready} flags are set to \texttt{y} (yes), and the corresponding dynamic clocks are set to 5.

As described in [Przytula,88], the two multiplier units of the SCAP system produce \textit{partial products} and \textit{cary values}, rather than final products. For each multiplier, the two partial results (stored in dynamic registers \texttt{PAR.PROD} and \texttt{CARY}, respectively), must then be added together to yield the final product. Note that \texttt{PAR.PROD} and \texttt{CARY} are not connected to the buses and thus cannot be accessed by the user directly. The reader will notice that the boxes in the above template corresponding to the multiplier units do not give the contents of the \texttt{PAR.PROD} and \texttt{CARY} registers. This is due to the fact that \texttt{sim} simulates multiplication as a monolithic operation, and therefore computes the final product directly, without producing intermediate partial product and cary values. However, this in no way compromises the correctness or usefulness of the simulator, as the SCAP hardware provides no way for the user to access and use these intermediate results. Moreover, all timing information concerning availability and maturity of the result as given by \texttt{sim} is consistent with the actual hardware behavior.

A final note to the user — \texttt{sim} updates all clocks \textit{at the beginning} of each cycle. The clock values displayed at any given point during program execution (\textit{i.e.}, simulation) will therefore be updated in the upcoming cycle \textit{before} the corresponding instruction is executed.


3 Discussion and Suggested Improvements

As of this writing, a couple of fixes to sim are still pending due to the prerequisite corrections/modifications that need to be done to the corresponding assembler. The most important of these concerns the loading stage of the simulator. With the current format of the assembler symbol tables, the simulator can not load descending data queues correctly, because the sizes of the queues are not given as part of the symbol table information. This should be fixed shortly.

Data is represented internally as unsigned long 32-bit integer data (see [Przytula, 88] for a description of the format). All inter-processor I/O, intra-processor data movements, and even some of the functional units (e.g., shifter) work with this format. Only when data is passed to arithmetic functional units, such as adders, multipliers, the divider, or the sorter, the representation changes and the data is converted to floating point format (double). The current conversion scheme effects the conversion of an unsigned long (representing 32-bit fixed point format) into a floating point format by casting the unsigned long into a double and scaling the result by $2^{-30}$. The reverse conversion is analogous. Each such format conversion introduces a slight error, and therefore the error compounds as successive arithmetic computations perform further conversions. An initial error in data representation is introduced by the conversions at loading time, where real values are read from datafile and stored internally as fixed point unsigned longs. Whereas we can clearly not escape some inaccuracies in the presence of multiple representations of data, we may be able to improve on the present scheme.

The current implementation of the simulator (i.e., loader) assumes that scaling of the data into the appropriate range $(-2.0, +2.0)$ has been done by the user and so the data appearing in datafile are expected to be in the correct range. Loader warnings are issued if this is not the case, and zeroes are loaded in place of out-of-range values. Whereas this assumption may seem limiting, it also gives the user greater flexibility in choosing her own scaling factor, which minimizes the loss of precision due to scaling for the particular application. The optimal scaling factor for an application which uses a relatively narrow numerical range of data will be different than that corresponding to an application where a very broad range of data and/or results is expected.

As mentioned above, all data is loaded into the top portion of the Data Memory.
The current syntax of the assembly language enables the user to specify a set of data queues, their types, and their sizes. However, the user is not given the option of specifying the starting addresses of where these data queues should appear in memory. Consequently, the data is loaded into Data Memory sequentially, starting at address 0. Should a future extension to the assembler modify the queue declaration mechanism (and reflect the additional information in the object code symbol tables), the simulator’s loader could be easily adapted to accommodate this new information.

In Section 2.5 we noted that multiplication in sim is handled differently from the actual hardware multiplication. The first stage of the multiplication operation on SCAP hardware produces the partial product and the carry, which are then added in the second stage to give the final result. sim, on the other hand (for reasons of simplicity) computes the final product already in the first stage and simulates the second stage of multiplication as a simple “move” to the appropriate adder output registers, rather than an “add” of the partial product and the carry. As mentioned, this has no effect on the correctness of the results and is completely transparent to the user, as the partial product (PAR_PROD) and carry (CARRY) dynamic registers are not accessible from the buses and thus can not be manipulated by the user. However, in keeping with the goal of faithfully adhering to the architecture and workings of the actual machine, this should perhaps be changed, so that multiplication in sim in fact mimics SCAP’s two-stage multiplication process exactly.

Finally, due to the size and internal complexity of the simulator/debugger system, the system has been only marginally tested. A reasonable stabilizing period of continued usage, testing, and debugging is to be expected.

4 Some Implementational Details

The simulator/debugger is implemented entirely in C and is currently running on a VAX-11/785 under Ultrix V2.0-1. The source code occupies 230 Kbytes of storage (≈ 7,000 source lines of code), and the optimized object code takes up 66 Kbytes of storage.
References


DEFQUESQ 1 16 : data queue (16 rows, asc)
DEFQUESQ 2 1 : data queue (1 row, constant)
DEFMASK 1 (1:2-7,13-17,23-28) : dummy mask (diagonal)
DEFMASK 2 (1-4,9-12,5-8,13-16) : processor mask (row/column)

WOP ;
READQ Q2 ; read from constant queue Q2
LOOP 15 L0 ; every proc has 1 in L0
WOP ;
READQ Q1 ; set read address to Q1
LOOP 15 L1 ; read 16 rows of data

: internal data movements (masked)
L2: MOV (A0,AB0) M2 ;
MOV (AB0,BO) M2 ;
ADD (A0,BO) M2 ;
MOV (SUMB,A1:SUMB,BO) M2 ;
SHFT (A1,BO) M2 ;
MOV (SHFT,A2:SHFTB,BO) M2 ;
ADD (A1,B1) M2 ;
MOV (SUMB,AB3) M2 ;
ADD (A3,4x) M2 ;

L3: MLT2Z (A1,SUMB2) M2 ; multiplication in Multiplier 2 (load)
WOP M2 ;
MOV (A0,BO) M2 ;
WOP M2 ;
WOP M2 ;
WOP M2 ;
MLT2Z M2 ;
MOV (PROD2B,BO) M2 ; B4 = (2x^4) + 4x^2
; subtract : AB3 = B1 (4x - 2x)

L4: ADD2 (,B1) M2 ;
CMPLZ (2x^4 + (-2x^3) - 2x^2)
ADD2 (SUMA,AB3) M2 ;
SUMB2 = 2x^2 + 4x^2 + 2x^2
MOV (SUMB,AB4) M2 ;
AB4 = 2x^4 (the difference)

L5: DIVY (A1,BO) M2 ;
DIVY (B0) M2 ;
DIVS M2 ;
WOP M2 ;
WOP M2 ;
WOP M2 ;
WOP M2 ;
ADD2 (SUMA,AB3) M2 ;
CMPLZ (2x^4 + (-2x^3) - 2x^2)

L5: DIVY (A1,BO) M2 ;
DIVY (B0) M2 ;
DIVS M2 ;
WOP M2 ;
WOP M2 ;
WOP M2 ;
WOP M2 ;
ADD (A3,4x) M2 ;

STOP ; return
Dumping coprocessor contents to file "core.dump" ... done.

quit
A.4 Implementation of Faddeev’s Algorithm
Implementation of Faddeev's Algorithm On A Systolic Array Processor ¹

Yehong Zhang

Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104

Abstract

Much of scientific computing involves matrix operations and this generally becomes the computation bottleneck in problem solving. Faddeev's algorithm provides not only a method general enough to deal with a large variety of matrix operations but also simple enough for systolic implementation. This report describes the Systolic Array Processor built to run Faddeev's algorithm and the coding of the Faddeev's algorithm. To avoid solution instability, the algorithm is slightly modified. The hardware is built by the Hughes' Research Laboratories. Some concrete results will be presented at the end.

¹This paper is based upon work supported by the National Science Foundation under Grant No. ECS-11879. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.
1 Faddeev’s Algorithm: Foundation for Concurrent Matrix Operations

In this paper, we describe the software and hardware implementation of an algorithm for efficient matrix operations of different kinds. This requires an algorithm that is general enough to be used in performing a large variety of matrix operations and yet lends itself for simple parallel implementation. The algorithm that satisfies these requirements is Modified Faddeev’s algorithm and it solves the following problem:

Given $Ax = B$, what is $Cx + D$?

Where $A$ is a nxn matrix, $x$ and $B$ are nx1 column vectors. $C$ is a 1xn row vector and $D$ is a scalar. This is a problem solved in classic linear algebra. First, the values of $x$ have to be solved by using Gaussian elimination and back-substitution. The results of $x$ are then multiplied by $C$ and added to $D$. But there is another way of solving the same problem. Let us first arrange the $A$, $B$, $C$ and $D$ matrices in the following array form:

$$
\begin{array}{c|c}
A & B \\
-C & D \\
\end{array}
$$

Let’s now apply Gaussian elimination to annull the $C$ vector while simultaneously applying the same operations to $B$ and $D$. Assuming that $W$ is the collective result of the Gaussian elimination process, the array looks like the following after Gaussian elimination:

$$
\begin{array}{c|c}
A & B \\
-C & D \\
\end{array} \rightarrow \begin{array}{c|c}
W_A & WB \\
0 & WB + D \\
\end{array}
$$

Obviously $W = CA^{-1}$.

The matrix in the lower right hand quadrant is:

$$WB + D = CA^{-1}B + D = Cx + D$$ (1)

But this is what we are looking for. Thus we have verified that by using the elementary algebraic operations to annull $C$, the result $Cx + D$ will appear in the lower right quadrant. Notice that the solution hinges on the $A$ matrix being non-singular. Thus our result is valid even when $x$, $B$, $C$ and $D$ are compatible rectangular matrices as long as the inverse of the $A$ matrix exists.

There are two problems however which impose serious constraints on the usefulness of Faddeev’s algorithm. The first problem arises when $A$ is a
rectangular matrix in which case the inverse of $A$ becomes non-unique. The second problem is concerned with the actual hardware implementation of the array processor. Partial pivoting in the Gaussian elimination process can be implemented using a systolic array processor only by increasing the complexity of the individual processor and that of the array. To keep the array organization simple and easy to use, partial pivoting is not supported by the Hughes' Systolic Array Processor. However, if no partial pivoting is allowed, division by zero becomes a possibility.

Both of these problems can be solved by modifying Faddeev's algorithm in the following way. Let's assume that $A$ is an arbitrary $m \times n$ matrix. The theorem of QR decomposition guarantees the existence of an $m \times m$ unitary matrix $Q$ and an $m \times n$ upper triangular matrix $R$ such that $A = QR$ [4, 5]. If $m = n$, the $R$ matrix is of the following form:

$$
R = \begin{pmatrix}
    r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\
    0 & r_{22} & r_{23} & \cdots & r_{2n} \\
    0 & 0 & r_{33} & \cdots & r_{3n} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & 0 & \cdots & r_{nn}
\end{pmatrix}.
$$

If $m > n$, the $R$ matrix will take the following form:

$$
R = \begin{pmatrix}
    r_{11} & r_{12} & r_{13} & \cdots & r_{1n} \\
    0 & r_{22} & r_{23} & \cdots & r_{2n} \\
    0 & 0 & r_{33} & \cdots & r_{3n} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & 0 & \cdots & r_{nn}
\end{pmatrix}.
$$

If $m < n$, the $R$ matrix will appear as follows:

$$
R = \begin{pmatrix}
    r_{11} & r_{12} & r_{13} & \cdots & r_{1m} & r_{1m+1} & \cdots & r_{1n} \\
    0 & r_{22} & r_{23} & \cdots & r_{2m} & r_{2m+1} & \cdots & r_{2n} \\
    0 & 0 & r_{33} & \cdots & r_{3m} & r_{3m+1} & \cdots & r_{3n} \\
    \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & 0 & \cdots & r_{mm} & r_{mm+1} & \cdots & r_{mn}
\end{pmatrix}.
$$
If the matrix $A$ has full rank then the diagonal elements of $R$ are not zero. Moreover, since $Q$ is a unitary matrix we have $Q^{-1} = Q^t$. The equation $Ax = B$ can now be changed to: $Rx = Q^tB$.

So in the cases where $m > n$ or $m = n$, Faddeev’s algorithm can again be used to solve the problem. The non-zero diagonal elements of the $R$ matrix are used as the pivots in Gaussian elimination process. The array form of the above operation is as follows:

$$\begin{array}{c|c}
R & Q^tB \\
-C & D \\
\end{array}$$

As long as $A$ is full rank, there will not be any division by zero in the Gaussian elimination process. In the case where $m < n$, it becomes impossible to annull $C$ using Gaussian elimination. Luckily, in most applications, the $A$ matrix is such that either $m > n$ or $m = n$.

Assuming that $R = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}$.

where 0 is the zero portion of the $R$ matrix. Faddeev’s algorithm is thus modified as follows:

First, the $A$ matrix is decomposed according to QR decomposition.

$$\begin{array}{c|c|c|c|c}
A & B & R_1 & Q^tB \\
-C & D & -C & D \\
\end{array}$$

Then Gaussian elimination is carried out using the diagonal elements of $R$ as pivot elements.

$$\begin{array}{c|c|c|c|c}
R_1 & Q^tB & R_1 & Q^tB \\
-C & D & 0 & CR_1^{-1}Q^tB + D \\
\end{array}$$

The modified algorithm will be numerically stable as long as the $A$ matrix has full rank. So by selecting different matrices for the different quadrants, we are able to do a wide varity of matrix operations. Table 1 are just some of the examples of matrix operations which can be performed using the Faddeev’s
algorithm[4].

Let’s use the above algorithm to calculate the pseudoinverse of the 6*7 Jacobian matrix of a redundant arm. Mathematically, the pseudoinverse of the Jacobian can be computed symbolically as follows[6].

\[ J^+ = J^T (J J^T)^{-1} \]

(2)

Where \( J^+ \) is the 7*6 pseudoinverse. \( J \) is the 6*7 Jacobian of the manipulator.

\( J J^T \) can be computed as follows:

\[
\begin{array}{c|c|c}
I & J^T & J J^T \\
\hline
-J & 0 &  \\
\end{array}
\]

\[
\begin{array}{c|c|c}
J J^T & I & (J J^T)^{-1} \\
\hline
-I & 0 &  \\
\end{array}
\]

\[
\begin{array}{c|c|c}
I & (J J^T)^{-1} & J^+ \\
\hline
-J^T & 0 &  \\
\end{array}
\]

2 Hardware Implementation of the Systolic Array Processor

Given the theoretical analysis of Faddeev’s algorithm, a Systolic Array Processor was developed to perform the Faddeev’s algorithm in parallel[4, 3, 10]. The SAP has four major components, a 16x16 array of processing elements, data memory, program memory and control subsystem. The particular design is aimed at obtaining as many levels of parallelism as possible.

The center piece of the SAP is a 16x16 array of processing elements. Each PE is connected to its four immediate neighbors and wrap-around connection is provided for PE’s in the first and last columns. The first row of the array is connected to the bottom of the data memory and the last row to the top of the data memory. Data flows from the bottom of the data memory into the top of the array for processing and the result comes out of the bottom of the array and into the top of the data memory. Figure2 is a schematic diagram of the array. All PE’s are identical and are designed to be able to perform
Examples of Matrix Operations Using Faddeev's Algorithm

Figure 1:
PROCESSOR ARRAY ORGANIZATION

Figure 2:
most of arithmetic operations required by QR decomposition and Gaussian elimination. The on chip functional units are 3 sets of eight storage registers, two adders and two multipliers, a sorter, a normalizer and a divider. To shorten computation time, two buses are used so that operands can be loaded into the appropriate functional units in one clock cycle. Figure 3 is a functional diagram of a processing element. The function of each unit is briefly described below.

**REGISTERS**

The registers labeled AB0-AB7, A0-A7 and B0-B7 are internal storage registers. The first group of eight can be accessed from both buses while the second and third can be accessed from either bus A or B but not both. Aside from these storage registers. There are the I/O registers which are used in inter-processor communication and dynamic output registers for every functional unit in a PE. There are four I/O ports in every PE, namely the north, south, west and east. Every port has two I/O registers one for input and the other for output.

**adders**

There are two adders in every PE. The operands for the adders can be loaded into the adders from both buses. The result in adder A can be accessed from bus A while the 1's complement of the result is accessible from bus B. It is exactly the opposite for adder B. The result of that addition is accessible from bus B and its 1's complement from bus A. The results of addition are available in the cycle following the loading cycle.

**multipliers**

There are two multipliers in every PE. The multiplication takes place in two stages. In the first, the multipliers produce partial product and carry. These then are loaded into the adders. The second stage takes place in the adders where the partial product and carry are added to obtain the product. The results of the first stage are directly loaded into their respective adders without using the buses. The two operands of multiplication are loaded into the multiplier concurrently from the two buses.

**sorter**

The sorter compares two values one coming from bus A and the other from bus B. The larger of the two values will be stored in a dynamic output register accessible from bus A. The smaller of the two values will be stored in a dynamic output register accessible from bus B.

**Normalizer**
Internal Organization of A PE

Figure 3:
Two values $x$ and $y$ are loaded into the sorter $x$ through bus A and $y$ bus B. In one cycle, the shifter will shift both $x$ and $y$ until $x$ falls within the range of 1 and 2. It is mainly used before division.

**DIVIDER**

If the operands are within range, they can be loaded directly into the loader for division. Otherwise, they will have to go through the shifter first and then be loaded into the divider for division.

### 3 System Configuration

The SAP has four major components as noted before. The processor array has been discussed in the previous section and we will briefly discuss the other three components. The SAP is to be controlled by a host computer with a VME bus. For our application, the host will be a SUN 3/160 running UNIX operating system or a real time operating system if it becomes available. Figure 4 is a schematic system diagram[4, 9].

#### 3.1 Data Memory

The data memory is a dual port memory and from Diagram 4 it can be seen that the top port is used by both the host and the processor array. The bottom port is however used exclusively by the processor array. The host uses the top port to load input data into the memory. The processor array access the input data from the bottom port of data memory. Because of the distinctively different functions each port performs, there is a need for maintaining the top and bottom parts of the memory separately. This is accomplished by having one address counter, one FIFO for each port. The data is organized into rows of 16 words each. In other words, the processor array can access 16 words of data at once one for each PE in a row. The data needed is organized into a queue which contains multiple rows of 16 words each. Each element in a given row has the same address as the others in the same row. The address counter for the top port contains the memory address into which a row of 16 words will be written. The address counter for the bottom port contains the memory address from which a row of 16 words will be read into the array. When a data queue is to used for either writing or reading, the beginning address of the queue is loaded into the appropriate
Figure 4: Array System Configuration
queue. The rows of data are sequentially accessed because the content of the appropriate address counter is updated automatically according to whether the queue is ascending, descending or constant.

3.2 Program Memory

Data memory stores the data to be processed and the results of processing. The processing is under the control of a program stored in the program memory. It consists of program memory, program counter, instruction counter and a program FIFO. The program FIFO contains the starting address of the program and all the destination addresses of jump instructions. Before the execution of the program, the host computer loads the starting address of the program from the FIFO into the program counter. The instruction is taken from the memory address indicated by the content of the program counter and loaded into the instruction register. The program counter is automatically updated to provide the address of the next instruction. If a jump instruction is encountered, the destination address of the jump is loaded into the program counter from the program FIFO.

3.3 Control and Clock

The control circuitry receives signals from the host and interprets them. They signals are then used to control various subsystems described above. It checks certain bits of the SAP instruction and flags. It also sets bits of the status registers. The clock system consists of three separate clocks, namely system clock, multiplier clock and divider clock. The system clock starts working once the power is turned on. The other two clocks are turned off until multiplication and division are to be performed. The system clock is to operate at a frequency of 8MHz. The multiplication and division clocks are to operate at 22MHz and 17MHz respectively. It takes 17 multiplication cycles to perform multiplication and 31 division cycles to perform division.
4 Implementation of the Modified Faddeev’s Algorithm Using the Systolic Array Processor

The coding of the Faddeev’s algorithm for the Systolic Array Processor is done at the GRASP Lab of the University of Pennsylvania, with the help of the Hughes Research Labs.

4.1 The Assembler

The first task was to write the assembler for the machine. This was done by Miriam A. Hartholz[2]. The machine instruction set contains around 25 112-bit instructions. These instructions can be roughly separated into 4 categories, namely arithmetic operations, on-chip data transfer, inter-chip communication and system functions. The large number of bits in each instruction was to give the system as much parallelism as possible. That translates into high computational efficiency. Since Given’s rotation in the QR decomposition routine is carried out in the first column of the processor array, the operations in the first column are thus different from those in the remaining processors. The instructions allow different operations to be executed concurrently in the first column and the rest of the array. Another feature of the systolic structure is that at any given time, the same command will be executed by multiple processors in a region of the array. The masking field of the instruction allows the masking of entire regions of the processor array for command execution. For more details, readers are referred to [2].

4.2 The Simulator

Realizing that debugging such a complex parallel system will be difficult, a simulator of the Systolic Array Processor was written by Janez Funda[1]. This simulator allows the user to step through the application program one instruction at a time. The program execution can be halted and the content at that moment of every processor can be examined. This allows the user to view the contents of all the dynamic and static registers on board. The simulator also provides information on I/O between processors. In the coding
of Faddeev’s algorithm, the simulator proves an invaluable tool. A detailed description can be found in [1].

4.3 System Operation

A and B matrices are first broadcast down the array and the processing elements perform Given’s rotation. When the A and B matrices finish going through the array, elements of the R and QB matrices are left in the array. Then matrices C and D are broadcast down the array and the processing elements perform a different set of operations to achieve Gaussian elimination using elements of R as pivots. The results come out row by row at the bottom of the array.

4.4 QR Decomposition Using the Systolic Array Processor

The first step of the Modified Faddeev’s Algorithm is the QR decomposition using Given’s rotation[8]. This can be accomplished when processor in different regions of the array act in the manner indicated by Figure 5.

The sin and cos values used in Given’s rotation are generated in the boundary elements and propagate diagonally through the array. The elements of the A and B matrices are broadcast down the array from the top and turn left when hit the main diagonal and turn downward once they encounter left most subdiagonal. The diagonal values of the resulting R matrix will be stored in the boundary elements.

4.5 Gaussian Elimination Using the Systolic Array Processor

The A and B matrices are immediately followed by the C and D matrices. But as C and D enter the processor array, Gaussian elimination will be performed on them using values stored in the boundary elements as pivots. This means that the processors will be executing a different set of commands than in QR decomposition. The data flow pattern and the different operations corresponding to different regions of the array are given in Figure 6[8].
Figure 5:

OPERATIONS IN QR DECOMPOSITION
OPERATIONS IN GAUSSIAN ELIMINATION

Figure 6:

X_{out} = X_{in} \quad X_{out} = X_{in} - Y \quad X_{out} = X_{in}/r
The matrix elements follow the same data pattern but the processor operations are quite different from before. For example, the boundary processors, instead of generating sin and cos, simply perform a division on the incoming $x$ value. This is the pivoting process. The internal cells perform the subtraction in Gaussian elimination. The result comes out of the processing array one row at a time.

### 4.6 Some Important Procedures

A few procedures are described below which perform square root computation, absolute value operation, branching and transition from QR decomposition to Gaussian elimination[8, 7].

#### 4.6.1 Logical Branching in QR Decomposition

As indicated in Figure 5, the boundary elements generate sin and cos values depending on the current $r$ value and $Xin$. But the machine instruction set does not support logical branching. This however can be overcome by noting that $0/0 = 0$ for this machine. Let's set the following variable:

$$ z = 1 - t/t $$

(3)

Compute sin, cos and $t$ as follows:

$$ t = \sqrt{Xin^2 + r^2} $$

(4)

$$ sin = Xin/t $$

(5)

$$ cos = r/t + z $$

(6)

If $Xin = 0$ and $r = 0$, then $t = 0$ and $z = 1 - 0/0 = 1$ which implies that $sin = 0/0 = 0$ and $cos = 1$. If otherwise, then $t = \sqrt{Xin^2 + r^2}$, $z = 0$, $sin = Xin/t$ and $cos = r/t$. Branching is thus done elegantly using the variable $z$ and the property that $0/0 = 0$.

#### 4.6.2 Square Root Computation

The boundary elements have to perform a square root operation during QR decomposition. This operation has to be carried out using the fundamental
arithmatic operations which are supported by the instruction set, such as addition, multiplication and division. The algorithm we used for evaluating square root is the method of Newton-Raphson Iteration with polynomial approximation.

Let's assume the following:

\[ x = \text{ABS}(x) \]  \hspace{1cm} (7)

\[ y = \text{ABS}(y) \]  \hspace{1cm} (8)

\[ x \geq y \]  \hspace{1cm} (9)

\[ u = y/x \]  \hspace{1cm} (10)

Then

\[
\sqrt{x^2 + y^2} = x\sqrt{1 + u^2}
\]

\[
= (x/2)(f_0 + (1 + u^2)/f_0)
\]

Where \( f_0 \) is computed by:

\[ f_0 = (-0.316394414(1 + u^2) + 1.052146819)(1 + u^2) + 0.259248366 \]  \hspace{1cm} (13)

The results of this method have shown to be very satisfactory.

4.6.3 Absolute Value Computation

As indicated above, there is a need to evaluate the absolute values of \( x \) and \( y \). This is done very nicely by the following two commands.

\[
\text{ADDD} (,x)
\]

\[
\text{SORT} (x,CSUM1B)
\]

The first command adds \( x \) to the unloaded bus in adder 1. The 1's complement of the result (CSUM1B) is the negative of \( x \). Sorting the two will give the absolute value of \( x \).

4.6.4 Transition From QR to Gaussian

It is shown that the processing elements perform different instructions during QR decomposition than during Gaussian elimination. The transition has to be smooth without adding too much computational overhead. This is
achieved by setting another variable $M$ and switching $M$ from 1 to 0 during transition.

Let's assume the following:

$$x' = Mx$$

(14)

Where $M = 1$ for QR and $M = 0$ for Gaussian. For all internal processors, the operations are given below at ALL times:

$$x_{out} = x_{in} \cos - r \sin$$

(15)

$$r = r \cos + x' \sin$$

(16)

For all external processors, the operations are given below at ALL times:

$$t = \sqrt{(x')^2 + r^2}$$

(17)

$$\sin = \frac{x_{in}}{t}$$

(18)

$$\cos = \frac{r}{t} + 1 - \frac{t}{t}$$

(19)

During QR decomposition, $M = 1$. The above equations give:

**Internal Processing Elements**

$$x_{out} = x_{in} \cos - r \sin$$

(20)

$$r = r \cos + x_{in} \sin$$

(21)

**External Processing Elements**

$$t = \sqrt{x_{in}^2 + r^2}$$

(22)

$$\sin = \frac{x_{in}}{t}$$

(23)

$$\cos = \frac{r}{t} + 1 - \frac{t}{t}$$

(24)

These operations correspond with those in section 5.3. During Gaussian elimination $M = 0$, the general equations give:

**Internal Processing Elements**

$$x_{out} = x_{in} - r \sin$$

(25)

$$r = r$$

(26)

(27)
External Processing Elements

\[
\begin{align*}
t &= r \\
sin &= \frac{x_n}{r} \\
cos &= 1
\end{align*}
\] (28) \hspace{5cm} (29) \hspace{5cm} (30)

These correspond with those in section 5.4. Consequently the data is to be organized as follows.

The M value travels down vertically as well as diagonally. The speed of diagonal propagation is twice that of vertical propagation.

5 Computational Cost Analysis

The Systolic Array Processor is built to solve real time computational problems such as common in the control of robots. This section gives a computational analysis of Faddeev's algorithm running on this machine.

5.1 Code Optimization

The assembly program for the Modified Faddeev's Algorithm can be found in Appendix A. The code was optimized by taking advantage of the following three features of the system.

1. There are multiple functional units on every processing element.

2. Boundary and internal elements can be instructed to concurrently execute different commands.

3. Other instructions can be executed while multiplication and division are taking place.

The code can probably be further optimized by a careful time dependancy analysis.
DATE FLOW PATTERN

Figure 7:
DATA PATH

Figure 8:
5.2 Cost

A data flow diagram is given in Figure 8.

As made obvious in the diagram, data enters the array one row at a time. Once it enters the top of the array, all elements in the row travel exactly the same number of steps before coming out. The number of steps taken inside the array is $2n + p$. Since there are $m + l$ rows of data outside of the array, the total number of steps is therefore $2n + p + m + l$. During each step, a set of arithmetic operations is performed in every processor. This set of operations does not change from step to step.

The external and internal processors perform two different sets of operations as indicated by equations 15-16 and equations 17-19. The two sets of operations are independent of each other in the sense that the results of one set do not influence that of the other. This can be taken advantage of by noting that boundary and internal elements can execute different commands concurrently. Therefore, a good measure of operational requirement will be the set with the larger number of operations and which takes longer to execute. In this case, the boundary processors take have many more operations and take much longer to execute than the internal processors, so the operations for the boundary processors will be used as a measure of the computational requirement in each step of the data flow.

First of all, the operations involved in the square root approximation are evaluated. Refering to equations 7-13, they are 1 absolute values (since $r$ is always positive), 1 comparison, 3 divisions, 4 multiplications and 4 additions. The other operations involve 3 divisions, 1 multiplication, 1 subtraction and 1 addition. The total number of different operations are listed below:

- division with shift 6.
- multiplication 5.
- addition 5.
- absolute value 1.
- comparison 1.
- subtraction 1.
For the Hughes' Systolic Array Processor, the following table gives the number of system clock cycles for executing each of the above listed operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>division with shift</td>
<td>11</td>
</tr>
<tr>
<td>multiplication</td>
<td>7</td>
</tr>
<tr>
<td>addition</td>
<td>1</td>
</tr>
<tr>
<td>absolute value</td>
<td>2</td>
</tr>
<tr>
<td>comparison</td>
<td>1</td>
</tr>
<tr>
<td>subtraction</td>
<td>2</td>
</tr>
<tr>
<td>internal transfer</td>
<td>1</td>
</tr>
<tr>
<td>external transfer with memory access</td>
<td>3</td>
</tr>
<tr>
<td>external transfer without memory access</td>
<td>2</td>
</tr>
</tbody>
</table>

Note that the last three entries have nothing to do with arithmetic operations. They are there because data transfer both inside the processing elements and outside of them is a very important part of the systolic array implementation of the Modified Faddeev’s Algorithm and takes up quite a lot of time. Internal transfer refers to the storing of results of computation for later use. It is internal because the computational result will be transferred from a dynamic register of the functional unit to a static register in the same processor. External transfer with memory access refers to the fetching of new rows of the matrices from the memory and broadcasting them into the array and this is done twice in every step, one for one row of new data and one for the value of $M$. External transfer without memory access refers to the data transfer with the array between different processing elements. It includes all data transfer caused by the propagation of $\sin$, $\cos$, $M$ and $x$ inside of the array. The numbers of all above data transfers are listed below.

<table>
<thead>
<tr>
<th>Data transfer</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>internal transfer</td>
<td>15</td>
</tr>
<tr>
<td>external transfer with memory access</td>
<td>2</td>
</tr>
</tbody>
</table>
external transfer without memory access 8.

Given the above information, the total number of cycles in a single step can be calculated as follow:

$$NUM_{cycle} = 6\times11 + 5\times7 + 5\times1 + 1\times2 + 1\times1 + 1\times2 + 15\times1 + 2\times3 + 8\times2 = 148$$  (31)

The total number of cycles required will be the product of 148 and the number of steps necessary in a given task.

$$NUM_{total} = 148 \times (m + l + 2n + p)$$  (32)

Let's assume a clock frequency of K MHz, then the amount of time (in seconds) needed for the task is as follows:

$$T = NUM_{total} \times \left(\frac{1}{K}\right) \times 10^{-6}$$  (33)

Let’s do two simple examples.

**EXAMPLE 1.**
To invert a 6x6 matrix, m=n=l=p=6. Let’s suppose the clock frequency is 8MHz or K = 8. Then the time needed for this inversion is $T = 0.555$ ms.

**EXAMPLE 2.**
To compute the pseudo-inverse of a 6x7 matrix assuming the same clock frequency as above. $T = (1/8)33x148x10^{-6} + (1/8)30x148x10^{-6} + (1/8)31x148x10^{-6} = 1.74$ ms.

6 Summary and Future Work

The fact that this machine can carry out matrix operations very quickly derives from the efficient parallel implementation of the Faddeev’s Algorithm on a parallel computer architecture. Equation 32 indicates that the computational complexity grows linearly with respect to the dimensions of the matrices. This is remarkable for the following reason. Most existing methods of matrix operations have their computational loads increase exponentially with respect to the dimensions of the matrices. Another interesting feature is that $A + B$ takes the same amount of time as $AB$. This is so because the amount of computational time depends only on the dimensions of the matrices and how they are positioned with regards to the array.
The computational capability of this Systolic Array Processor makes it very attractive to robotics research where most of the operations are matrix oriented and time requirement is crucial. Some areas in which this processor could become useful are real-time robot optimal trajectory planning, redundancy control and real-time sensory information processing. All of them require the computation of many matrices in real time.
References


A.5 Compliant Wrist Design
COMPLIANT WRIST DESIGN FOR HYBRID POSITION FORCE
CONTROL

Yangsheng Xu
Richard P. Paul
Peter I. Corke **

General Robotics Active Sensory Perception Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

ABSTRACT

A six DOF compliant wrist which combines passive compliance and active sensing was developed to provide the necessary flexibility for force and contact control as well as being controllable in position. The paper describes such a compliant wrist and sensing mechanism design. Utilizing the sensed information from this wrist allows the apparent stiffness of the end-effector to be increased in unconstrained mode and decreased in constrained modes, where the contact force is controlled. An active feedback control scheme for both position and force control is presented. As the manipulator is partially constrained by environment, the hybrid control algorithm is proposed. Applicability of the control strategy and the compliant wrist is shown by the experiments and the results are discussed under different conditions. The system dynamic performance and the active feedback control scheme are analyzed with a simple model and some useful results are obtained. The paper is concluded and some further research issues are addressed.

1. INTRODUCTION

When robots are used in operations where end effectors contact the environment, passive compliance is beneficial in allowing external constraints to modify the trajectory so as to accommodate geometric uncertainties and dimensional tolerances. There are a number of advantages such as improved positioning capability and relaxed workpiece tolerances. The high forces or moments usually caused by assembly operation or other contact operations can be reduced, and the costly electronic instruments normally required in precision manufacturing avoided[1][5].

Passive compliance is also advantageous for force control. From analysis and experimental results [1][4], the gain of the force control loop can be selected higher if a passive compliance is incorporated in the robot end-effector. When active derivative control is necessary but not permissible in the system, passive compliance can provide significant damping as an alternative.

The manipulator moves between constrained and unconstrained modes continuously. In the constrained mode, force is controlled while in the unconstrained mode position is controlled. Between these two states is a transition. In the transition the force or velocity from which control is achieved may be discontinuous, and the control becomes uncertain. Passive compliance may be applied to accommodate

This material is based on work supported by the National Science Foundation under Grant No. DMC-8512838. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

** Research Scientist, CSIRO Division of Manufacturing Technology, Melbourne, Australia.
this discontinuity, therefore improving the performance of the entire system[2][3].

Passive compliance however presents some problems in robot control if it is not carefully used. The robot end-effector position stiffness is decreased due to the passive compliance, thereby reducing the robotic positioning accuracy. The passive compliance can also present an uncertainty problem in force control. These problems can be avoided only if the deformation information of the passive compliance or position displacement of the end-effector can be measured[1][4][12].

Such a device, with a passive compliance and active sensing together, was developed in our lab. The passive compliance is made of a compliant rubber element yielding compliance along and about all axes. The device is instrumented by providing a simple six joint serial link device with potentiometer sensors in its joints.

The sensing information is used in two ways. In position control, the sensed information is utilized to compensate deflection of the wrist, due to the load or external forces, so as to increase apparent stiffness of the manipulator wrist system. In forces control, the wrist sensor is used as a force sensor by which means the manipulator is driven in the same direction as the sensed force and the desired contact force is maintained.

This paper describes the compliant wrist and sensing mechanism design, and its dynamic performance. Using this device experiments in both position control and force control were conducted and the results are presented. The control strategy for both position and force control is presented. The experimental results show the applicability of the wrist device and the control scheme. The results are discussed under different conditions. As the force only dominates in some of degrees of freedom, while position control is desired in the remaining directions, a hybrid control strategy to such a system is proposed. The system dynamic performance and control schemes are analyzed with a simple system model from which some useful results are obtained. Finally, the paper is concluded and some further research issues are addressed.

2. COMPLIANT WRIST AND SENSOR MECHANISM DESIGN

The device includes two parts: a special rubber element acting as a damped compliance, and a sensing mechanism. The sensing mechanism is a serial link manipulator except that the six joints consist only of potentiometer sensors rather than actuators. The motivation for using such a mechanism is the simplicity of both the device and the direct kinematics. The six joint measurements define the Cartesian position and orientation of the end-effector. The analysis of that mechanism will be discussed later.

The rubber element is chosen because the stiffness of rubber and its shape is such as to yield reasonable stiffness in each direction. Also, from a stability analysis, some damping in the device is necessary as the damping ratio of the system is critical for control stability[1]. The rubber element we have used provides some damping. The stiffness in each direction was measured, and the results are listed in the Table below. The stiffness of the device can be represented in matrix form as

\[
K_w = \begin{bmatrix}
K_{ll} & 0 & 0 & 0 & K_{bl} & 0 \\
0 & K_{ll} & 0 & -K_{bl} & 0 & 0 \\
0 & 0 & K_{aa} & 0 & 0 & 0 \\
0 & -K_{bb} & 0 & K_{bb} & 0 & 0 \\
K_{lb} & 0 & 0 & 0 & K_{bb} & 0 \\
0 & 0 & 0 & 0 & 0 & K_{ll}
\end{bmatrix}
\]

- \(K_{ll}\): Lateral force/lateral displacement;
- \(K_{aa}\): Axial force/axial displacement;
- \(K_{ll}\): Torsional torque/torsional angle;
- \(K_{lb}\): Bending torque/bending angle;
- \(K_{bb}\): Bending torque/lateral displacement;
- \(K_{bl}\): Lateral force/bending angle.
Compliant Wrist Stiffness Characteristics

<table>
<thead>
<tr>
<th>$K_{ll}$</th>
<th>$K_{aa}$</th>
<th>$K_{nn}$</th>
<th>$K_{bb}$</th>
<th>$K_{hl}$</th>
<th>$K_{lb}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(lbs/in)</td>
<td>(lbs/in)</td>
<td>(lbs-in/degree)</td>
<td>(lbs-in/degree)</td>
<td>(lbs)</td>
<td>(lbs/degree)</td>
</tr>
<tr>
<td>2.54</td>
<td>31.75</td>
<td>0.50</td>
<td>3.00</td>
<td>9.77</td>
<td>0.27</td>
</tr>
<tr>
<td>(N/m)</td>
<td>(N/m)</td>
<td>(N-m/grad)</td>
<td>(N-m/grad)</td>
<td>(N)</td>
<td>(N/grad)</td>
</tr>
<tr>
<td>441.00</td>
<td>5512.5</td>
<td>0.056</td>
<td>0.34</td>
<td>43.09</td>
<td>1.19</td>
</tr>
</tbody>
</table>

The reason that we introduced some compliance in each direction is to correct for lateral and torsional errors in assembly operations and to absorb kinetic energy when the robot tool stops suddenly on making contact with the environment. However, the positioning capability will not be degraded because of the active sensing and compensation control in the feedback loop. The compliance in our device is independently programmable in each direction and may be specified according to the task operation and the positioning compensation capability required.

The sensor has to be able to measure six DOF motions of the upper plate related to the lower one. If we consider the device mechanism as a manipulator, this task is simply the direct kinematics, that is, identifying the Cartesian space motion from measured joint space motion. We at first, intended to use a parallel mechanism as in the paper [7], and LVDTs as displacement sensors. However, the direct kinematics is difficult for a parallel mechanism, while inverse kinematics is easy. On the contrary, for a series mechanism the direct kinematics is much easier than the inverse [8], so this was chosen. The computer interface using A/D convertor is also simpler than the digital convertor as the LVDT is used.

The mechanism's kinematic skeleton is as Fig.1 with coordinate frame assigned to the links.

\[
T_w = \text{Trans}(-l_1,l_3,l_1) \text{Rot}(z,\theta_1) \text{Trans}(-l_2,0,0) \text{Rot}(x,\theta_2) \text{Trans}(0,-l_3,0) \text{Rot}(x,\theta_3) \text{Trans}(l_4,-l_5,0) \\
\text{Rot}(z,\theta_4) \text{Trans}(0,l_6) \text{Rot}(y,\theta_5) \text{Trans}(0,l_5,0) \text{Rot}(z,\theta_6) \text{Trans}(l_7,0,l_8)
\] (2-1)

Using the notation in [9], the A transformation matrices for the device are as follows.

\[
A_1 = \text{Trans}(-l_1,l_3,l_1) \text{Rot}(z,\theta_1) = \begin{bmatrix}
C_1 & -S_1 & 0 & -l_7 \\
S_1 & C_1 & 0 & l_3 \\
0 & 0 & 1 & l_1 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (2-2)

\[
A_2 = \text{Trans}(-l_2,0,0) \text{Rot}(x,\theta_2) = \begin{bmatrix}
1 & 0 & 0 & -l_2 \\
0 & C_2 & -S_2 & 0 \\
0 & S_2 & C_2 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (2-3)

\[
A_3 = \text{Trans}(0,-l_3,0) \text{Rot}(x,\theta_3) = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & C_3 & -S_3 & l_3 \\
0 & S_3 & C_3 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (2-4)

\[
A_4 = \text{Trans}(l_4,-l_5,0) \text{Rot}(z,\theta_4) = \begin{bmatrix}
C_4 & -S_4 & 0 & l_4 \\
S_4 & C_4 & 0 & -l_5 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (2-5)
\[ A_5 = \text{Trans}(0,0,l_6) \text{ Rot}(y,\theta_5) = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ 0 & 1 & 0 & 1 \\ -S_5 & 0 & C_5 & l_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  
(2-6)

\[ A_6 = \text{Trans}(0,l_5,0) \text{ Rot}(z,\theta_6) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C_6 & -S_6 & l_5 \\ 0 & S_6 & C_6 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \]  
(2-7)

\[ A_7 = \text{Trans}(l_7,0,l_6) = \begin{bmatrix} 1 & 0 & 0 & l_7 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  
(2-8)

The products of the A transformations for the the device, starting at the upper plate and working back to the robot wrist, are

\[ U_7 = A_7 = \begin{bmatrix} 1 & 0 & 0 & l_7 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & l_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  
(2-9)

\[ U_6 = A_6 U_7 = \begin{bmatrix} 1 & 0 & 0 & l_7 \\ 0 & C_6 & -S_6 & -l_5 S_6 + l_5 \\ 0 & S_6 & C_6 & l_5 C_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  
(2-10)

\[ U_5 = A_5 U_6 = \begin{bmatrix} C_5 & S_5 S_6 & S_5 C_6 & l_7 C_5 + l_5 S_5 C_6 \\ 0 & C_6 & -S_5 & -l_5 C_6 + l_5 \\ -S_5 & C_6 S_5 & C_6 & -l_7 S_5 + l_5 C_5 C_6 + l_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  
(2-11)

let

\[ U_{512} = S_5 S_6 \]
\[ U_{513} = S_5 C_6 \]
\[ U_{514} = l_7 C_5 + l_5 S_5 C_6 \]
\[ U_{524} = -l_5 S_6 + l_5 \]
\[ U_{532} = C_5 S_6 \]
\[ U_{533} = C_5 C_6 \]
\[ U_{534} = -l_5 S_5 + l_5 C_5 C_6 + l_6 \]

\[ U_4 = A_4 U_5 = \begin{bmatrix} C_4 C_5 & C_4 U_{512} + S_4 C_6 & C_4 U_{513} + S_4 S_6 & C_4 U_{514} + l_4 \\ S_4 C_5 & S_4 U_{512} + C_4 C_6 & S_4 U_{513} + C_4 S_6 & S_4 U_{514} - l_4 \\ -S_5 & U_{532} & U_{533} & U_{534} \\ 0 & 0 & 0 & 1 \end{bmatrix} \]  
(2-12)
let

\[ U_{411} = C_4 C_5 \]
\[ U_{421} = S_4 C_5 \]
\[ U_{412} = C_4 U_{512} - S_4 C_6 \]
\[ U_{422} = S_4 U_{512} + C_4 C_6 \]
\[ U_{413} = C_4 U_{513} + S_4 S_6 \]
\[ U_{423} = S_4 U_{513} + C_4 C_6 \]
\[ U_{414} = C_4 U_{514} + l_4 \]
\[ U_{424} = S_4 U_{514} - l_5 \]

Since

\[ A_2 A_3 = \begin{bmatrix}
1 & 0 & 0 & -l_2 \\
0 & C_{22} & -S_{22} & -l_3 C_2 \\
0 & S_{22} & C_{22} & -l_3 S_2 \\
0 & 0 & 0 & 1
\end{bmatrix} \] (2-13)

\[ U_2 = A_2 U_4 = \begin{bmatrix}
U_{411} & U_{412} & U_{413} & U_{414} + l_2 \\
C_{23} U_{421} + S_{23} S_5 & C_{23} U_{422} - S_{23} U_{432} & C_{23} U_{423} - S_{23} U_{433} & C_{23} U_{424} - S_{23} U_{434} + l_3 C_2 \\
S_{23} U_{421} - C_{23} S_5 & S_{23} U_{422} + C_{23} U_{432} & S_{23} U_{423} + C_{23} U_{433} & S_{23} U_{424} + C_{23} U_{434} - l_3 S_2 \\
0 & 0 & 0 & 0
\end{bmatrix} \] (2-14)

let

\[ U_{221} = C_{23} U_{421} + S_{23} S_5 \]
\[ U_{231} = S_{23} U_{421} - C_{23} S_5 \]
\[ U_{222} = C_{23} U_{422} - S_{23} U_{432} \]
\[ U_{232} = S_{23} U_{422} + C_{23} U_{432} \]
\[ U_{223} = C_{23} U_{423} - S_{23} U_{433} \]
\[ U_{233} = S_{23} U_{423} + C_{23} U_{433} \]
\[ U_{224} = C_{23} U_{424} - S_{23} U_{434} + l_3 C_2 \]
\[ U_{234} = S_{23} U_{424} + C_{23} U_{434} - l_3 S_2 \]

\[ U_1 = A_1 U_2 = \begin{bmatrix}
C_1 U_{411} - S_1 U_{221} & C_1 U_{412} - S_1 U_{222} & C_1 U_{413} - S_1 U_{223} & C_1 (U_{414} - l_2) - S_1 U_{224} - l_7 \\
S_1 U_{411} + C_1 U_{221} & S_1 U_{412} + C_1 U_{222} & S_1 U_{413} + C_1 U_{223} & S_1 (U_{414} - l_2) + C_1 U_{224} + l_3 \\
U_{231} & U_{232} & U_{233} & U_{234} + l_1
\end{bmatrix} \] (2-15)

The determinant of the Jacobian matrix was calculated to investigate singularities. The results show that there is no singularity around the home position where the device works.
3. POSITION CONTROL

Experiments with the compliant wrist which is shown in Picture were performed on a PUMA 560. A video to demonstrated the experiment is available in the conference time. Before the experiment, six potentiometers were adjusted in a proper range and the compliant wrist sensor was calibrated carefully. The control was executed on a Microvax 2 using the RCI primitives of RCCL [6], which allowed the software to directly command robot joint angles. The software package allowed various parameters to be set, and also allowed trajectory and wrist displacement data to be logged to a file for subsequent analysis. A block diagram is shown in Fig.2 for the experimental control system.

Position control of the robotic system, including a compliant wrist, was proposed to increase the apparent stiffness of the device by moving the manipulator in the opposite direction to the sensed displacement. In other words, the deflection of the compliant wrist due to load or external forces is compensated and the positioning accuracy of robot manipulator is maintained. A detailed analysis for selecting proper controller parameters, as well as the effect of the compliant wrist on the system performance, can be found in our earlier papers [1][2].

Two different control schemes, Cartesian space control and the joint rate control, were investigated for position control.

In Cartesian space control, we suppose the transformation between the base and robot wrist coordinate is $T_6$, that between two plates of the compliant wrist is $T_w$, and that between the base and upper plate of the compliant device is $B$ which is assumed as the task coordinate transformation. The kinematic relation at the initial state is

$$ T_6 T_w = B $$ (3-1)

Suppose at the current state the compliant wrist coordinate frame $T_w$ is changed to $T'_w$ due to the load or other external force, the task coordinate transformation is changed to $B'$, so that the current state kinematics relation is

$$ T_6 T'_w = B' $$ (3-2)

In the position control case, it is our aim that the robot coordinate transformation $T_6$ can be modified to $T'_6$ so that the task coordinate transformation $B$ is not changed. Therefore, the control strategy is

$$ T_6 T'_w = B $$ (3-3)

Equating (3-1) and (3-3) yields

$$ T'_6 T'_w = T_6 T_w $$

or,

$$ T'_6 = T_6 T_w (T'_w)^{-1} $$ (3-4)

An alternative is the joint rate control utilizing the differential displacement of the compliant wrist. There are two ways to get a differential displacement vector $\Delta X_w$ with three position displacements and three orientations related to the initial position where the deflection is zero. Firstly the three angles and displacements may be extracted from the updated transformation matrix of the compliant wrist $T_w$. Secondly these six differential displacements may also be calculated from the wrist mechanism Jacobian matrix $J_w$ and $\Delta \theta_w$ which are measured from in potentiometer readings in the sensor mechanism.

$$ \Delta X_w = J_w \Delta \theta_w $$ (3-5)

Results for both methods are the same, but we chose to use the first approach.

Joint rate control of the manipulator is defined [9] as

$$ \Delta X = J_m \Delta \theta_m $$ (3-6)

where $J_m$ and $\Delta \theta_m$ are the Jacobian matrix and joint differential change of the manipulator. Since the compensation is desired in position control so that the manipulator moves in the opposite direction as the displacement, $\Delta X$ in (3-6) should be negative of $\Delta X_w$ in (3-5). Therefore,
\[
\Delta \theta_m = -J_m^{-1}\Delta X_w
\]

More generally we can add a gain matrix \( K_p \) so that the desired joint command vector \( \theta_{des} \) becomes

\[
\theta_{des} = \theta_{req} + \Delta \theta_m = \theta_{req} - J_m^{-1}K_p \Delta X_w
\]

where \( \theta_{req} \) is the desired joint angles, perhaps supplied by a trajectory generator function.

Because the sensor has mechanical backlash and electronic noise, the sensor information is passed through a unity-gain first order digital filter

\[
\frac{Y(z)}{U(z)} = \frac{1-a}{z-a}
\]

The low pass corner frequency is controlled by the pole of the digital filter \( a \)

\[
a = e^{\omega_o T}
\]

where \( T \) is the sample interval, in our case 28ms.

The experiment consisted of an incremental change in load, in or around a certain direction recording the six DOF wrist deformations and six DOF manipulator endpoint position response. Shown in Fig.4 are the results as the load is applied around the x direction, with a gain \( K_p \) of 1.0, and \( a \) the digital filter pole set to 0.98. The upper curve is the observed wrist angle deformation around the x direction \( \theta_x \) and the lower curve is a recording of the manipulator endpoint response in the same direction. From these curves, we can see that the manipulator moves in the opposite direction but by the same amount as the wrist deformation so that the absolute endpoint location is maintained.

The experimental results indicate that the use of an active sensing of the compliant wrist makes it possible to retain the original static stiffness characteristics of the manipulator in spite of the presence of a passive compliance in the wrist.

The response of the digital filter dominates the response time, but a faster filter response yields a rougher control due to noise from the sensor. Choice of the filter pole is critical to performance of the system. Using the same load condition as the experiment shown in Fig.4, the pole is set to 0.80 and the results are shown in Fig.5 where we see the settling time is much shorter. If we lower the pole still further to 0.50, the results as shown in Fig.6 indicate the system is not stable. That is because the filter corner frequency (4.0 HZ, if the pole is 0.50) is close to the first resonance frequency (around 3.8 HZ in this case) which can be verified by calculation of the natural frequency from the system parameters. From the analysis given in a later section, the system dynamics becomes dominated near the first resonance frequency and the gain margin of the original system is less than zero if an unit feedback gain is used, therefore the system become unstable. The filter with a corner frequency less that the first resonance frequency of the system can increase the gain margin and thereby stabilize the system. Hence, the corner frequency of the filter must be less than the first resonance frequency of the system. For the given passive compliance and load condition, this effectively puts a lower bound on the response time of the system. The softer the passive compliance or the larger the load, the lower corner frequency must be set and the slower the response becomes. Therefore, the filter corner frequency must be adjusted to ensure stable closed-loop behavior under the worst case of passive compliance and load condition.

4. FORCE CONTROL

In the force control case where the end-effector is partially constrained by the workpiece, we use the compliant wrist as a sensor to detect the force generated in the end-effector. This is used to drive the manipulator in the same direction as the deformation of the wrist, i.e., the generalized force direction so that the apparent stiffness is decreased and a desired contact force is maintained.

We utilize a joint rate control scheme for the force control experiments. As in the position control case, we obtain the six DOF generalized displacement of the wrist \( \Delta X_w \).

Rate control of the manipulator is given by

\[
\Delta \theta_m = J_m^{-1}\Delta X
\]

and the displacement \( \Delta X \) is related to the exerted force \( F_w \) by the desired stiffness \( K_d \).
\[ F_w = K_d \Delta X \]  
\[ (4-2) \]

whereas measured displacement is related to exerted force by

\[ F_w = K_w \Delta X_w \]  
\[ (4-3) \]

where \( K_w \) is the actual stiffness of the wrist. Substituting yields

\[ \Delta X = K_s F \Delta X_w \]  
\[ (4-4) \]

where \( K_F \) is a dimensionless stiffness ratio

\[ K_F = K_d^{-1} K_w \]  
\[ (4-5) \]

Equation (4-2) becomes

\[ \Delta \theta_m = J_n^{-1} K_F \Delta X_w \]  
\[ (4-6) \]

The desired joint angles are thus

\[ \theta_{\text{des}} = \theta_{\text{curr}} + \Delta \theta_m = \theta_{\text{curr}} + J_n^{-1} K_F \Delta X_w \]  
\[ (4-7) \]

Comparing (4-7) to (3-8), we may note

(1) The equations are very similar, both contain a dimensionless gain term \( K_F \) or \( K_F \) with the different sign in front. The former represents the gain by which we want to modify the displacement of the wrist according to how much amount of the deflection we expect to compensate in position control. The latter represents another gain matrix by which the stiffness of the end-effector relates to the effective stiffness of the system according to how much compliance is required in the force control task. If complete compensation in all directions is required in position control, the gain \( K_F \) is identity matrix. If the desired compliance level is just the natural compliance, \( K_w \), of the wrist, the gain matrix \( K_F \) is again the identity matrix.

(2) Since in position control the end-effector is supposed to move in the opposite direction to the displacement we measured while it is to move in the same direction in force control, the differential displacement \( \Delta \theta_m \) in the Equation (3-8) is negative while positive in the Equation (4-7).

(3) Provided the end-effector is in steady state and a constant deformation (i.e., force) exists in the compliant wrist, the manipulator should keep moving in the force control mode while it should stop if a constant compensation has been already responded to in position control mode. Therefore, the desired joint angles \( \theta_{\text{des}} \) should be based on the specified joint angles \( \theta_{\text{adj}} \) in position control (3-8), but based on the current joint angles \( \theta_{\text{curr}} \) in force control (4-7).

To check the force control mode, a constant force was applied suddenly, simulating the end-effector coming into contacted with environment. Shown in Fig.7 are the results of the deformation of the wrist and the endpoint response of the manipulator provided the desired force level is zero. At first, the force was applied and the wrist was deformed, and the manipulator end-point was moving in the same direction as the measured force. Then, as the manipulator reached the desired force level which is specified as zero, the manipulator moves from the surface, the wrist has no deformation and the manipulator stops.

5. SYSTEM ANALYSIS

We may analyze the system dynamic performance and the position and force control strategy described above with a simple single degree of freedom system as shown in Fig.8. The end-effector is represented as a mass \( m \) and the compliant wrist is represented as a spring with stiffness \( K_w \) and a viscous damping \( C_w \), which are attached to the manipulator end-point. The end-effector motion is \( x_2 \) and the manipulator end-point motion is \( x_1 \). The compliant sensor records the difference of the motions \( \Delta x = x_2 - x_1 \). The controller uses the error information \( \Delta x \) with the end-point command \( x_{1c} \) together to drive the end-point motion \( x_1 \), so that the end-effector motion \( x_2 \) is not affected by the external force \( F_{\text{ext}} \) in position control while dominated by \( F_{\text{ext}} \) in force control.

Using the position control and force control scheme discussed above to this simple system, the control strategy can be described by the block diagram in Fig.9 where the feedback loop can switch to a negative gain in position control and to a positive gain in force control. Without losing generality, we assume
the desired force is zero. The transfer function from \(x_1\) to \(x_2\), \(G_1\), and from the external force \(F_{ext}\) to \(x_2\), \(G_2\) can be obtained from the system in Fig.8.

\[
G_1(s) = \frac{X_2(s)}{X_1(s)} = \frac{K_w + C_w s}{m s^2 + C_w s + K_w}
\]

\[
G_2(s) = \frac{X_2(s)}{F_{ext}(s)} = \frac{1}{m s^2 + C_w s + K_w}
\]  

(5-1)  

(5-2)

In position control, the system function can be derived,

\[
(1-K_F)G_2 F_{ext} + G_1 X_{1c} = (1-K_F + K_F G_1)X_2
\]

(5-3)

To make a comparison, the system function for Fig.8 without active feedback control may also be written as

\[
G_2 F_{ext} + G_1 X_{1c} = X_2
\]

(5-4)

From (5-3), we may be clear that the effect of the external force is decreased by \((1-K_F)\). If a unit gain \(K_F\) is used, the external force can be eliminated and the system is totally independent of the force \(F_{ext}\).

The system dynamic stiffness \(Z(s)\) can be derived from (5-3) if the second term is not considered.

\[
Z(s) = \frac{F_{ext}(s)}{X_2(s)} = m s^2 + \frac{C_w s}{1-K_F} + \frac{K_w}{1-K_F}
\]

(5-5)

We also can derive it for the case without feedback control (5-4).

\[
Z(s) = \frac{F_{ext}(s)}{X_2(s)} = m s^2 + C_w s + K_w
\]

(5-6)

Therefore, the virtual stiffness in steady state is increased by \(1/(1-K_F)\) due to introduction of the active feedback and the entire system has a better positioning ability despite the presence of the passive compliance.

Similarly in force control the system function can be derived as

\[
(1+K_F) G_2 F_{ext} + G_1 X_{1c} = (1+K_F - K_F G_1) X_2
\]

(5-7)

Here we may see the feedback control make the external force dominate the system. The dynamic stiffness is

\[
Z(s) = \frac{F_{ext}(s)}{X_2(s)} = m s^2 + \frac{C_w s}{1+K_F} + \frac{K_w}{1+K_F}
\]

(5-8)

Comparing (5-8) to (5-6), the virtual stiffness of the system is decreased by \(1/(1+K_F)\) times. The higher gain is set, the softer the system becomes.

We may obtain the system characteristic equation from (5-3) and (5-7) for position and force control respectively as follows.

\[
(1-K_F) m s^2 + C_w s + K_w = 0
\]

(5-9)

\[
(1+K_F) m s^2 + C_w s + K_w = 0
\]

(5-10)

Utilizing the feedback gain \(K_F < 1\) in position control which ensures the system stability from (5-9). If unit gain is used, which can provide complete compensation as shown in our paper[1], the system open-loop function \(G_1\) (5-1) can present a gain margin less than zero. To stabilize the system, a lowpass filter is beneficial as discussed in the early section.

The equation (5-10) indicates that the system is always stable in force control. However, as the gain \(K_F\) is increased, the pole moves closer to the origin and the relative stability is affected. Since the location of the pole is dependent on the value \(K_w/m(1+K_F)\) as well as \(C_w\), the gain has to be selected according to the physical stiffness, damping and mass. The higher the stiffness and damping of the passive compliance

* i.e, impedance, the term "stiffness" should only make sense in statics, but one gets accustomed to use this term and so do we.
device and the smaller the end-effector mass, the more force gain can be used. On the other hand, the force gain selection is also dependent on the desired pole assignment of the system. Since the physical and desired stiffness in each direction or around each axis are different, the force gains were set differently in our experiment.

We may note another fact that the virtual damping of the system is also increased in position control while decreased in force control if Equation (5-5) and (5-8) are compared to (5-6). For position control, this causes a time delay. For force control, it produces a less damped system so that the system may be unstable as observed by others[14]. Therefore, the gain \( K_F \) or \( K_F \) and the natural passive damping must be selected properly.

If the environmental compliance must be considered, the contact force can be modeled as spring and damper between the end-effector \( x_2 \) and a rigid hypothetic surface \( X_3 \) as shown in Fig.10. The stiffness and damping of the environment are represented as \( K_e \) and \( C_e \). The problem can be interpreted as the compliance between \( x_2 \) and \( X_3 \) is controlled to a desired level. Hence, the force control problem is actually the problem of controlling the compliance between the end-effector and the environment. In our case, the problem can be specified so that the end-effector motion \( x_2 \) is modified by adjusting the manipulator end-point motion \( X_1 \) in the presence of change in the environmental characteristics (including surface geometry and material elasticity). In this way the compliance between \( x_2 \) and \( X_3 \) is maintained in a desired level. The control block diagram can be shown in Fig.11. Similarly as in the analysis above, we can derive the system transfer functions as follows.

\[
G_1 = \frac{C_w s + K_w}{m s^2 + (C_e + C_w) s + (K_e + K_w)}
\]

\[
G_3 = \frac{C_e s + K_e}{m s^2 + (C_e + C_w) s + (K_e + K_w)}
\]

\[
\frac{X_2}{X_3} = \frac{(1+K_F)G_3}{1+K_F-K_F G_1} = \frac{C_e s + K_e}{m s^2 + \left(C_e + \frac{C_w}{1+K_F}\right) s + \left(K_e + \frac{K_w}{1+K_F}\right)}
\]

The equation (5-13) and (5-12) indicate clearly that the active feedback control makes the system softer and the compliance of the contact can be controlled by adjusting the feedback gain \( K_F \).

6. HYBRID CONTROL

It is well known that every manipulator task can be broken down into elemental components that are defined by a particular set of contacting surfaces, a generalized surface can be defined in a constraint space having six degree of freedom, with position constraints along the normal to this surface and forces constraints along the tangents. These two type of constraints, force and position, partition the degree of freedom of possible end-effector motions into two orthogonal sets, that must be controlled according to different control strategies (3-8) and (4-7). Since the desired angles \( \theta_{des} \) is based on \( \theta_{tra} \) in position control (3-8), but based on \( \theta_{con} \) in force control (4-7), the hybrid control can not simply combine Equation (3-8) and (4-7) together. Here we present a hybrid control scheme for the robot system which includes a passive compliance as follows.

First, we partition \( \Delta X_w \) which is the Cartesian error measured from the wrist sensor into two sets \( \Delta X_{\xi} \) corresponding to the direction in which force control is required, and \( \Delta X_{\xi} \) in the remaining directions in which the position control is required. For example, if the force in the z direction and the torques around the x and y directions are controlled and in the remaining directions are position controlled, we partition \( \Delta X_w \)

\[
\Delta X_w = \begin{bmatrix} \Delta x & \Delta y & \Delta z & \Delta \theta_x & \Delta \theta_y & \Delta \theta_z \end{bmatrix}^T
\]

into

\[
\Delta X_{\xi} = \begin{bmatrix} 0 & 0 & \Delta \theta_x & \Delta \theta_y & 0 \end{bmatrix}^T
\]

\[
\Delta X_{\xi} = \begin{bmatrix} \Delta x & \Delta y & 0 & 0 & \Delta \theta_z \end{bmatrix}^T
\]
Then, we multiply these vectors by the gain matrices $K_F$ and $K_F$ as in (3-8) and (4-7) to obtain the desired differential motion of the end-effector.

$$\Delta X_P = K_F \Delta X_P^E$$  \hspace{1cm} (6-4) \\
$$\Delta X_F = K_F \Delta X_F^E$$  \hspace{1cm} (6-5)

where $\Delta X_P$ is compensated in position control mode and $\Delta X_F$ is controlled in force control mode provided that the desired force level is zero.

From differential motions $\Delta X_P$ and $\Delta X_F$ we can form differential transform matrices [9] $T_{\Delta X}$ and $T_{\Delta X_F}$ respectively. When force control is considered in Cartesian space, the desired motion of the end-effector is with respect $T^{ij}$, where the superscript $j$ refers to the time.

$$T^{ij} = T_{\Delta X}^{-1} \ast T_{\Delta X}$$  \hspace{1cm} (6-6)

Since we must consider the deflection of the end-effector in the presence of the passive compliance, the required motion $T^{ij}$ has to be modified by the differential motion $T_{\Delta X}$ which represents the deformation of the compliant wrist in (6-4). Therefore, the required motion $T^{ij}$ to yield the desired motion $T^{ij}$.

$$T^{ij} = T^{ij} \ast T_{\Delta X}$$  \hspace{1cm} (6-7)

Thus, the end-effector not only provides the desired compliance in the specified degrees of freedom, but also compensates the deflection of the compliant wrist simultaneously.

Similarly in joint space, we also can perform hybrid control. The joint rate control scheme from Equation (3-6) is

$$\Delta \theta = J_m^{-1} \Delta X$$  \hspace{1cm} (6-8)

The desired joint angles of the end-effector must move in the same direction as the differential joint angles caused by $\Delta X_P$, $J_m^{-1} \Delta X_P$, based on the current desired joint angles as in Cartesian space control (6-6).

$$(\theta_{\text{des}})_j = (\theta_{\text{des}})_{j-1} + J_m^{-1} \Delta X_P$$  \hspace{1cm} (6-9)

Also, the end-effector motion must be modified by the differential joint angles which represents the deflection of passive compliance $\Delta X_P$ in (6-4), $J_m^{-1} \Delta X_P$.

$$(\theta_{\text{req}})_j = (\theta_{\text{des}})_j - J_m^{-1} \Delta X_P$$  \hspace{1cm} (6-10)

Equation (6-9) and (6-10) represents hybrid position and force control in joint space in correspondence with Equation (6-6) and (6-7) for Cartesian space. We chose the latter method because of simplicity. An experiment based on this control scheme was performed. We specified that the force along the $z$ axis and the torques around the $x$ and $y$ axes while in the remaining directions position control was performed. Force in an arbitrary direction was applied and the wrist deformation as well as the end-effector motion was recorded. The experimental curves in force control mode are identical with the curves shown in Fig. 7 and those in position control mode are identical with the curves shown in Fig.5. The results were satisfactory and demonstrated the control scheme is stable.

7. CONCLUSIONS

(1) The compliant wrist performs successfully both in passive compliance and active sensing. Using such a device makes it possible to provide the system with a flexibility which simplifies contact force control and the transient state control, and compensates the end-effector deformation due to the external forces.

(2) The applicability of the position and force control strategies is shown by experiment. In position control, the manipulator moves in the opposite direction to the deformation of the wrist so that the apparent stiffness of the end-effector is increased. In force control, the manipulator is driven in the same direction as the sensed force so as to decrease the stiffness.

(3) Force feedback gain should be selected carefully according to the stiffness of the passive device, the mass of the end-effector, and the desired compliance of the system. The different force gains are necessary for each direction and around each axis because the physical stiffness, the effective inertia or
mass, and the desired compliance are different.

(4) Hybrid position and force control for the robot system including passive compliance is possible and the control scheme's feasibility is demonstrated by the experimental results. The scheme can also be utilized as a free joint control scheme, as well as a conventional hybrid control without passive compliance.

(5) Because of the mechanical and electronic noise of sensing information, the filter dynamics dominate the system's response. The pole of the first order filter can be selected according to the load condition and different control modes.

REFERENCES


Fig. 1 Kinematic skeleton of the Wrist sensor mechanism

Fig. 2 Block diagram of the control system of the manipulator and compliant wrist

Fig. 3 First order filter
Fig. 4 Deformation reading of the wrist and manipulator end-point response in position compensation (pole of the filter is 0.98)
Fig. 5 Deformation reading of the wrist and manipulator end-point response in position compensation (pole of the filter is 0.80)
Fig. 6  Deformation reading of the wrist and manipulator end-point response in position compensation (pole of the filter is 0.50)
Fig. 7  Deformation reading of the wrist and manipulator end-point response in force control (pole of the filter is 0.50)
Fig. 8  Simple model of the wrist system

Fig. 9  Block diagram of the active feedback control
Fig. 10  Simple model of the wrist system and environment

Fig. 11  Block diagram of the active compliance control
A.6 Hybrid Control in the Presence of Passive Compliance
THE IMPLEMENTATION OF HYBRID CONTROL IN THE PRESENCE OF PASSIVE COMPLIANCE

Richard P. Paul
Yangsheng Xu
General Robotics Active Sensory Perception Laboratory
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

ABSTRACT

A new compliance motion control method combining passive compliance and active control is presented. Position compensation and force control of the robot manipulator in the presence of passive compliance is discussed and a new hybrid control scheme is derived. The applicability of the method is demonstrated by experimental results. The entire system performance is analyzed under different conditions, such as environment characteristics, feedback gain, digital filter, and contact force, etc. A sinusoid surface tracking experiment was performed as an application of the hybrid control scheme and some useful results were obtained. The velocity discontinuity as the robot makes contact with, or breaks from, the environment is accommodated by the passive compliance at the wrist. The method shown is simple, economical, and applicable in industry.

1. INTRODUCTION

Today's robot needs to provide for a more sophisticated compliant motion. Considerable attention has been directed to compliant motion of robot manipulators in this decade. We may categorize currently available compliance motion control techniques as two basic types. One is the active compliance which is specified in the joint servo either by setting a linear relation between the force and displacement (or velocity and displacement) such as impedance control [14], damping control [24], stiffness control [12], or by controlling force in certain degrees while controlling position in the remaining degrees, such as compliance control [19], compliance and force control [11], hybrid control [15]. Another is the passive compliance which is provided by an additional tool near the end-effector such as a wrist, hand, or fingers. The most famous one is the Remote Center of Compliance (RCC) [4] although many different versions have been developed in Japan [20][21], France [16][22], West Germany [18] and USA [13][23].

There are fundamental problems for both techniques. For the active compliance, an instability problem in a stiff environment has been observed, thus a passive compliance installed in the end-effector is desirable to reduce the overall system stiffness. For the passive compliance alone, a positioning capability of robot is degraded. Based on these two main problems, much research has been performed recently [8][9][10][17], but a simple, economical and reliable method is still demanded so that the compliant motion of the robot manipulators could be finally implemented in industry.

In this paper, we propose to use a passive compliance mechanism with six DOF compliance which is also capable of measuring of six DOF deflection of the device between the end-effector and robot wrist. Passive compliance can correct the positioning error automatically and relax the tolerance as well as

This material is based on work supported by the National Science Foundation under Grant No. DMC-8512838. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.
accommodate the transition between the position and force control modes. The sensed deflection in the wrist can be used for feedback control such that the entire system is controllable.

Such a device, with a passive compliance and active sensing mechanism together, was developed in our lab. The passive compliance is made of a compliant rubber element yielding compliance along and about all axes. The device is instrumented by providing a simple six joint serial linkage with potentiometer sensors at its joints. The sensing information is used in two ways. In position control, the sensed information is utilized to compensate deflection of the wrist, due to the load or external forces, so as to increase apparent stiffness of the manipulator wrist system. In force control, the wrist sensor is used as a force sensor by which means the manipulator is driven in the same direction as the sensed force and the desired contact force is maintained.

The paper is organized as follows. In the first section, the compliant wrist device combining passive compliance and sensing mechanism is described. In the second section, position compensation of the robotic system in the presence of a passive compliance is investigated. Two control algorithms in Cartesian space and joint space is proposed. The analytical and experimental results are illustrated. In the third section, force control scheme is presented and demonstrated by the experiment and analysis. In the fourth section, based on the position and force control scheme, a new hybrid control strategy is developed. In the fifth section, a sinusoid surface tracking experiment is performed as an application of the hybrid control scheme and some useful results are obtained. In the sixth section, the system performance is analyzed and an effect of a digital filter on the entire system is discussed. We conclude the paper in the last section.

2. PASSIVE COMPLIANCE AND SENSING MECHANISM

The compliance device includes two parts: a special rubber element acting as a damped compliance, and a sensing mechanism to measure the deflection of the device during the end-effector motion. The sensor mechanism is capable of measuring six DOF motions of the upper plate of the device related to the lower one. The mechanism is a serial linkage mechanism with six potentiometer sensors at its joints instead of six actuators for the manipulator. The task of the computation from the sensed data of the joints is simply the direct kinematics so that the end-effector motion is identified, while that for the robot manipulator is the inverse kinematics so that the joint motion is determined from the desired Cartesian trajectory at the end-effector. We at first, intended to use a parallel mechanism as in the paper [6], and LVDTs as displacement sensors. However, the direct kinematics is difficult for a parallel mechanism, while inverse kinematics is easy. On the contrary, for a series mechanism the direct kinematics is much easier than the inverse [7], so this was chosen. The computer interface using A/D converter is also simpler than the digital converter as the LVDT is used. The mechanical structure of the serial linkage is easier to make than that of the parallel one. Only disadvantage is the error accumulation of the serial mechanism, while the error is compensated in the parallel mechanism. We, however, use the filter and carefully calibrate the mechanism and potentiometers, so the designed precision of the device is obtained. The device feature is shown in Fig. 1.

The rubber element is chosen because the stiffness of rubber and its shape is such as to yield reasonable stiffness in each direction. Also, from a stability analysis, some damping in the device is necessary as the damping ratio of the system is critical for system performance [1]. The rubber material in the device can provide significant inherent damping.

We introduced passive compliance in each of six directions instead of three or five directions for most of passive compliance devices. The reason is that the device is used not only to correct lateral and torsional errors in assembly operations but also to absorb kinetic energy when the robot tool stops suddenly on making contact with the environment so that the transition between the force and position is accommodated. The positioning capability will not be degraded because of the active sensing and compensation control in the feedback loop.

3. POSITION CONTROL

Experiments with the compliant wrist shown in Fig. 2 were performed on a PUMA 560. A video to demonstrate the experiment is available in the conference time. Before the experiment, six potentiometers were adjusted in a proper range and the compliant wrist sensor was calibrated carefully. The control was executed on a Microvax 2 using the RCI primitives of RCCL [5], which allowed the software to directly command robot joint angles. The software package allowed various parameters to be set, and also allowed
trajectory and wrist displacement data to be logged to a file for subsequent analysis.

Position control of the robotic system, including a compliant wrist, was proposed to increase the apparent stiffness of the device by moving the manipulator in the opposite direction to the sensed displacement. In other words, the deflection of the compliant wrist due to load or external forces is compensated and the positioning accuracy of robot manipulator is maintained. A detailed analysis for selecting proper controller parameters, as well as the effect of the compliant wrist on the system performance, can be found in our earlier papers [1][2].

Two different control schemes, the Cartesian space control and the joint rate control, were investigated for position control.

In the Cartesian space control, we define the transformation from the base coordinate to the attached plate of the compliant wrist device is $T_6$, that between two plates of the compliant wrist is $T_w$, and that between the base and upper plate of the compliant device is $B$ which is considered as the task coordinate transformation. The kinematic relation at the initial state is

$$T_6 T_w = B \quad (1)$$

Suppose at the current state the compliant wrist coordinate frame $T_w$ is changed to $T_w'$ due to the load or other external force, the task coordinate transformation is thus changed to $B'$, so that the current state kinematics relation is

$$T_6 T_w' = B' \quad (2)$$

In the position control case, it is our aim that the robot coordinate transformation $T_6$ can be modified to $T_6'$ so that the task coordinate transformation $B$ keeps unchanged. Therefore, the control strategy is

$$T_6' T_w' = B \quad (3)$$

Equating (1) and (3) yields

$$T_6' T_w' = T_6 T_w$$

or,

$$T_6' = T_6 T_w (T_w')^{-1} \quad (4)$$

An alternative is the joint rate control utilizing the differential displacement of the compliant wrist. There are two ways to get a differential displacement vector $\Delta X_w$ with three position displacements and three orientations related to the initial position where the deflection is zero. Firstly the three angles and displacements may be extracted from the updated transformation matrix of the compliant wrist $T_w$. Secondly these six differential displacements may also be calculated from the wrist mechanism Jacobian matrix $J_w$ and $\Delta \theta_w$ which are measured from in potentiometer readings in the sensor mechanism.

$$\Delta X_w = J_w \Delta \theta_w \quad (5)$$

Results for both methods are same, but we chose the first approach because of simplicity.

Joint rate control of the manipulator is defined as

$$\Delta X = J_m \Delta \theta_m \quad (6)$$

where $J_m$ and $\Delta \theta_m$ are the Jacobian matrix and joint differential change of the manipulator. Since compensation is desired in position control so that the manipulator moves in the opposite direction as the displacement, $\Delta X$ in (6) should be negative of $\Delta X_w$ in (5) if a complete compensation is desired. Therefore,

$$\Delta \theta_m = -J_m^{-1} \Delta X_w \quad (7)$$

More generally we may introduce a gain matrix $K_p$ so that the desired joint command vector $\theta_{des}$ becomes

$$\theta_{des} = \theta_{req} + \Delta \theta_m = \theta_{req} - J_m^{-1} K_p \Delta X_w \quad (8)$$

where $\theta_{req}$ is the desired joint angles, perhaps supplied by a trajectory generator function.
The experiment consisted of an incremental change of load in or around a certain direction. The six DOF wrist deflections and six DOF manipulator endpoint were recorded. Shown in Fig. 3 are the results as the load is applied around the x direction, with a gain \( K_p \) of 1.0, and the digital filter pole \( a \) set to 0.8. The upper curve is the observed wrist angle deflection around the x direction \( \theta_x \) and the lower curve is a recording of the manipulator endpoint response in the same direction. From these curves, we can see that the manipulator moves in the opposite direction but by the same amount as the wrist deflection so that the absolute endpoint location is maintained.

The experimental results indicate that the use of an active sensing of the compliant wrist makes it possible to retain the original static stiffness characteristics of the manipulator in spite of the presence of a passive compliance in the wrist.

We may analyze the system dynamic performance with a simple single degree of freedom system as shown in Fig. 4. The end-effector is represented as a mass \( m \) and the compliant wrist is represented as a spring with stiffness \( K_w \) and a viscous damping \( C_w \) which are attached to the manipulator end-point. The manipulator is assumed to be rigid. The end-effector motion is \( x_2 \) and the manipulator end-point motion is \( x_1 \). The compliant sensor records the difference of the motions \( \Delta x = x_2 - x_1 \). The controller uses the error information \( \Delta x \) with the end-point command \( x_{1e} \) together to drive the end-point motion \( x_1 \) so that the end-effector motion \( x_2 \) is not affected by the external force \( F_{ext} \) in position control while dominated by \( F_{ext} \) in force control.

Using the position control scheme discussed above to this simple system, the control strategy can be described by the block diagram in Fig. 5 where the feedback loop can be switched to a negative gain in position control or a positive gain in force control which we will discuss later. The transfer function from \( x_1 \) to \( x_2 \), \( G_1 \), and from the external force \( F_{ext} \) to \( x_2 \), \( G_2 \) can be obtained from the system in Fig. 4.

\[
G_1(s) = \frac{X_2(s)}{X_1(s)} = \frac{C_w s + K_w}{ms^2 + C_w s + K_w} \tag{9}
\]

\[
G_2(s) = \frac{X_2(s)}{F_{ext}(s)} = \frac{1}{ms^2 + C_w s + K_w} \tag{10}
\]

In position control, the system function can be derived,

\[
(1-K_p)G_2F_{ext} + G_1X_{1e} = (1-K_p+K_p G_1)X_2 \tag{11}
\]

To make a comparison, the system function for Fig. 4 without active feedback control may also be written as

\[
G_2F_{ext} + G_1X_{1e} = X_2 \tag{12}
\]

From (11), we may be clear that the effect of the external force is decreased by \( (1-K_p) \) times. If a unit gain \( K_p \) is used, the external force can be eliminated and the system is totally independent of the force \( F_{ext} \).

The system dynamic stiffness \( Z(s) \) can be derived from (11) if the second term is not considered.

\[
Z(s) = \frac{F_{ext}(s)}{X_2(s)} = ms^2 + \frac{C_w s}{1-K_p} + \frac{K_w}{1-K_p} \tag{13}
\]

We also can derive it for the case without feedback control (12).

\[
Z(s) = \frac{F_{ext}(s)}{X_2(s)} = ms^2 + C_w s + K_w \tag{14}
\]

Therefore, the virtual stiffness in steady state is increased by \( 1/(1-K_p) \) due to introduction of the active feedback and the entire system has a better positioning ability despite the presence of the passive compliance.

We may obtain the system characteristic equation from (11) for position control mode

\[
(1-K_p)ms^2 + C_w s + K_w = 0 \tag{15}
\]

Utilizing the feedback gain \( K_p < 1 \) in position control which ensures the system stability from (15). If a unit gain is used, which can provide complete compensation as shown in our paper [1], the system open-loop function \( G_1 \) (9) can present a gain margin less than zero.
4. FORCE CONTROL

In the force control case where the end-effector is partially constrained by the workpiece, we use the compliant wrist as a sensor to detect the force exerted in the end-effector. The sensed deflection is used to drive the manipulator in the same direction as the deflection of the wrist, i.e., the generalized force direction so that the apparent stiffness is decreased and a desired contact force is obtained.

We utilize a joint rate control scheme for the force control experiments. As discussed above, we can obtain the six components generalized displacement of the wrist $\Delta X_w$ from either the wrist Jacobian matrix or the transformation matrix.

The manipulator rate control scheme is given by

$$\Delta \theta_m = J_m^{-1} \Delta X$$  \hspace{1cm} (16)

and the displacement $\Delta X$ relates to the exerted force $F_w$ by the desired stiffness $K_d$

$$F_w = K_d \Delta X$$  \hspace{1cm} (17)

whereas measured displacement relates to the exerted force by

$$F_w = K_w \Delta X_w$$  \hspace{1cm} (18)

where $K_w$ is the actual stiffness of the wrist. Substituting yields

$$\Delta X = K_F \Delta X_w$$  \hspace{1cm} (19)

where $K_F$ is a dimensionless stiffness ratio

$$K_F = K_d^{-1} K_w$$  \hspace{1cm} (20)

Equation (17) becomes

$$\Delta \theta_m = J_m^{-1} K_F \Delta X_w$$  \hspace{1cm} (21)

The desired joint angles are thus

$$\theta_{des} = \theta_{curr} + \Delta \theta_m = \theta_{curr} + J_m^{-1} K_F \Delta X_w$$  \hspace{1cm} (22)

Comparing (8) with (22), we may note

(1) The equations are very similar, and both contain a dimensionless gain term $K_F$ or $K_F$ with the different sign in front. The former represents the gain by which we want to modify the displacement of the wrist according to how much amount of the deflection we expect to compensate in position control. The latter represents another gain matrix by which the stiffness of the end-effector relates to the effective stiffness of the system according to how much compliance is required in the force control task. If complete compensation in all directions is required in position control, the gain $K_F$ is identity matrix. If the desired compliance level is just the natural compliance, $K_w$, of the wrist, the gain matrix $K_F$ is again the identity matrix.

(2) Since in position control the end-effector is supposed to move in the opposite direction to the displacement we measured while it is to move in the same direction in force control, the differential displacement $\Delta \theta_m$ in Equation (8) is negative while positive in Equation (22).

(3) Provided the end-effector is in steady state and a constant deflection (i.e., constant force) exists in the compliant wrist, the manipulator should keep moving in the force control mode while it should stop if a constant compensation has been already responded to in position control mode. Therefore, the desired joint angles $\theta_{des}$ should be based on the specified joint angles $\theta_{req}$ in position control (8), but based on the current joint angles $\theta_{curr}$ in force control (22).

To check the force control mode, a constant force was applied suddenly, simulating the end-effector coming into contact with environment. Shown in Fig.6 are the results of the deflection of the wrist and the endpoint response of the manipulator, provided the desired force level is zero. At first, the force was applied and the wrist was deformed, and the manipulator end-point was moving in the same direction as the measured force. Then, as the manipulator reached the desired force level which is specified as zero, the manipulator moves from the surface, the wrist has no deflection and the manipulator stops.
In force control, we also can analyze the system as in position control case. The system block diagram is shown in Fig.5 where the feedback loop is switched to "force control". The system function can be derived as

\[(1+K_F)G_2F_{ext} + G_1X_{1e} = (1+K_F-K_F G_1)X_2\]  

(23)

Here we may see the feedback control makes the external force dominate the system. The dynamic stiffness is

\[Z(s) = \frac{F_{ext}(s)}{X_2(s)} = ms^2 + \frac{C_w s}{1+K_F} + \frac{K_w}{1+K_F}\]  

(24)

Comparing (24) to (14), the virtual stiffness of the system is decreased by \(1/(1+K_F)\) times. The higher gain is set, the softer the system becomes. The system characteristic equation is given as

\[(1+K_F)ms^2 + C_w s + K_w = 0\]  

(25)

which indicates that the system is always stable in force control. However, as the gain \(K_F\) is increased, the gain margin and phase margin are decreased and the relative stability is affected. On the other hand, the force gain selection is also dependent on the desired pole assignment of the system. Since the physical and desired stiffness in each direction or around each axis are different, the force gains were set differently in our experiment.

If the environmental compliance must be considered, the contact force can be modeled as spring and damper between the end-effector \(x_2\) and a rigid hypothetic surface \(x_3\) as shown in Fig.7. The stiffness and damping of the environment are represented as \(K_e\) and \(C_e\). The problem can be interpreted as that the compliance between \(x_2\) and \(x_3\) is controlled to a desired level. Hence, the force control problem is actually the problem of controlling the compliance between the end-effector and the environment. In our case, the problem can be specified so that the end-effector motion \(x_3\) is modified by adjusting the manipulator endpoint motion \(x_1\) in the presence of change in the environmental compliance (including surface geometry and material elasticity). In this way the compliance between \(x_2\) and \(x_3\) is maintained in a desired level. The control block diagram can be shown in Fig.8. Similarly as in the analysis above, we can derive the system transfer functions as

\[G_1 = \frac{C_w s + K_w}{ms^2 + (C_e + C_w)s + (K_e + K_w)}\]  

(26)

\[G_3 = \frac{C_e s + K_e}{ms^2 + (C_e + C_w)s + (K_e + K_w)}\]  

(27)

\[\frac{X_2}{X_3} = \frac{(1+K_F)G_3}{1+K_F - K_F G_1} = \frac{C_e s + K_e}{ms^2 + (C_e + C_w)s + (K_e + K_w)}\]  

(28)

Equation (28) and (27) indicate clearly that the active feedback control makes the system softer and the compliance of the contact can be controlled by adjusting the feedback gain \(K_F\).

5. HYBRID CONTROL

It is well known that every manipulator task can be broken down into elemental components that are defined by a particular set of contacting surfaces, a generalized surface can be defined in a constraint space having six degree of freedom, with position constraints along the normal to this surface and forces constraints along the tangents. These two type of constraints, force and position, partition the degree of freedom of possible end-effector motions into two orthogonal sets, that must be controlled according to different control strategies (8) and (22). Since the desired angles \(\theta_{des}\) is based on \(\theta_{sys}\) in position control (8), but based on \(\theta_{err}\) in force control (22), the hybrid control can not simply combine Equation (8) and (22) together. Here we present a hybrid control scheme for the robot system including a passive compliance as follows.

First, we partition \(\Delta X_w\) which is the Cartesian error measured from the wrist sensor into two sets \(\Delta X_{w}^f\) corresponding to the direction in which force control is required, and \(\Delta X_{w}^p\) in the remaining directions in which the position control is required. For example, if the force in the \(f\) direction and the torques around
the \( x \) and \( y \) directions are controlled and in the remaining directions are position controlled, we partition \( \Delta X_w \)

\[
\Delta X_w = \begin{bmatrix} \Delta x & \Delta y & \Delta z & \Delta \theta_x & \Delta \theta_y \end{bmatrix}^T
\]

into

\[
\Delta X_p = \begin{bmatrix} 0 & 0 & \Delta z & \Delta \theta_x & \Delta \theta_y \end{bmatrix}^T
\]

(30)

\[
\Delta X_F = \begin{bmatrix} \Delta x & \Delta y & 0 & 0 & \Delta \theta_z \end{bmatrix}^T
\]

(31)

Then, we multiply these vectors by the gain matrices \( K_p \) and \( K_F \) as in (8) and (22) to obtain the desired differential motion of the end-effector.

\[
\Delta X_p = K_p \Delta X_p^r
\]

(32)

\[
\Delta X_F = K_F \Delta X_F^r
\]

(33)

where \( \Delta X_p \) is compensated in position control mode and \( \Delta X_F \) is controlled in force control mode provided that the desired force level is zero.

From differential motions \( \Delta X_p \) and \( \Delta X_F \) we can form differential transform matrices \( T_{AX} \) and \( T_{AX_x} \) respectively. When force control is considered in Cartesian space, the desired motion of the end-effector is with respect to \( T_{\eta_j} \), where the superscript \( j \) refers to the time.

\[
T_{\eta_j} = T_{\eta_{j-1}} \ast T_{AX_x}
\]

(34)

Since we must consider the deflection of the end-effector in the presence of the passive compliance, the desired motion \( T_{\eta_j} \) has to be modified by the differential motion \( T_{AX_x} \) which represents the deflection of the compliant wrist in (32). Therefore, the required motion \( T_{\eta_j} \) to yield the desired motion \( T_{\eta_j} \) is

\[
T_{\eta_j} = T_{\eta_j} \ast T_{AX_x}
\]

(35)

Thus, the end-effector not only provides the desired compliance in the specified degrees of freedom, but also compensates the deflection of the compliant wrist simultaneously.

Similarly in joint space, we also can perform hybrid control. The joint rate control scheme from Equation (6) is

\[
\Delta \theta = J_m^{-1} \Delta X
\]

(36)

The desired joint angles of the end-effector must move in the same direction as the differential joint angles caused by \( \Delta X_F \), \( J_m^{-1} \Delta X_F \), based on the current desired joint angles as in Cartesian space control (34).

\[
(\theta_{\text{des}})_j = (\theta_{\text{des}})_{j-1} + J_m^{-1} \Delta X_F
\]

(37)

Also, the end-effector motion must be modified by the differential joint angles which represents the deflection of passive compliance \( \Delta X_p \) in (32), \( J_m^{-1} \Delta X_F \).

\[
(\theta_{\text{req}})_j = (\theta_{\text{des}})_j - J_m^{-1} \Delta X_p
\]

(38)

Equation (37) and (38) represents hybrid position and force control in joint space in correspondence with Equation (34) and (35) for Cartesian space. We chose the latter approach because of simplicity. An experiment based on this control scheme was performed. We specified that the force along the \( z \) axis and the torques around the \( x \) and \( y \) axes while in the remaining directions position control was performed. Force in an arbitrary direction was applied and the wrist deflection as well as the end-effector motion was recorded. The experimental curves in force control mode are identical with the curves shown in Fig.6 and those in position control mode are identical with the curves shown in Fig.3. The results were satisfactory and demonstrated the control scheme is stable.
6. SURFACE TRACKING

A surface tracking experiment was performed as an application of the hybrid control strategy of the robotic system with a compliant wrist. The surface is sinusoid curved as shown in Fig. 2. The robot does not have any information about the surface. The end-effector trajectory is modified by the sensed contact forces. When the robot comes down with a certain velocity and makes contact with the surface, the controller is automatically switched from a full position control (i.e., six DOF position control) to a hybrid control scheme. The force normal to the surface is set to be controlled, while in other directions position is controlled. A desired contact forces, force feedback gain, position gain, and other parameters can be specified in the terminal.

A detail analysis for the system performance with a digital filter will be discussed in next section. Here, we present some experimental results, in order to investigate the effect of different conditions on the system performance.

(1) Force gain $K_F$ must be selected carefully. As we expected, the less stiff the support of the contacted surface (i.e., environment) is, or the more compliant the end-effector is, the larger gain can be utilized. This is the reason why we introduce some passive compliance in the wrist. When we provide a passive compliance device in the end-effector, a much larger force gain can be used than that without it. However, the force gain has to be set according to the compliance we provided and environment behavior. As a too large force gain is set, tracking will also get unstable. Fig. 9 are the curves recorded for two translation motions $y$ and $z$ in the end effector as the end-effector is tracking a sinusoid surface. We may see that in the direction normal to the surface, $z$ direction, the record is a sinusoid curve. Since the tracking is accomplished by moving along $y$ direction in a certain velocity, the curve of $y$ direction motion is a line with a certain slope. The upper curves is for the case that force gain is 0.2, while the lower one is that the force gain is 0.6. It is clear that in the latter case, tracking can not copy the surface behavior perfectly because of a too large gain.

(2) Transition as the robot makes contact with the surface or breaks contact from the surface can be accommodated by the method proposed above. The passive compliance provides a mechanism to absorb a kinetic energy of impulsive force as the robot suddenly stops on the workpiece from a significant velocity. Without the passive compliance, the force or velocity may be discontinuous in this moment. From our experiment, we used different velocities before the contact is made. In all cases, the contact is very smooth and no shock was observed. In Fig. 9, we also can find that the robot initially moves down with a certain velocity and then make contact with the environment smoothly as in the area we circled.

(3) A reasonably large contact force specified is desirable. The experiment shows that the larger contact force we specified, the more smooth the contact becomes. It is also needed to know that a large contact force specified results in a large deflection of the compliant wrist, thus the contact force is limited since the deflection of the wrist has to be within a permitted range.

7. SYSTEM PERFORMANCE

There is a need to use a digital filter in the close-loop of the system because of mechanical backlash in the potentiometers and electronic noise. A first order digital filter with a unity-gain is utilized in the closed-loop system. The digital filter in the $Z$ transformation form is

$$F(z) = \frac{Y(z)}{U(z)} = \frac{1-a}{z-a}$$

(39)

The low pass corner frequency is controlled by the pole of the digital filter, $a$

$$a = e^{-\omega_0 T}$$

(40)

where $T$ is the sample interval, in our case 28 ms.

The system performance is dominated by the digital filter. The system block diagram without the filter in Fig.5 can be modified as the diagram in Fig 10, where $F$ denotes the digital filter. It has been known that the discrete variable $Z$ relates the Laplace variables $s$ by

$$z = \frac{\frac{2}{T} + s}{\frac{2}{T} - s}$$

(41)
Thus, substituting yields that $F(z)$ in Equation (39) becomes the Laplace transformation form $F(s)$

$$F(s) = \frac{(1-a)/(1+a)}{s+2/(1-a)/(1+a)}$$

Let

$$a_0 = \frac{(1-a)/(1+a)}, \quad f_0 = \frac{2}{T}$$

thus

$$F(s) = \frac{a_0f_0-a_0s}{s+a_0f_0}$$

As the pole of the digital filter $a$ becomes zero, $a_0$ goes to unity, and vice versa. The system transfer function from $X_1$ to $X_2$ in position control mode (11) is changed as

$$G_1 = \frac{C_w s + K_w}{1 - K_P F + K_P G_1 F}$$

$$= \frac{C_w s + K_w}{(ms^2 + C_w s + K_w) - K_P ms^2 F}$$

Thus the characteristic function is

$$(m + K_P a_0 m) s^3 + (ma_0 f_0 - K_P a_0 m f_0 + C_w) s^2 + (C_w a_0 f_0 + K_w) s + K_w a_0 f_0 = 0$$

The stability conditions for position control mode can be derived as

$$ma_0 f_0 - K_P ma_0 f_0 + C_w > 0$$

$$[ma_0 f_0 (1 - K_P) + C_w] (C_w a_0 f_0 + K_w) > K_w a_0 f_0 (1 + K_P a_0) m$$

From the stability condition, we may see:

(i) Passive damper is of significance. If the damping $C_w = 0$, the second condition and first condition yield respectively

$$-1 > a_0 \quad \text{and} \quad K_p < 1$$

The former is contradictory and the latter means that a full compensation is impossible.

(ii) The pole of the digital filter $a$ should be selected as close to unity as possible. As $a$ goes to unity, $a_0$ become zero, and the first and second conditions yield

$$C_w > 0, \quad \text{and} \quad C_w K_w > 0$$

which are always true, while as $a_0$ goes to unity, the conditions are not always satisfied.

(iii) The gain $K_p$ dominates the system. As $K_p$ is selected close to unity, both conditions are getting critical, especially when the less damper is used.

Since the mass of the end-effector in our experiment is small and parameters $K_P$ and $a_0$ are both less than 1, we neglect the third order term in (46), thus the nature frequency of the system $\omega_n$ can be obtained

$$\omega_n = \frac{K_w a_0 f_0}{(1 - K_P a_0 m f_0 + C_w)^{1/2}}$$

Compared to the nature frequency of the original system $\omega_n0$

$$\omega_n0 = \frac{K_w}{m}$$

Equation (49) can be written in the form of

$$\omega_n = P_0 \omega_n0$$

where the coefficient $P_0$ is the ratio of $\omega_n/\omega_n0$
For most cases $P_0$ is less than unity. As $C_w$ is decreased and $K_F$, $a_0$, $m$ are increased, $P_0$ is increased and goes to near unity, and thus an oscillation may occur.

The curve in Fig. 11, can give us a clear picture of the discussion above, where $a_0^*$ is the variable $a_0$ corresponding to the unity $P_0$. If $K_F = 0.8$, $C_w = 10$ kg/s, $f_0 = 100$ $1/s$, $m = 0.1$ kg, the critical value $a_0^* = 0.8$. The permitted region for $a_0$ is $0 \leq a_0 < a_1$, where $a_1$ should be less than critical value $a_0^*$. In other words, $a_0$ has to be set near zero, which means the pole of the digital filter has to be close to 1, so that the system oscillation is avoided. For example, if $a_1 = 1/3$, the pole is allowed to be selected within $1 > a > 1/2$.

From here, we conclude that in position control mode the system bandwidth is limited by the nature frequency of passive compliance $\omega_n$. In other words, the compliant wrist device cannot be made too soft.

The surface tracking experiment demonstrated that for position control model if the pole of the digital filter is lowered to a certain value, an oscillation may occur. Using the same parameters except for the pole of the filter in position control mode, the experiment results are shown in Fig. 12. The upper curve is for the case in which the pole is 0.95, while the lower curve is for the case in which the pole is 0.6 where an oscillation is evident.

We also can investigate the system performance for force control mode with the similar analysis as shown above, provided a zero contact force is desired. For simplicity, we only give a stability condition.

$$K_F \ a_0 < 1$$

It is clear that if the force gain $K_F$ and the pole of digital filter are both selected less than unity, the system will not be unstable. The nature frequency in force control mode is

$$\omega_n = (\frac{K_w a_0^*}{m a_0^* (1 + K_F) + C_w})^{1/2} = Q_0 \omega_n$$

where

$$Q_0 = (\frac{1}{(1 + K_F)^{1/2} - C_w})^{1/2}$$

Compared $P_0$ in position control (51) with $Q_0$ in force control (53). We may see that there is no resonance in force control since the denominator in (53) always larger than 1 and $Q_0$ is never reach to unity. This is also verified by the experiment.

8. CONCLUSIONS

(1) The compliant wrist performs successfully both in passive compliance and active sensing. Using such a device makes it possible to provide the system with a flexibility which simplifies contact force control and the transient state control, and compensates the end-effector deflection due to the external forces.

(2) The applicability of the position and force control strategies is shown by experiment. In position control, the manipulator moves in the opposite direction to the deflection of the wrist so that the apparent stiffness of the end-effector is increased. In force control, the manipulator is driven in the same direction as the sensed force so as to decrease the stiffness.

(3) Hybrid position and force control for the robot system including passive compliance is possible and the control scheme's feasibility is demonstrated by the experimental results. The scheme can also be utilized as a free joint control scheme, as well as a conventional hybrid control without a passive compliance.

(4) The digit filter dynamics dominate the systems response. In position control, a large pole of the filter results in a slow response. To lower the pole to an extent, an oscillation may occur. The lower critical value of the pole is limited by the nature frequency of passive compliance. In other words, a trade off has to be made between a fast response and a large compliance of the wrist device.
(5) Force gain should be selected carefully according to the stiffness of the passive compliance device, the mass of the end-effector, and the desired compliance of the system. The different force gains are necessary for each direction and around each axis. The less stiff the passive compliance device or environment, the larger force gain can be used.

9. REFERENCES


Fig. 1 Mechanical structure of the compliant wrist

Fig. 2 Experiment with the compliant wrist on PUMA 560

Fig. 3 Deflection reading of the wrist and manipulator end-point response in position compensation
Fig. 4 Simple model of the wrist system

Fig. 5 Block diagram of the active feedback control

Fig. 6 Deflection reading of the wrist and manipulator end-point response in force control

Fig. 7 Simple model of the wrist system and environment

Fig. 8 Block diagram of the compliance control

Fig. 9 Sinusoid
Fig. 10 Block diagram of the active feedback control with a digital filter

Fig. 11 Ratio $\omega_n/\omega_{n0}$ is a function of the filter pole $\alpha$. $\alpha_0 = (1-\alpha)/(1+\alpha)$. 

Sinusoid surface tracking reading of the end-effector (in y and z direction)
Fig. 12  Sinusoid surface tracking reading of the end-effector (in y and z direction)
A.7 Decoupling Mechanisms for Robot Manipulators
DECOUPLING COMPLIANCE MECHANISMS OF ROBOT MANIPULATORS

Yangsheng Xu
Richard P. Paul
General Robotics Active Sensory Perception Laboratory
School of Engineering and Applied Science
University of Pennsylvania, Philadelphia, PA 19104-6389, USA

ABSTRACT

For the proper correction of position and angular errors with a passive compliance, the position measurements, force sensing, and the hybrid control for active compliance are simplified if the compliance of the end-effector is decoupled. To find a geometric configuration corresponding to such a decoupled compliance of the end-effector is an unsolved problem. The paper presents a new concept of an orthogonal Jacobian mechanism in which the compliance matrix is diagonal. The mechanism configuration can be found using Plucker coordinates. The process is simple and illustrated by several examples. The nonlinear equations is derived and a program is designed to generate such mechanism configurations. The mechanism can be selected for a specific compliance of each direction in the task space. Dynamics of the mechanism is also addressed and the dynamic performance can be similarly specified easily by certain configurations and geometric parameters. When the end-effector is in motion, avoidance of the intersection is one of the key problems in implementation of the parallel manipulator. We present a method to determine whether the links are intersected for a given trajectory in off-line planning. The method can also be used for all parallel mechanisms whose configurations are specified.

1. INTRODUCTION

Passive compliance of the end-effector of the manipulator allows external constraints to modify the trajectory of the end-effector. Passive compliance mechanisms have an adaptation capability which permit self-correction to accommodate to the geometric errors and uncertainties in assembly tasks.

This material is based on work supported by the National Science Foundation under Grant No. DMC-8512838. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.
Part of this material was presented in the International Conference on Advances in Robot Kinematics, Ljubljania, Yugoslavia, Sept. 1988
The passive compliance device known as an RCC is designed for this purpose[7]. The device, located at the robot wrist, has a number of advantages: automatic assembly could be facilitated in more applications, the positioning error or dimensional tolerances can be relaxed, the possible high forces generated in assembly operation can be reduced, and expensive electronics normally required for precision positioning eliminated. The passive compliance device instrumented by a force or position sensor which measures the deflections of the compliance structure or the forces exerted exhibits the advantages of both passive compliance and active sensing, thereby the position and force control of the end-effector become possible as discussed in our earlier papers [1][2].

Similarity in geometric configuration and task can be found in another kind of mechanism, the parallel manipulator or Steward plateform mechanism. It consists of two plates connected by six articulated links. The end-effector is manipulated by six prismatic joints in six links. The mechanism was designed to provide compliance in assembly workstations primarily and then developed as a small motion manipulator with passive compliance[3][4][5][6].

It is well known that the forces applied at the compliance device must be decoupled so that the force only causes a pure translation and the torque produces only pure rotation. The RCC is the device that the lateral forces and rotational torques are partially decoupled.

The decoupling problem becomes more important if the compliance device is used as a force sensor. If the deflections of the device are coupled, the generalized forces applied are coupled and cannot be obtained directly from the deflection sensing. As both position and contact force is controlled by a hybrid system, the controller incorporates two sets of closed loops for force and position control. The force control is driven by the motion of the end-effector, and becomes difficult if the force and displacement are coupled.

The decoupling problem can be interpreted as orthogonalizing the Jacobian matrix of the mechanism. This paper presents a method to find mechanism configurations in which the Jacobian matrix is orthogonal. Because the inverse Jacobian can be replaced by its transpose, the joint rate control scheme is not complicated by singularities of the Jacobian. Therefore, a diagonal compliance mechanism is attractive not only for the passive compliance but also for active control.

The problem has been addressed by some papers. Whitney [7] indicated significance of the decoupling problem and analyzed the RCC device mechanics based on this principle. The compliance center was calculated by choosing proper parameters. However, the RCC is not fully decoupled and the results are not suitable for parallel manipulators. Loncaric [8] presented a method to normalize the compliance matrix by a particular choice of the coordinates frame using the Lie Group approach. But, the choice of the frame is not clear physically and is difficult to realize. Merlet [6] analyzed some mechanisms which can be used as parallel manipulators, and one of mechanisms was mentioned as a diagonal compliance matrix mechanism. A systematic way to find and analyze such orthogonal Jacobian mechanisms is needed.
This paper presents the relation between the diagonalizing compliance matrix of the end-effector and orthogonalizing the Jacobian matrix of the manipulator. The Plucker Line Coordinates method is applied to find the mechanisms in which Jacobian matrix is orthogonal. The geometric configuration of the mechanism is very clear physically and the process is illustrated by some examples. The compliance and dynamics performance can be specified easily by choosing mechanism configuration, or modifying some geometric parameters for a given mechanism. The nonlinear equations associated with the parameters of the mechanisms are derived and a program to generate such mechanisms is designed using the Damped Newton algorithm. Since the links may intersect each other as the top plate is in motion, we present a method to determine the perpendicular distance between the links, thereby to identify intersection for a given trajectory in off-line planning. The method we proposed is not only suitable to the decoupling compliance mechanism, but also can be used in all kinds of the parallel mechanism.

2. ORTHOGONAL JACOBIAN MATRIX AND COMPLIANCE

We now consider a parallel manipulator as described in [4][5][6] which has two platforms connected by six movable links as shown in Fig.1. The top plate, or called the mobile plate, is controlled by adjusting six prismatic joint motions. It is known that there is relation between the forces $f$ in each link along its axes and the generalized forces $F$ applied to the center of the top plate.

$$f = J^T F$$

where $J$ is Jacobian matrix of the mechanism. If the forces along the link are proportional to the deflections of the length $l$, then

$$f = K \Delta l$$

where $K$ is the stiffness matrix, $K = \text{diag}(k_1, k_2, \ldots, k_6)$, $k_i$ is stiffness of each link. The deformation of the top plate $\Delta X$ is related to the deflection of each link by

$$\Delta l = J^{-1} \Delta X$$

Combining those equations and assuming each link has a same stiffness $k$, we obtain

$$F = k \ (J^T)^* J^{-1} \Delta X$$

or

$$\Delta X = k^{-1} \ J \ J^T F$$

We define the compliance matrix $C$ and the stiffness matrix $K$ as

$$C = k^{-1} \ J \ J^T$$

(4)

$$K = k \ (J^T)^* J^{-1}$$

(5)

Namely,

$$F = K \Delta X$$

(6)
Hence, the problem becomes clear. In order to decouple compliance or stiffness matrix, the Jacobian matrix must be orthogonal.

\[ \Delta X = C F \]  
(7)

where

\[ J J^T = D \]  
(8)

Some interesting facts associated with the orthogonal Jacobian mechanism may be noted:

(1) Since the Jacobian matrix is orthogonal, the inverse Jacobian can always be replaced by its transpose. From (8), we have

\[ J^{-1} = J^T D^{-1} \]  
(9)

\( D^{-1} \) is easy to obtain because \( D \) is diagonal. Actually, as shown in the later sections we calculate \( D^{-1} \) first instead of \( D \). Therefore, the inverse Jacobian computational cost and excessive joint rates near a singularity can be avoided.

(2) The end-effector deflections of the manipulator can be easily predicted since the compliance matrix is diagonal. The maximum deflection and minimum deflection occur in the weakest stiffness and the strongest stiffness direction. Therefore, the desired stiffness of each direction in Cartesian space which is dependent upon the tasks of the robot manipulator can be specified by selecting the geometric parameters and configuration of the mechanism.

(3) The manipulability measure is an ability of the robot in positioning and orienting the end-effector. It can be defined by [11]

\[ w = \sqrt{\text{det}(J J^T)} \]  
(10)

for determining the best posture of various types of manipulators and articulated robot fingers. If the Jacobian matrix is orthogonal, the manipulability can be evaluated by multiplication of each diagonal element \( d_i \) as in Equation 8, thereby selecting the mechanism configuration according to manipulability becomes easy in the design state.

3. PLUCKER LINE COORDINATES AND MECHANISM CHOICE

In this section, we are going to find the mechanism whose Jacobian is orthogonal by the method of Plucker Line Coordinates.
The line in space can be represented by so called Plucker line coordinates [9]. As shown in Fig.2, two points \( M_1 \) and \( M_2 \) can be formed by a vector \( S = M_1 M_2 \). Let us construct another three dimensional vector \( M \) by

\[
M = OM_1 \times S = OM_1 \times OM_2 = S \times M_2 O
\]

where

\[
S = M_1 M_2
\]

We combine these vectors to form a six dimensional vector which is the vector \( U \) of the Plucker coordinates of this line \( M_1 M_2 \).

\[
U = [S_x, S_y, S_z, M_x, M_y, M_z]^T
\]

\( U \) can be normalized as

\[
U' = \frac{U}{|S|} = [S'_x, S'_y, S'_z, M'_x, M'_y, M'_z]^T
\] (11)

Therefore, first three components of the vector \( U' \) are the components of a unit vector \( S \). The last three components are orthogonal with the first three as

\[
S'_x M'_x + S'_y M'_y + S'_z M'_z = 0
\] (12)

and can be given by

\[
OM \times S'
\]

where \( M \) is any point of the line.
The mutual moment of two line segments \( S_1 \) and \( S_2 \) is
\[
S_{x1}M_{x1} + S_{y1}M_{y1} + S_{z1}M_{z1} + S_{x2}M_{x2} + S_{y2}M_{y2} + S_{z2}M_{z2} = d \sin \alpha
\]  
where \( d \) is the perpendicular distance between the lines, and \( \alpha \) is the angle between them. When the mutual moment is zero the lines either intersect or are parallel.

A matrix \( P \) can be formed as
\[
P = (U'_1, U'_2, \ldots, U'_n)
\]  
Now, if we consider the vectors \( S'_i \) as unit force along the axes of the link, \( M'_i \) as the moment around the reference point generated by the force \( S'_i \). Suppose the external generalized force vector \( F \) including the forces and torques about the origin \( O \) acting on the top plate, using the notation (14), we can obtain
\[
P f = F
\]  
Comparing (15) with the Equation 1, we see that Plucker Coordinate vector matrix has a relation with the Jacobian matrix of the device.
\[
P^{-1} = J^T
\]  
or,
\[
J = (P^T)^{-1}
\]
Both (16) and (17) are useful to find the required mechanisms. The orthogonal Jacobian matrix condition (8) or (9) can be satisfied if
\[
P P^T = \Lambda
\]  
where \( \Lambda \) is a diagonal matrix and is an inverse of \( D \) in (8). From (18), we can find a mechanism whose Jacobian matrix is orthogonal by choosing proper Plucker line coordinates matrix \( P \) because the Equation 18 only involves the transpose, and the configuration is clear physically.

A detailed discussion to find such a mechanism will be addressed in the next section. Here we try to show an alternative way to find the configuration by using the symmetric structure without numerical calculation of the nonlinear equations.

An example is shown in Fig. 3, \( S_i \) is the line vector corresponding to line \( i \), \( B_i \) is the distance vector from the origin to the vector \( S_i \), \( M_i \) is the moment vector corresponding to \( S_i \) and \( B_i \). The link \( i \) is denoted by two end points \((i, j)\), as \((1,1)\) for the first link and etc.
At first, we look for six vectors \( S_i \) which will be satisfied to the orthogonality condition (18). Namely,

\[
\sum_{i=1}^{6} S_x S_x = 0 \\
\sum_{i=1}^{6} S_y S_y = 0 \\
\sum_{i=1}^{6} S_z S_z = 0
\]

To facilitate the process, we partition the above equation into two sets of three

\[
\sum_{i=1}^{3} S_x S_x = 0 \\
\sum_{i=1}^{3} S_y S_y = 0 \\
\sum_{i=1}^{3} S_z S_z = 0
\]

From here, it is not difficult to find a number of sets of the vectors which are satisfied the above equations. As an example we list one set of the vectors.

\[
S_1 = (0, a, h) \\
S_2 = (\frac{\sqrt{3}}{2} a, -\frac{a}{2}, h) \\
S_3 = (-\frac{\sqrt{3}}{2} a, -\frac{a}{2}, h)
\]
Now we need to choose the set of the vectors for which the vectors \( B_i \), and thereby \( M_i \) satisfy the orthogonal condition (8). We know if \( S_i \) and the reference point \( O \) are determined, then \( B_i \) and thereby \( M_i \) are determined. Similarly, we partioned the first three sets and the last three sets of equations and let

\[
\sum_{i=1}^{3} M_i x_i y_i = 0 \quad \sum_{i=1}^{3} M_i x_i z_i = 0 \\
\sum_{i=1}^{3} M_i y_i x_i = 0 \quad \sum_{i=1}^{3} M_i y_i z_i = 0 \\
\sum_{i=1}^{3} M_i z_i x_i = 0 \quad \sum_{i=1}^{3} M_i z_i y_i = 0
\]

As for the vectors \( S_i \) listed above, we can find their \( B_i \) and \( M_i \).

\[
\begin{align*}
B_1 &= (\frac{\sqrt{3}}{2}a, -\frac{a}{2}, 0) \\
B_2 &= (\frac{\sqrt{3}}{2}a, 0, 0) \\
B_3 &= (-\frac{\sqrt{3}}{6}a, -\frac{a}{2}, 0) \\
B_4 &= (-\frac{\sqrt{3}}{6}a, -\frac{a}{2}, 0) \\
B_5 &= (-\frac{\sqrt{3}}{6}a, 0, 0) \\
B_6 &= (\frac{\sqrt{3}}{6}a, \frac{a}{2}, 0) \\
M_1 &= (\frac{1}{2}ah, \frac{\sqrt{3}}{6}ah, -\frac{\sqrt{3}}{6}a^2) \\
M_2 &= (0, -\frac{\sqrt{3}}{3}ah, -\frac{\sqrt{3}}{6}a^2) \\
M_3 &= (-\frac{1}{2}ah, \frac{\sqrt{3}}{6}ah, -\frac{\sqrt{3}}{6}a^2) \\
M_4 &= (-\frac{1}{2}ah, \frac{\sqrt{3}}{6}ah, \frac{\sqrt{3}}{6}a^2) \\
M_5 &= (0, \frac{\sqrt{3}}{3}ah, \frac{\sqrt{3}}{6}a^2) \\
M_6 &= (\frac{1}{2}ah, \frac{\sqrt{3}}{6}ah, \frac{\sqrt{3}}{6}a^2)
\end{align*}
\]

As we find such a set of vectors \( M_i \), we verify whether the mutual orthogonality of \( S_i \) and \( M_i \) (18) holds.

\[
(S_i \cdot M_i) = \sum_{i=1}^{6} S_i x_i M_i = 0
\]
so as

\[(S_x, M_x) = (S_y, M_y) = (S_z, M_z) = (S_y, M_y) = (S_z, M_z) = 0\]

If the orthogonal condition holds, the mechanism configuration is found. If not, we have to go back to the second step to choose another set of vectors \(B_i\) and repeat the process until we find a solution. The vectors we listed satisfy the orthogonality condition.

\[P = (U'_1, U'_2, \ldots, U'_6)\]

\[PP^T = \Lambda\]

and

\[\Lambda = diag (\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6)\]

\[= diag (3a^2/\rho, 3a^2/\rho, 6h^2/\rho, a^2h^2/\rho, a^2h^2/\rho, 1/2a^4/\rho)\]

where

\[\rho = a^2+h^2 = |S_i|^2\]

Two other mechanisms are found as shown in Fig.4 and Fig.5.

---

Fig. 5 Orthogonal Jacobian Mechanism

---
4. PROPERTIES OF THE DECOUPLING COMPLIANCE MECHANISMS

The vector of the Plucker line coordinates $U_i$ has six variables but not all of them are independent if the reference point is given. From now, we suppose the reference point is in the origin of the coordinates. The vector $S_i$ has three variables. But if the length of the vector is given, only two variables are independent. The possible location of the end of the vector is in a spherical surface whose radius is the length of the link and so has the dimension two. If we also give the height of the device (i.e., the $z$ coordinate), only one variable is free and the possible location of the end of the vector lies on a curve produced by cutting the spherical surface by a plane $z = h$. Therefore, if the location of the vector $S_i$ is known (i.e., $B_i$ is given), only one variable is needed to determine the vector $S_i$ in our particular problem.

Next, since $B_i$ could be any vector from the origin to a given vector, without losing any generality we assumed the vector $B_i$ is always in the X-Y plane, thereby two variables can identify $B_i$. Since
\( \mathbf{M}_i = \mathbf{B}_i \times \mathbf{S}_i \), if \( \mathbf{S}_i \) is determined, \( \mathbf{M}_i \) is only dependent on \( \mathbf{B}_i \). From here, it is clear that three independent variables are sufficient to identify the vector \( \mathbf{U}_i \). Therefore, we need only to find 18 variables to determine six vectors \( \mathbf{U}_i \), i.e., a full matrix \( \mathbf{P} \).

The orthogonality condition of the Jacobian matrix (18) includes 15 equations for 15 vanished non-diagonal elements. We will show later that there is only three independent elements for six diagonal elements. Therefore, if the three variables for the diagonal elements are given, 18 variables can be solved by 18 equations. If the diagonal elements are not given but one vector of \( \mathbf{S}_i \) is known, 15 variables can also be solved by 15 equations. The program to do this is shown.

The vector \( \mathbf{S}_i \) is located at \( (X_i, Y_i) \) in X-Y plane. Therefore, the vector \( \mathbf{B}_i \) is
\[
\mathbf{B}_i = (X_i, Y_i, 0)^T
\]
and the constrains condition of \( \mathbf{S}_i \) is
\[
\begin{align*}
x^2 + y^2 + z^2 &= \rho^2 \\
z &= h
\end{align*}
\]
If the spherical coordinates are used,
\[
\begin{align*}
x &= \rho \cos \phi \cos \theta = r \cos \theta \\
y &= \rho \cos \phi \sin \theta = r \sin \theta \\
z &= \rho \sin \phi
\end{align*}
\]
where \( \rho = \rho \cos \phi \) and \( \phi = \sin^{-1}(h/\rho) \) are known, so that only variable \( \theta \) is free to determine \( \mathbf{S}_i \).

Suppose six links have the same length \( \rho \) and the device height is \( h \), then
\[
\mathbf{S}_i = (\rho \cos \theta, \rho \sin \theta, h)^T
\]
thereby,
\[
\mathbf{M}_i = (h Y_i, -h X_i, r X_i \sin \theta, -r Y_i \cos \theta)^T
\]
\[
\mathbf{U}_i = (\rho \cos \theta, \rho \sin \theta, h, h Y_i, -h X_i, r X_i \sin \theta, -r Y_i \cos \theta)^T
\]
For the X-Y-Z coordinates expression,
\[
\mathbf{U}_i = (x_i, \pm(\rho^2 - x_i^2 - h^2)^{\frac{1}{2}}, h, h Y_i, -h X_i, \pm x_i (\rho^2 - x_i^2 - h^2)^{\frac{1}{2}}, Y_i)^T
\]
The Cartesian space unit compliance (i.e, the compliance as each link has a unit stiffness) of the device corresponding to (26) are
\[
d_1 = \sum_{i=1}^{6} x_i^2
\]
\[
d_2 = \sum_{i=1}^{6} (\rho^2 - x_i^2 - h^2) = 6(\rho^2 - h^2) - d_1
\]
\[
d_3 = 6h^2
\]
We may notice:

1. Two lateral compliance $d_1$ and $d_2$ are related and only one is free. If it is desired to have the same stiffness, i.e., $d_1 = d_2$, then
   \[ \sum_{i=1}^{6} x_i^2 = 3(p^2-h^2) \]

2. Since the compliance is always positive, $d_2$ must be positive, $d_2 > 0$, then
   \[ \sum_{i=1}^{6} x_i^2 < 6(p^2-h^2) \]

3. Whenever $h$ is determined, $d_3$ is determined which is equal to $6h^2$. The higher the device, the more compliant its axial stiffness becomes. However, the relative ratio of the stiffness could be arranged by the length and different configuration.

4. If the axial compliance is equal to the lateral compliance, $d_3 = d_2 = d_1$, then
   \[ \sum_{i=1}^{6} x_i^2 = d_1 = d_2 = d_3 = 2p^2 \]

5. Usually the large axial stiffness and the symmetric lateral stiffness are desired, i.e., $d_3 \leq d_1 = d_2$, In this case, the height of the device must be satisfied the following condition.
   \[ h^2 \leq p^2/3 \]

6. From $d_1$ to $d_6$ expressions, we know if eighteen variables $x_i, y_i, X_i, Y_i$ are given, $d_i$ are all determined. These eighteen variables can be solved by the orthogonality condition of 15 equations as a function of any three free variables. Now, if we add one equation from $d_1$ or $d_2$, and two equations from $d_4, d_5,$ or $d_6,$ normally we can solve the problem. However, if the eighteen variables are identified, we always can determine six $d_i$ as we mentioned before. Therefore, we conclude there are only three independent variables for six stiffness in each direction.

The program was designed to generate the configuration of the mechanism using the spherical coordinate expression (25) and the orthogonality condition (8). The Damped Newton algorithm is applied to
solve the nonlinear equations. The computation is rather efficient.

5. PARAMETERS AND DYNAMICS SPECIFICATION

First, if we assume each link has the same stiffness, the compliance of the end-effector is dependent on the configuration and geometric parameters. For a given mechanism configuration, the compliance is only dependent on the geometric parameters. Taking the mechanism shown as in Fig.4 for instance, we have

$$C = \frac{1}{kA} = \text{diag} \left(3a^2/kp, 3a^2/kp, 24h^2/kp, a^2h^2/kp, a^2h^2/kp, 1/2a^4/kp \right)$$

(27)

The lateral, axial, bending and torsional compliance are function of the distance between two links in the base plate and the height of the link. As the mechanism is chosen, the geometric parameters can obtained by specific stiffness desired in each direction. For example, the lateral and axial compliance for a unit \(k\) and unit height \(h\) can be evaluated as shown in Fig.6, while the bending and torsional compliance for a unit distance \(a\) is shown in Fig.7. From the curves, we may know how to modify \(a\) and \(h\) in order to obtain a desired compliance in each direction.

Further, if the sensitivities of the compliance to the geometric parameters are evaluated, a more precise adjustment can be made. The derivative of the stiffness in each direction to the distance \(a\) are obtained from Equation 27 as following.

$$\frac{dc_1}{da} = \frac{6ah^2}{(a^2+h^2)k}$$

$$\frac{dc_2}{da} = \frac{dc_1}{da}$$

$$\frac{dc_3}{da} = -\frac{24ah^2}{(a^2+h^2)k}$$

$$\frac{dc_4}{da} = \frac{2ah^4}{(a^2+h^2)k}$$

$$\frac{dc_5}{da} = \frac{dc_4}{da}$$

$$\frac{dc_6}{da} = \frac{(2h^2-3a^2)h^3}{(a^2+h^2)k}$$
We also can obtain the sensitivities with respect to the height \( h \). From the sensitivity, we can understand how to optimize the mechanism parameters in the sense of the desired compliance.

Dynamics of the mechanism can be derived if the mass of the link is neglected. The inertia matrix is

\[
M = \text{diag} (m, m, m, I_x, I_y, I_z)
\]

The inertia forces are

\[
F_I = M \Delta \ddot{X}
\]

The spring reaction forces are

\[
F_s = k \Lambda \Delta X
\]

where

\[
\Lambda = \text{diag} (\mu_1, \mu_2, \mu_3, \mu_4, \mu_5, \mu_6)
\]

The damping forces can be given if the linear viscous properties is assumed.

\[
f_d = d \Delta l
\]

\[
F_d = d \Lambda \Delta \ddot{X}
\]

where \( d \) is viscous damping coefficient. Combining above three terms of the forces, the motion equation of the Laplace form is

\[
(s^2M + ds\Lambda + k\Lambda)\Delta X = 0
\]  

(28)

From (28), the motion equation is decoupled as a single degree of freedom system. Thus, the dynamics performance can be specified by different mechanism configurations and geometric parameters. The natural frequencies and the damping ratio of the mechanism in Fig.4 are:

\[
\omega_x = \omega_y = \left(\frac{k_0}{3a^{-1}m}\right)^{\frac{1}{2}}
\]

\[
\omega_z = \left(\frac{k_0}{2Ah^2m}\right)^{\frac{1}{2}}
\]

\[
\omega_{x_1} = \left(\frac{k_0}{a^2h^2I_x}\right)^{\frac{1}{2}}
\]

\[
\omega_{x_2} = \left(\frac{k_0}{a^2h^2I_z}\right)^{\frac{1}{2}}
\]

\[
\omega_{y_1} = \left(\frac{2k_0}{a^2I_y}\right)^{\frac{1}{2}}
\]

\[
\zeta_x = \zeta_y = \frac{d}{2m\omega_x}
\]

\[
\zeta_z = \frac{d}{2m\omega_z}
\]

\[
\zeta_{x_1} = \frac{d}{2I_x\omega_{x_1}}
\]
\[ \zeta_{d} = \frac{d}{2I_{y} \omega_0} \]

\[ \zeta_{r} = \frac{d}{2I_{z} \omega_0} \]

6. AVOIDANCE OF INTERSECTION

Another problem that many of researchers are concerned with is the intersection of the links when the top plate of the mechanism is in motion. The intersection problem is not only a problem for the decoupling compliance mechanism, but also for all parallel mechanism whose configuration is similar as that shown in Fig. 1. Therefore intersection has been considered as one of the key problems in the design of parallel mechanisms as the singularity problem is in serial mechanisms. In this section, we will present a method to determine the intersection of the links for a given trajectory in off-line planning.

We define the intersection as the case when the minimum perpendicular distance between two links axes is less than a given small value \( \delta \), in order to account for the geometric size of the links in the implementation. In the special case when the value is zero, the intersection is ideal in the geometric sense.

We supposed that base of the mechanism is not moving and the links rotate around their lower joints attached to the base when the top ends of the links are moving with the top plate. The position of the center and orientation of the top plate can be represented as a 4x4 homogeneous matrix \( A_{0} \), which depends on the task configuration and can be simulated in the designed workspace. The position and orientation of the top end of the link \( i \) can also be represented as a matrix \( A_{i} \), which is related to the origin coordinate \( A_{0} \) and is constant. Therefore we always can determine the position of the top end of the link \( i \), \( (x_{i}, y_{i}, z_{i}) \), corresponding to the lower end position \( (x_{i}^{0}, y_{i}^{0}, z_{i}^{0}) \). The links (1,1) and (2,2) have top end coordinates \( (x_{1}, y_{1}, z_{1}) \) and \( (x_{2}, y_{2}, z_{2}) \), such that the line equations of these two links' axes can be formulated as

\[
\frac{x-x_{1}^{0}}{x_{1}-x_{1}^{0}} = \frac{y-y_{1}^{0}}{y_{1}-y_{1}^{0}} = \frac{z-z_{1}^{0}}{z_{1}-z_{1}^{0}} \tag{29}
\]

and

\[
\frac{x-x_{2}^{0}}{x_{2}-x_{2}^{0}} = \frac{y-y_{2}^{0}}{y_{2}-y_{2}^{0}} = \frac{z-z_{2}^{0}}{z_{2}-z_{2}^{0}} \tag{30}
\]

Equation 29 and 30 can also be written as

\[
\frac{x-x_{1}^{0}}{l_{1}} = \frac{y-y_{1}^{0}}{m_{1}} = \frac{z-z_{1}^{0}}{n_{1}} \tag{31}
\]

and

\[
\frac{x-x_{2}^{0}}{l_{2}} = \frac{y-y_{2}^{0}}{m_{2}} = \frac{z-z_{2}^{0}}{n_{2}} \tag{32}
\]

where \( l_{i} \), \( m_{i} \), \( n_{i} \) represents the denominators in Equation 29 and 30.
The perpendicular distance between two lines can be derived as

$$d_{(a,b)} = \frac{|(x_2 - x_1) m_a + (y_2 - y_1) n_a + (z_2 - z_1) l_a|}{(m_a^2 + n_a^2 + l_a^2)^{1/2}}$$  (33)

where

$$M_n = \begin{vmatrix} m_1 & n_1 \\ m_2 & n_2 \end{vmatrix}$$  (34)

$$N_l = \begin{vmatrix} n_1 & l_1 \\ n_2 & l_2 \end{vmatrix}$$  (35)

$$M_m = \begin{vmatrix} l_1 & m_1 \\ l_2 & m_2 \end{vmatrix}$$  (36)

which are all known from a given trajectory. We can calculate the distances for all possible pairs of the links $d_{(i,j)}$ and determine whether each of them is greater than a given minimum distance $\delta$. We can perform this process for any given trajectory of the end-effector by incremental stepping in off-line planning. If the step is small enough and the distance function (33) is continuous, we can estimate all possible intersections in a given workspace.

We also can calculate the mutual moment of two lines' Plucker coordinates from Equation 13 in which the right hand side is zero or small number. However, we can only know the links are either intersected or parallel, but cannot determine which. In other words, the intersection is a necessary condition for zero mutual moment, but the condition is not sufficient. Therefore, we would rather use the method proposed above.

We may also note that if the top ends of two links intersect in the mechanism, the links never intersect each other and therefore the intersection does not have to be investigated. What we need to calculate is only the distance to those links which do not intersect at the ends.

7. CONCLUSIONS

In the paper, decoupling problem in implementations of passive compliance and parallel mechanisms is addressed. To find a geometric configuration corresponding to such a decoupled compliance of the end-effector is an unsolved problem. The paper presents a new concept of an orthogonal Jacobian mechanism in which the compliance matrix is diagonal. The mechanism configuration can be found using Plucker coordinates. The process is simple and illustrated by several examples. The nonlinear equations is derived and a program is designed to generate such mechanism configurations. The mechanism can be selected for
a specific compliance of each direction in the task space. Dynamics of the mechanism is also addressed and the dynamic performance can be similarly specified easily by certain configurations and geometric parameters. When the end-effector is in motion, avoidance of the intersection is one of the key problems in implemention of the parallel manipulator. We present a method to determine whether the links are intersected for a given trajectory in off-line planning. The method can also be used for all parallel mechanisms whose configurations are specified.

8. REFERENCES


A.8 Nonlinear Control of a Robot with a Passive Compliant Wrist
NONLINEAR FEEDBACK CONTROL OF ROBOT MANIPULATOR AND COMPLIANT WRIST

Yangsheng Xu
Xiaoping Yun
Richard P. Paul

General Robotics Active Sensory Perception Laboratory
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104

ABSTRACT

Many robotic applications require the direct contact of the end-effector with the environment. Passive compliance is desirable to produce smooth transitions between the free motion and the constrained motion. However, the introduction of the passive compliance degenerates the positioning capability of the manipulator. In this paper, the dynamic position control of the manipulator with a compliant wrist is addressed. The measured deformation information of the instrumented compliant wrist is utilized in the feedback loop to increase the stiffness of the overall system. Applying nonlinear feedback control techniques, the dynamics of the manipulator-wrist system is linearized and decoupled, which allows the controller design to be carried out by using the linear system theory.

1. INTRODUCTION

Most industrial robots today are utilized to perform tasks in which the end-effectors are in contact with the environment. The manipulator continuously changes between the constrained and unconstrained motions. When the manipulator is constrained, force is controlled, while as the manipulator is in free space, position is controlled. Between those two modes is a transition, where the force or velocity are discontinuous. In order to accommodate the discontinuity, passive compliance may be inserted between the manipulator and its end-effector [8][3]. Passive compliance is also beneficial in providing the ability to let external constraints modify the trajectory of the end-effector so that the robot has an adaptive output which allows self-correction in order to accommodate geometric uncertainties in assembly task which normally presents costly problems. If a robot manipulator is equipped with such a device, the positioning capability and workpiece tolerances can be relaxed, and the high forces or moments usually caused by assembly of parts reduced [2][7].
Passive compliance can present advantages in force control. The gain of the force control feedback can be higher because of passive compliance [8][4], and the passive compliance can provide significant damping which is necessary in the force control [8]. However, in position control problems, the end-effector stiffness is greatly decreased due to the passive compliance and the positioning capability is reduced. Also, the passive compliance could present an uncertainty problem in the force control. To overcome those problems, active sensing in the compliance is desirable [8]. Such a device was developed in the University of Pennsylvania [3]. The passive compliance is made of a compliant rubber which has compliance in all directions and about all axes. The device is instrumented by providing a simple six joint serial link device with potentiometers at its joints. Displacement is continuously available for control purpose. The device provides the spring-shock absorber analogue for the manipulator while also providing sensor information.

The sensor information of the wrist can be utilized to increase the apparent stiffness of the manipulator-wrist system in position control and to further improve the compliance of the system in force control. This paper is concerned with the dynamic position control of the manipulator-wrist system. The nonlinear feedback technique is applied to linearize and decoupled the nonlinear dynamics of the actuators, manipulator links, and compliant wrist.

Nonlinear feedback control theory and its application to the robot manipulator control have been addressed in [6][5]. The technique has been shown to be powerful for robot control, especially if the dynamics of the system is of significance. Since the compliant wrist contributes a strong dynamic property to the robot system and thereby the dynamic control of the system is rather important. In this paper, the robot manipulator and compliant wrist system is modeled with consideration of the compliance of actuators, links, and wrist. The nonlinear feedback control method is used to deal with the dynamic control of the system. Control of a two-link manipulator and a compliant wrist with one degree of freedom is derived to illustrate the design procedure.

2. DYNAMICS OF THE MANIPULATOR-WRIST SYSTEM

Instead of modeling the actuators as pure torque/force sources, we consider the actuator dynamics and the drive train system compliance. Most industrial robots are actuated electrically, hydraulically, or pneumatically. We confine ourselves to robot manipulators driven by DC motors.

Neglecting the inductance, the voltage equation of the armature circuit of the DC motor is given by

\[ R_a + K_b \dot{\theta}_a / n = v \]  

(1)

where
$R_a$ is armature resistance,

$i_a$ is armature current,

$K_b$ is voltage constant of the motor,

$\theta_a$ is angular rotor position reflected to the load shaft (manipulator joint), i.e., $\theta_a = n \theta_m$

$\theta_m$ is the angular rotor position,

$n$ is gear reduction ratio,

$\nu$ is armature voltage.

The torque produced by the motor is proportional to the armature current, i.e.

$$\tau_a = K_a i_a$$

(2)

The actuators drive the manipulator links through drive system such as gears. We model the drive system as a spring device, i.e., the torque applied to the manipulator links is given by

$$\tau = K_f (\theta_a - \theta_l)$$

(3)

where $\theta_l$ is the position of the link and $K_f$ is the stiffness of the drive system.

The motion equation of the motor is

$$J_{eff} \ddot{\theta}_a + f_{eff} \dot{\theta}_a + K_f (\theta_a - \theta_l) = n \tau_a$$

(4)

Substituting Equations (1) and (2) into Equation (3), we obtain

$$R_a J_{eff} \ddot{\theta}_a + R_a f_{eff} \dot{\theta}_a + K_a K_b \theta_a + R_a K_f \theta_a = n K_a \nu - R_a K_f \theta_l$$

(5)

For a robot manipulator with six joints, we view Equation (5) as a vector equation. That is, $\theta_a = [\theta_{a1}, \theta_{a2}, ..., \theta_{a6}]^T$, $\theta_l = [\theta_{l1}, \theta_{l2}, ..., \theta_{l6}]^T$, and all coefficients are diagonal matrices, e.g., $R_a = \text{diag}(R_{a1}, R_{a2}, ..., R_{a6})$.

The compliant wrist is installed at the end of the manipulator. If we use $X_w$ to represent the position and orientation of the outer plate of the wrist in the base coordinate frame, the dynamics of the wrist can be written as follows

$$M_w \ddot{X}_w + C_w (\dot{X}_w - J_f \dot{\theta}_l) + K_w (X_w - f_l(\theta_l)) = F$$

(6)

where
\( M_w \): inertia matrix of the compliant wrist,

\( C_w \): damping coefficient matrix of the wrist,

\( K_w \): stiffness matrix of the wrist,

\( F \): external force applied to the wrist,

\( f_1 \): the forward kinematics of the manipulator --- a 6×1 vector function representing the position and orientation of the tool coordinates in the base coordinates,

\( J_t \): the Jacobian matrix of the manipulator.

The motion equations of the manipulator links are described by

\[
D(\dot{\theta}_t)\ddot{\theta}_t + H(\dot{\theta}_t, \theta_t) + J_t^T [C_w(\ddot{X}_w - J_t \dot{\theta}_t) + K_w(X_w - f_1(\theta_t))] = \tau
\]

where \( D(\dot{\theta}_t) \) is the inertia matrix and \( H(\dot{\theta}_t, \theta_t) \) is the Coriolis, centrifugal, and gravity forces. The third term in the bracket represents the reaction force between the manipulator and the compliant wrist. The torque \( \tau \) on the right-hand side is the same torque as in Equation (3).

We will write Equations (5), (6) and (7) together in matrix form

\[
\begin{bmatrix}
R_a I_{eff} & 0 & 0 \\
0 & D(\dot{\theta}_t) & 0 \\
0 & 0 & M_w
\end{bmatrix}
\begin{bmatrix}
\ddot{\theta}_a \\
\dot{\theta}_t \\
\ddot{X}_w
\end{bmatrix}
+ \begin{bmatrix}
(R_a[f_{eff} + K_a K_b] \dot{\theta}_a + R_a K_t \theta_a - R_a K_t \theta_t \\
H(\dot{\theta}_t, \theta_t) + J_t^T [C_w(\ddot{X}_w - J_t \dot{\theta}_t) + K_w(X_w - f_1(\theta_t))] + K_t(\dot{\theta}_t - \theta_a) \\
C_w(\ddot{X}_w - J_t \dot{\theta}_t) + K_w(X_w - f_1(\theta_t)) - F
\end{bmatrix}
= \begin{bmatrix}
nK_a \\
0 \\
0
\end{bmatrix} \nu
\]

This equation can be rewritten as

\[
\begin{bmatrix}
\ddot{\theta}_a \\
\dot{\theta}_t \\
\ddot{X}_w
\end{bmatrix}
= \begin{bmatrix}
J_{\theta\theta}^{-1}\{[f_{eff} + R_a^{-1}K_a K_b] \dot{\theta}_a + K_t \theta_a - K_t \theta_t\} - D_{\text{eff}}(\theta_t) \{H(\dot{\theta}_t, \theta_t) + J_t^T [C_w(\ddot{X}_w - J_t \dot{\theta}_t) + K_w(X_w - f_1(\theta_t))] + K_t(\dot{\theta}_t - \theta_a)\}
+ M_w^{-1}[C_w(\ddot{X}_w - J_t \dot{\theta}_t) + K_w(X_w - f_1(\theta_t)) - F]
+ nJ_{\theta\theta}^{-1}R_a^{-1}K_a
\\
0 \\
0
\end{bmatrix} \nu
\]
Using a shorthand notation, we represent the above equation by
\[
\begin{bmatrix}
\dot{\theta}_a \\
\dot{\theta}_l \\
\dot{X}_w
\end{bmatrix} = f^2 + g^2 v = f^2 +
\begin{bmatrix}
g^2 \\
0 \\
0
\end{bmatrix}
\tag{10}
\]
where \(g^2 = \mathbf{n} \mathbf{J}_q \mathbf{J}^T \mathbf{K}_a^{-1} \mathbf{K}_a\), and \(f^2\) and \(g^2\) can be easily identified.

3. NONLINEAR FEEDBACK CONTROL

Equation (10) describes the complete motion equations of the actuator, manipulator and wrist system. Our purpose is to control such a system so that the wrist will follow a desired trajectory. That is, we want the actual \(X_w\) to follow the desired trajectory. In this case, our output equation is clearly \(X_w\). We use the following notation to represent the output equation
\[
y = h(\theta_a, \theta_l, X_w) = X_w
\tag{11}
\]
To control such a complicated system, we intend to linearize and decouple it by using an appropriate nonlinear feedback. For convenience, we introduce the following notation
\[
x^1 = [\theta_a, \theta_l, X_w]^T
\]
\[
x^2 = [\dot{\theta}_a, \dot{\theta}_l, \dot{X}_w]^T
\]
\[
x = [x^1, x^2]^T
\]
From Equation (10), we have
\[
\dot{x}^2 = f^2(x^1, x^2) + g^2 v
\]
Considering \(\dot{x}^1 = x^2\), we have the motion equations of the actuator, manipulator and wrist in terms of \(x\)
\[
\dot{x} = \begin{bmatrix}
\dot{x}^1 \\
\dot{x}^2
\end{bmatrix} = \begin{bmatrix}
x_2 \\
f^2(x^1, x^2) + g^2 v
\end{bmatrix}
\]
or in more compact form
\[
\dot{x} = f(x) + g v
\tag{12}
\]
The control problem of the manipulator-wrist system becomes a standard control problem of an affine nonlinear system described by state Equation (12) and output Equation (11). We will find a nonlinear feedback of form
\[
v = \alpha(x) + \beta(x) u
\tag{13}
\]
such that the present nonlinear system will be equivalent to a decoupled linear system. In case that the dynamics of actuators and compliant wrist is absent, the corresponding problem (position control of manipulators) can be solved easily. Nonlinear cancellation by inspection results in the computed torque method. For the present problem, it is not easy to cancel the nonlinearity by inspection. We will apply the
differential geometric control theory [1][6] in order to find a nonlinear feedback of the form of Equation (13). In doing so, the following computation of Lie derivatives is necessary.

\[ L_g h = \frac{\partial h}{\partial x} g = 0 \]

\[ L_f h = \frac{\partial h}{\partial x} f = \frac{\partial h}{\partial x} x^2 = \dot{X}_w \]

\[ L_g L_f h = \frac{\partial L_f h}{\partial x} g = 0 \]

\[ L_f^2 h = \frac{\partial L_f h}{\partial x} f = M_w^{-1} [C_w (\dot{X}_w - J_f \dot{\theta}) + K_w (X_w - f_i(\theta))] - F \]

\[ L_g L_f^2 h = 0 \]

\[ L_f^3 h = \frac{\partial L_f^2 h}{\partial x} f = \frac{\partial L_f^2 h}{\partial x} x^2 + \frac{\partial L_f^2 h}{\partial x} f^2 \]

\[ = \frac{\partial L_f^2 h}{\partial \dot{\theta}_i} \dot{\theta}_i - M_w^{-1} K_w \ddot{X}_w + \left[0, -M_w^{-1} J_1, M^{-1} C_w \right] f^2 \]

\[ L_g L_f^3 h = 0 \]

\[ L_f^4 h = \frac{\partial L_f^3 h}{\partial x} f = \frac{\partial L_f^3 h}{\partial x} x^2 + \frac{\partial L_f^3 h}{\partial x} f^2 \]

\[ = -M_w^{-1} J_1 D^{-1}(\theta) K_1 \dot{\theta}_a + \frac{\partial L_f^3 h}{\partial \theta_1} \dot{\theta}_1 + \frac{\partial L_f^3 h}{\partial X_w} \ddot{X}_w + \left[0, \frac{\partial L_f^3 h}{\partial \theta_1}, \frac{\partial L_f^3 h}{\partial X_w} \right] f^2 \]

\[ L_g L_f^4 h = \frac{\partial L_f^4 h}{\partial x} g = \frac{\partial L_f^4 h}{\partial \theta_a} g^2 = -M_w^{-1} J_1 D^{-1}(\theta) K_1 g^2 \]

\[ = -n M_w^{-1} J_1 D^{-1}(\theta) K_1 J_1^{-1} R_a^{-1} K_a = \Phi(x) \]

where \( \Phi(x) \) is the so-called decoupling matrix.

Then the \( \alpha(x) \) and \( \beta(x) \) in the nonlinear feedback (13) are defined by

\[ \alpha(x) = -\Phi^{-1}(x) L_f^4 h \] \hspace{1cm} (14)

\[ \beta(x) = \Phi^{-1}(x) \] \hspace{1cm} (15)

We notice that the \( \alpha(x) \) and \( \beta(x) \) are well defined in the whole work space but the singular configurations of the manipulator. In that case, \( J_1 \) is singular. \( J_1^{-1} \) does not exist, neither does \( \Phi^{-1} \).

From the differential geometric control theory [1], application of the derived nonlinear feedback results in a chain of integrators from the new reference input \( u \) to the output \( y \). Specifically, we have

\[ y^{(5)} = u \] \hspace{1cm} (16)

That is, the input-output relation is linearized and decoupled. Therefore, position control of the manipulator-wrist system becomes a design problem of linear system (16).
4. CONTROL OF A TWO-LINK PLANAR ROBOT MANIPULATOR WITH A COMPLIANT WRIST

We now consider a two-link planar manipulator. A compliant wrist is attached to the end of the second link. We assume that the wrist has compliance in only one direction along the length of the second link. (see Figure 1.)

The tool position $X_t$ is given by

$$X_t = f_t(\theta_1) = \begin{bmatrix} l_1 \cos \theta_1 + l_2 \cos (\theta_1 + \theta_2) \\ l_1 \sin \theta_1 + l_2 \sin (\theta_1 + \theta_2) \end{bmatrix}$$

The Jacobian is then

$$J_t(\theta_1) = \begin{bmatrix} -l_1 \sin \theta_1 - l_2 \sin (\theta_1 + \theta_2) & -l_2 \sin (\theta_1 + \theta_2) \\ l_1 \cos \theta_1 + l_2 \cos (\theta_1 + \theta_2) & l_2 \cos (\theta_1 + \theta_2) \end{bmatrix}$$

The motion equation of the compliant wrist are governed by

$$M_w \ddot{X}_w + C_w (\dot{X}_w - J_t(\theta_1) \dot{\theta}_1) + K_w (X_w - f_t(\theta_1)) = F$$

where $M_w$, $C_w$, and $K_w$ are scalar constants associated with the wrist, and $X_w = (X_{wx} X_{wy})$ is the position of the outer plate of the wrist.

The dynamics of the manipulator links is described by

$$D(\theta_1) \ddot{\theta}_1 + H(\dot{\theta}_1, \theta_1) + J_t^T [C_w (\dot{X}_w - J_t \dot{\theta}_1) + K_w (X_w - f_t(\theta_1))] = \tau$$

where

$$\theta_1 = [\theta_{11}, \theta_{12}]^T$$

$$D(\theta_1) = \begin{bmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{bmatrix}$$

$$H(\dot{\theta}_1, \theta_1) = \begin{bmatrix} -m_2 l_1 l_2 \sin \theta_{12} (\dot{\theta}_{12}^2 + 2 \dot{\theta}_1 \dot{\theta}_{12}) + G_1 \\ m_2 l_1 l_2 \sin \theta_{12} \dot{\theta}_{12}^2 + G_2 \end{bmatrix}$$

and

$$D_{11} = m_1 l_2^2 + I_1 + m_2 (l_1^2 + l_2^2 + 2 l_1 l_2 \cos \theta_{12}) + I_2$$

$$D_{22} = m_2 l_2^2 + I_2$$

$$D_{12} = D_{21} = m_2 l_1 l_2 \cos \theta_{12} + m_2 l_2^2 + I_2$$

$$G_1 = m_1 l_{12} \cos \theta_{11} m_2 g (l_2 \cos (\theta_{11} + \theta_{12}) + l_1 \cos \theta_{11})$$

$$G_2 = m_2 l_2 g \cos (\theta_{11} + \theta_{12})$$

For simplicity, we exclude the dynamics of the actuators and regard them as pure torque sources. Let
\[ x^1 = [\dot{\theta}_1, \dot{\theta}_{12}, X_{wx}, X_{wy}]^T \]
\[ x^2 = [\ddot{\theta}_1, \dot{\theta}_{12}, \dot{X}_{wx}, \dot{X}_{wy}]^T \]
\[ x = [(x^1)^T, (x^2)^T]^T \]

Then we have
\[
\dot{x} = \begin{bmatrix} \dot{x}^1 \\ \dot{x}^2 \end{bmatrix} = \begin{bmatrix} x^2 \\ f^2(x) \end{bmatrix} + \begin{bmatrix} 0 \\ g^2 \end{bmatrix} \tau = f(x) + g \tau
\] (21)

where
\[
f^2(x) = \begin{bmatrix} f^2 \\ \dot{f}^2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} -D^{-1}(\theta_1)H(\dot{\theta}_1, \theta_1) \end{bmatrix} - D^{-1}(\theta_1)J^T(\theta_1)[C_w (\ddot{X}_{wx} - J_1(\theta_1) \dot{\theta}_1) + K_w (X_{wy} - f_1(\theta_1))] \\
-M_w^{-1}(C_w (\ddot{X}_{wx} - J_1(\theta_1) \dot{\theta}_1) + K_w (X_{wy} - f_1(\theta_1))] + M_w^{-1}F \end{bmatrix} \\
0 \end{bmatrix}
\]

\[ g^2 = \begin{bmatrix} D^{-1}(\theta_1) \\ 0 \end{bmatrix} \]

For the trajectory-following tasks, the output equation of the two-link planar robot is
\[ y = h(x) = X_w \] (22)

We now follow the procedure in the previous section to compute the nonlinear feedback for this particular case.

\[ L_x h = \frac{\partial h}{\partial x} g = 0 \]
\[ L_f h = \frac{\partial h}{\partial x} f = \dot{X}_w \]
\[ L_{xx} h = \frac{\partial L_f h}{\partial x} g = 0 \]
\[ L_{xf} h = \frac{\partial L_f h}{\partial x} f = f \dot{f} \]
\[ L_{xx} L_f h = \frac{\partial L_{xf} h}{\partial x} g = M_w^{-1}C_w J_1(\theta_1)D^{-1}(\theta_1) = \Phi(x) \]

Then the nonlinear feedback \( \tau = \alpha(x) + \beta(x) u \) will fulfill linearization and decoupling with
\[ \alpha(x) = -\Phi^{-1}(x)L_f h = -D(\theta_1)J_1^{-1}(\theta_1)C_w^{-1}M_w L_f h \] (23)
\[ \beta(x) = \Phi^{-1}(x) = D(\theta_1)J_1^{-1}(\theta_1)C_w^{-1}M_w \] (24)

The term \( L_{xx} h \) in the expression of \( \alpha(x) \) is computed as follows.
\[
L_{xx} h = \frac{\partial L_{xf} h}{\partial x} f
\]
\[ = M_w^{-1} \frac{\partial}{\partial \theta_1} [C_w J_1(\theta_1) \dot{\theta}_1 + K_w f_1(\theta_1)] \dot{\theta}_1 - M_w^{-1} K_w \ddot{X}_w + M_w^{-1} C_w J_1(\theta_1) f \dot{f} - M_w^{-1} C_w f \dot{f} \]
\[ = M_w^{-1} [(C_w J_1(\theta_1) + K_w J_1(\theta_1)) \dot{\theta}_1 - K_w \ddot{X}_w + C_w J_1(\theta_1) f \dot{f} - C_w f \dot{f}] \]
where
\[
J_i^w(\theta_t) = \begin{bmatrix}
-l_1 \cos \theta_{t1} - l_2 \cos (\theta_{t1} + \theta_{t2}) & -l_2 \cos (\theta_{t1} + \theta_{t2}) \\
-l_1 \sin \theta_{t1} - l_2 \sin (\theta_{t1} + \theta_{t2}) & -l_2 \sin (\theta_{t1} + \theta_{t2})
\end{bmatrix}
\begin{bmatrix}
\dot{\theta}_{t1} \\
\dot{\theta}_{t2}
\end{bmatrix}
\begin{bmatrix}
\cos (\theta_{t1} + \theta_{t2}) & \cos (\theta_{t1} + \theta_{t2}) \\
\sin (\theta_{t1} + \theta_{t2}) & \sin (\theta_{t1} + \theta_{t2})
\end{bmatrix}
\begin{bmatrix}
l_2 \dot{\theta}_{t2}
\end{bmatrix}
\]
which is simply the Jacobian of \(J_i(\theta_t)\dot{\theta}_t\).

Applying the nonlinear feedback to the dynamics of the two-link manipulator and the wrist, we obtain a linear and decoupled system
\[
y^{(3)} = u
\]  
(25)
or
\[
X_o^{(3)} = u_1
\]
\[
X_o^{(3)} = u_2
\]
provided that we have the perfect model parameters. We note that the \(\alpha(x)\) and \(\beta(x)\) are completely determined by the system parameters and the external force \(F\). They can be straightforwardly programmed in simulations and implementation.

To complete the controller design, we simply use a linear constant feedback to properly locate the poles of the linear system (25). We comment here that \(X_o\) is not directly measured since we do not assume utilization of camera or such. However, the position sensors mechanism is instrumented in the compliant wrist to provide the deformation information, which, together with the joint position of the manipulator, will allow us to compute \(X_o\).

5. CONCLUSIONS

The dynamic position control of a manipulator with a compliant wrist was discussed in this paper. The actuator model and drive train compliance were considered. The measured deformation data from the compliant wrist was utilized in feedback loop to increase the virtual stiffness of the system.

The nonlinear feedback theory was successfully applied to the manipulator-wrist system. A nonlinear feedback was found which linearizes and decouples the entire system. An example of two-link planar manipulator was presented to demonstrate the computation of the nonlinear feedback.

6. REFERENCES


Figure 2. Diagram of Nonlinear Feedback Control

Figure 1. Two-Link Manipulator and one-DOF Compliant Wrist
A.9 The Hamilton Wrist
The Hamilton Wrist: A Four-Revolute-Joint Spherical Wrist Without Singularities

Gregory L. Long* Richard P. Paul* William D. Fisher**

* GRASP Laboratory
  University of Pennsylvania
  Philadelphia, Pennsylvania 19104

** Hewlett-Packard Laboratories
  Palo Alto, California 94304

Abstract

A spherical wrist can perform differential Cartesian motions only when the set of joint-screws belong to a third order screw system, and the corresponding twist-velocities are well-defined and finite. When the joint-screws do not belong to a third order system, and/or the twist-velocities are ill-defined, control of the wrist will be difficult if not impossible. It is impossible for a three-revolute-joint spherical wrist to provide for a complete set of spatial orientations while still maintaining a third order system. However, when an additional joint is added to the original three-revolute-joint wrist, it is possible to maintain a third order system while providing for a set of well-defined and finite twist-velocities. From kinematic considerations, a closed-form non-conservative inverse kinematic algorithm is derived which satisfies the above criteria of motion. The number of mathematical operations in the algorithm for the four-revolute-joint wrist is comparable with those for a three-revolute-joint wrist, which makes the solution suitable for real-time computation.

1 Introduction

The spatial motion of a robot manipulator can be expressed in several forms. It is convenient to decouple the motion into the translation of a point followed by an orientation about that point. For many six-joint manipulators, this decoupling is realized by utilizing the first three joints as a positioning mechanism, while the last three joints are used as an orienting mechanism.

The joints used for the orienting mechanism are usually revolute pairs whose axes intersect. When these intersecting axes are coupled in serial form with the appropriate link twist
angles, they comprise a *spherical wrist*, as the orienting structure is kinematically equivalent to the three degree-of-freedom spherical pair.

Six-joint robot manipulators with 3-\(R\) spherical wrists have been studied by Paul and Stevenson [1983] and by Asada [1985]. Because of the excessive twist-velocities (joint velocities) near singular regions, Paul and Stevenson observed that these regions must be eliminated from the wrist's workspace. Hence, they suggest that a six-joint manipulator is best characterized as a five degree-of-freedom device than as a six degree-of-freedom device.

To alleviate the orienting singularity, Fisher [1984], Hollerbach [1984], and Stanišić and Pennock [1985] have suggested using a 4-\(R\) spherical wrist. Hollerbach examines the 4-\(R\) spherical wrist in the context of determining the optimum kinematic design for a seven degree-of-freedom manipulator, while Fisher, and Stanišić and Pennock have closed-form solutions.

Using topological arguments, Gottlieb [1986], Baker and Wampler [1987, 1988], and Wampler [1987] make clear that "additional" singularities exist in redundant orientation devices that use *conservative* inverse kinematic algorithms (a conservative algorithm is independent of path and time). Essentially, for redundant \(n\)-\(R\) spherical wrists with conservative inverse kinematic algorithms, it may be possible to maintain a third order screw system, however, at least one of the twist-velocities will be ill-defined somewhere in space; the conservative algorithm given by Stanišić and Pennock [1985] is an example where a third order system is always maintained, however, the twist-velocity of the first joint is ill-defined when the first and fourth joint-screws are collinear.

Fisher [1984] has developed a non-conservative inverse kinematic algorithm for the 4-\(R\) spherical wrist which maintains a third order screw system while providing for a set of bounded twist-velocities. Fisher's conceptualization of the 4-\(R\) wrist as a 3-\(R\) wrist with one redundant joint is embodied in this paper, however, we endeavor to improve upon Fisher's algorithm with modifications based on the kinematic geometry of the 4-\(R\) wrist.
2 Kinematic Singularities and Screw Systems

Control algorithms for a robot manipulator are complimented with information from the manipulator’s screw system. In this section, we discuss the elements of screw systems which are applicable to spherical wrists. For more details on screw theory, the summary given by Roth [1980] is recommended; for in depth studies, the reader is directed to the works by Ball [1900] and Hunt [1979].

2.1 Screw Systems: Fundamentals

The screw theory of mechanics is based on two fundamental theorems, one given by Chasles’ and the other by Poinset. Chasles’ theorem states that the cannonical form of a rigid-body displacement consists of a single rotation about a unique axis coupled with a translation parallel to that axis. The theorem due to Poinset states that a system of forces and moments acting on a rigid-body can be reduced to a single force and a moment, such that both the force and moment lie along the same axis. This unique axis is called the screw-axis; when a pitch is associated with this axis, the combination is simply called a screw. Hence, a rigid-body displacement can be achieved by a twist about a screw, and a system of forces and moments applied to a rigid-body is equivalent to a wrench on a screw.

In robotics, the joint-screw has either a zero pitch, a revolute joint, or an infinite pitch, a prismatic joint; in the analysis of revolute-joint spherical wrists, we are concerned only with zero-pitch joint-screws. Hence, a twist about zero-pitch joint-screw is a pure rotation, and the twist-velocity is the joint’s angular velocity.

Ball [1900] named the aggregate of joint-screws corresponding to an \( n \) degree-of-freedom system a screw system of order \( n \), or \( n \)-system. Hunt [1979] extended Ball’s work by classifying the screw systems into what he calls ‘special’ screw systems. When the joint-screws have zero pitch, by Hunt’s categorization, the screw system which describes spherical motion is the second special three-system (see Hunt [1979] Chapter 12).

A screw \( \hat{S} \) is reciprocal to another screw \( \hat{S}^' \) if every wrench along \( \hat{S}^' \) does no work on
a rigid-body constrained to twist about $\hat{S}$. At a kinematic singularity, there is at least one screw which is reciprocal to the original set of joint-screws. This reciprocal screw is characterized by a direction in which the manipulator cannot apply a wrench.

2.2 The Reciprocal Screw for the 3-R Wrist

By limiting our analysis to the system of screws which are fundamental to spherical wrists, we find that for non-singular differential motion, our joint-screws must belong to the second special three-system. When the 3-R wrist is in a singular configuration, the system of screws reduces from a three-system to a two-system, and there will be "one" screw reciprocal to this two-system (Note: ordinarily, a four-system is reciprocal to a two-system, and a four-system comprises a triple-infinity ($\infty^3$) of screws. Again, however, we note that we are only concerned with those screws fundamental to revolute-joint spherical wrists).

This reciprocal screw can be found by inspection. However, as we desire to form a general method for the treatment of kinematic singularities, we take the exercise of finding this reciprocal screw.

Since the wrist has only zero-pitch screws, the screw coordinates of $\hat{S}_i$ are given by Plücker's line coordinates

$$\hat{S}_i = \{S_i; S_{0i}\}$$

where $S_i$ is a vector along the screw-axis, and $S_{0i}$ is the moment of $S_i$ about the origin.

The two screws $\hat{S}$ and $\hat{S}'$ are reciprocal when

$$S \cdot S_0' + S_0 \cdot S' = 0$$

From Figure (1) we see that the screw coordinates of $\hat{S}_1$, $\hat{S}_2$, and $\hat{S}_3$, in the $O_w$ coordinate system, are given by

$$\hat{S}_1 = \{0,0,1;0,0,0\}$$
$$\hat{S}_2 = \{S\theta_1, C\theta_1, 0;0,0,0\}$$
$$\hat{S}_3 = \{S\theta_2 C\theta_1, S\theta_2 S\theta_1, C\theta_2;0,0,0\}$$

4
Figure 1: 3-R Spherical Wrist in Singular Configuration

where,

\[ S\theta_i = \sin \theta_i \quad \text{and} \quad C\theta_i = \cos \theta_i \]

At a singular configuration \( S\theta_2 = 0 \), and \( \hat{S}_1 \) and \( \hat{S}_3 \) are dependent. Thus, the reciprocal screw system can be obtained from

\[
\begin{bmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
S\theta_1 & C\theta_1 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
S'_{ox} \\
S'_{oy} \\
S'_{oz} \\
\hat{S}'_x \\
\hat{S}'_y \\
\hat{S}'_z
\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

\[ (4) \]

Again, since we are only concerned with the screw system fundamental to the wrist

\[ S'_x = S'_y = S'_z = 0 \]

\[ (5) \]

and

\[ S'_{ox} = \mp C\theta_1, \quad S'_{oy} = \pm S\theta_1, \quad \text{and} \quad S'_{oz} = 0 \]

\[ (6) \]

Hence, the wrench on this reciprocal screw is given by

\[ w\hat{S}' = \pm w\{0, 0, 0; -C\theta_1, S\theta_1, 0\} \]

\[ (7) \]
where, \( w \) is the \textit{intensity} of the wrench. We note that Equation (7) represents a moment of arbitrary intensity, perpendicular to the plane spanned by \( S_1 \) and \( S_2 \).

When the 3-R wrist is in a singular configuration, if it where possible to place a "redundant joint-screw", \( \hat{S}_r \), perpendicular the plane spanned by \( S_1 \) and \( S_2 \), then, theoretically, there is nothing which states that differential motion cannot be achieved. The problem is then reduced to integrating this redundant joint-screw into the overall control scheme, such that the twist-velocities are well-defined and finite. The implementation of such a control scheme is the subject of this paper.

3 Nomenclature and Wrist Parameters

The right-handed Cartesian coordinate system \( x_i-y_i-z_i \) is denoted by the symbol \( O_i \), and the location of one coordinate system with respect to another is given by the Denavit-Hartenburg coordinate transformation matrix, where the coordinate transformation from the \( O_i \) system to the \( O_{i-1} \) system is given by [Paul, 1981]

\[
A_{i-1} = \begin{bmatrix} 
\cos \theta_i & -\sin \theta_i \cos \alpha_i & \sin \theta_i \sin \alpha_i & a_i \cos \theta_i \\
\sin \theta_i & \cos \theta_i \cos \alpha_i & -\cos \theta_i \sin \alpha_i & a_i \sin \theta_i \\
0 & \sin \alpha_i & \cos \alpha_i & d_i \\
0 & 0 & 0 & 1
\end{bmatrix}
\] (8)

where,

\( \theta_i \equiv \) the rotation about \( z_{i-1} \)

\( \alpha_i \equiv \) the angle between \( z_{i-1} \) and \( z_i \) along the common normal

\( a_i \equiv \) the length of the common normal

\( d_i \equiv \) the distance from \( O_{i-1} \) to the common normal

As shown in Figure (2), the joint-screws \( \hat{S}_1, \hat{S}_2, \hat{S}_3, \) and \( \hat{S}_4 \) are directed along the \( z_w, z_1, z_2, \) and \( z_3 \) axes, respectively. The normal, orientation, and approach vectors, \( w_n, w_o, \) and \( w_a \) [Paul, 1981], are given with respect to the \( O_w \) system and are directed along the \( x_4, y_4, \) and \( z_4 \) axes, respectively.
Figure 2: Coordinate Systems for the 4-R Wrist

The link twist angles \( \alpha_1, \alpha_2, \alpha_3, \) and \( \alpha_4 \) are \( \pi/2, -\pi/2, \pi/2, \) and 0, respectively. These link twist angles are optimum for a 4-R orienting mechanism; optimum in the sense that with a combination of any two joint variables \( \{\theta_1, \theta_2, \theta_3, \theta_4\} \), the approach vector, \( \omega_a \), can be directed anywhere on a unit sphere centered at \( O_w \) [Long, 1988].

The reference configuration for the wrist is given by the Denavit-Hartenburg parameters in Table (1) (see Figure (3) for configuration). Hence, the coordinate transformation from the \( O_4 \) system to \( O_w \) system is given by

\[
T^w_4 = T^w_1 A_1^2 A_3^2 A_4^3 = \begin{bmatrix}
\omega_{n_x} & \omega_{n_y} & \omega_{n_z} & 0 \\
\omega_{o_x} & \omega_{o_y} & \omega_{o_z} & 0 \\
\omega_{a_x} & \omega_{a_y} & \omega_{a_z} & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\]

(9)

where \( T^w_1, A_1^2, A_3^2, \) and \( A_4^3 \) are given by

\[
T^w_1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad A_1^2 = \begin{bmatrix} -S_2 & 0 & C_2 & 0 \\ C_2 & 0 & S_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\]

(10)
Table 1: Denavit-Hartenburg Parameters for 4-R Wrist

<table>
<thead>
<tr>
<th>link</th>
<th>( \theta_i )</th>
<th>( \alpha_i )</th>
<th>( a_i )</th>
<th>( d_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \theta_1 )</td>
<td>(-\pi/2)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>( \theta_2 + \pi/2 )</td>
<td>( \pi/2 )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>( \theta_3 + \pi/2 )</td>
<td>(-\pi/2)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>( \theta_4 - \pi/2 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
A^3_3 = \begin{bmatrix}
-S_3 & 0 & -C_3 & 0 \\
C_3 & 0 & -S_3 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}, \quad A^4_3 = \begin{bmatrix}
S_4 & C_4 & 0 & 0 \\
-C_4 & S_4 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\] (11)

In this paper, several important orientations of \( \omega_a \) are used to derive the inverse kinematic algorithm for the wrist, and the description of these orientations is simplified using geocentric terminology. The polar axis intersects the orienting region at \( \pm 1 \), where the point at \( +1 \) is called the north pole and the point at \( -1 \) is called the south pole. The equatorial plane is coincident with the \( x_w-y_w \) plane, and the equator is the great circle which lies in this plane.
4 Inverse Kinematic Algorithm

As Gottlieb [1987], and Baker and Wampler [1987, 1988] have shown, the kinematic mapping from Cartesian space to joint space for a 2-R "pointing" mechanism can be conceptualized as a topological mapping from a spherical surface to a toroidal surface, and this mapping is inherently non-continuous. With several extensions to the above argument, Baker and Wampler also show that for any redundant pointing mechanism, which includes all n-revolute-joint spherical wrists, this mapping cannot be both conservative and continuous.

Although the above authors show that any conservative inverse kinematic algorithm will be non-continuous, this does not imply that non-conservative and continuous algorithms are impossible. In other words, it may be possible to obtain a path-dependent and/or time-dependent algorithm which is continuous.

Fisher [1984] has shown that by conceptualizing the 4-R wrist as 3-R wrist with one "redundant" joint, that a non-conservative and continuous inverse kinematic algorithm can be obtained. First, Fisher determines the inverse kinematics of the 4-R wrist with one of the internal joint variables held constant ($\theta_2$ or $\theta_3$). Next, he gives an algorithm such that when the 3-R wrist encounters a joint continuity problem (which will be joint 1), the twist-velocity, $\dot{\theta}_1$, is given its maximum value and the redundant joint ($\theta_2$ or $\theta_3$) compensates for any errors in orientation.

By studying the kinematic geometry of the 4-R wrist, we have observed that it is unnecessary to give $\dot{\theta}_1$ its maximum value when the wrist encounters a continuity problem, and we strive to extend Fisher’s initial conceptions and algorithm.

First, we observe that $\theta_2$ and $\theta_3$ have symmetric roles in the kinematic geometry of the 4-R wrist, and a primary 3-R wrist can be formed with either of the following two sets of joint variables:

\[
\{\theta_1, \theta_2, \theta_4\} \text{ or } \{\theta_1, \theta_3, \theta_4\}
\]
We have arbitrarily chosen the first set of joint variables to comprise the primary wrist, and we denote the inverse kinematic solution for this wrist with a subscript \( p \) for primary.

Hence, when \( \theta_{3,p} \) is given a predetermined value, explicit solutions for \( \theta_{1,p}, \theta_{2,p}, \) and \( \theta_{4,p} \) can be obtained in terms of \( w_n, w_o, \) and \( w_a. \)

A characteristic of the primary solution is that as \( w_a \) approaches and passes through a singular region, \( \dot{\theta}_{1,p} \) gets exceedingly large and becomes unbounded. In fact, \( \theta_{1,p} \) is unstable in the vicinity of a singular region. Indeed, for any \( n \)-revolute-joint orientation device, Baker and Wampler [1988] show that any conservative inverse kinematic algorithm will have either two singular regions, one in both its northern and southern hemispheres, or one singular region at its equator. For the primary solution given in this paper, the singular regions are at the north and south poles.

We note that these singularities in the primary solution are not configuration singularities for the 4-\( R \) wrist, as we have judiciously placed our redundant joint-screw, \( S_3, \) so that it will be perpendicular to the plane spanned \( S_1 \) and \( S_4. \) Hence, when the wrist approaches and passes through the singular regions of the primary wrist, in theory, differential motion can still be achieved.

Near the singular regions, the following two sets of joint-screws form three-systems

\[
\{\dot{S}_1, \dot{S}_2, \dot{S}_3\} \text{ and } \{\dot{S}_2, \dot{S}_3, \dot{S}_4\}
\]

However, since the twist-velocity corresponding to \( \dot{S}_1 \) will be ill-defined in the primary solution, \( \dot{\theta}_{1,p} \) is unbounded, the second set of joint-screws is a better choice for achieving differential motion. The objective of this paper is to present a method by which the control of the wrist is transferred from the primary set of joint-screws, \( \{\dot{S}_1, \dot{S}_2, \dot{S}_4\}, \) to a secondary set of joint-screws given by \( \{\dot{S}_2, \dot{S}_3, \dot{S}_4\}, \) such that the transition is smooth and continuous, and that the control algorithm is simple.
4.1 Primary Solution

The primary values $\theta_{1,p}, \theta_{2,p},$ and $\theta_{4,p}$ are obtained by setting $\theta_{3,p} = 0$. From Long and Paul [1988], the inverse kinematics for the primary solution is given by

\[
\theta_{1,p} = \tan^{-1}\left\{ \frac{w a_y}{w a_x} \right\} \tag{12}\]

\[
\theta_{2,p} = \tan^{-1}\left\{ \frac{\sqrt{(w a_x)^2 + (w a_y)^2}}{w a_x} \right\} \tag{13}\]

\[
\theta_{3,p} = 0 \tag{14}\]

and

\[
\theta_{4,p} = \tan^{-1}\left\{ \frac{-w n_z S \theta_{1,p} + w n_y C \theta_{1,p}}{-w o_x S \theta_{1,p} + w o_y C \theta_{1,p}} \right\} \tag{15}\]

Since the $\tan^{-1}$ function in Equation (12) is unstable when both $w a_x$ and $w a_y$ approach zero, the two singular points in the orienting region correspond to when $w a_x = \pm 1$, which occurs at the north and south poles, respectively.

By differentiating $\theta_{1,p}$ with respect to time, we see that $\dot{\theta}_{1,p}$ is infinite at the two singular points

\[
\dot{\theta}_{1,p} = \frac{\partial \theta_{1,p}}{\partial w a_x} \frac{d w a_x}{dt} + \frac{\partial \theta_{1,p}}{\partial w a_y} \frac{d w a_y}{dt} = \frac{w a_x \frac{d w a_y}{dt} - w a_y \frac{d w a_x}{dt}}{(w a_x)^2 + (w a_y)^2} \tag{16}\]

4.2 Secondary Solution

The secondary solution uses the fact that when $w a$ is directed along the polar axis, the set of joint-screws given by $\{\hat{S}_2, \hat{S}_3, \hat{S}_4\}$ forms a third order screw system.

Since, in the primary mode, the twist-velocity of $\hat{S}_1$, $\dot{\theta}_{1,p}$, becomes infinite when $\hat{S}_1$ and $\hat{S}_4$ become dependent, our desire is to to force $\dot{\theta}_{1,p}$ to zero before this dependency occurs. Since the magnitude of the cross-product of $S_1$ with $S_4$ approaches zero as $\hat{S}_1$ and $\hat{S}_4$ become dependent, multiplying $\dot{\theta}_{1,p}$ by this magnitude yields a secondary twist-velocity for $\hat{S}_1$

\[
\dot{\theta}_{1,s} = \dot{\theta}_{1,p} |S_1 \times S_4| = \dot{\theta}_{1,p} \sqrt{(w a_x)^2 + (w a_y)^2} \tag{17}\]
However, since $\dot{\theta}_{1,p}$ is infinite when $\omega a$ is directed along the polar axis, a wiser choice would be to use the maximum value of $\dot{\theta}_1, \dot{\theta}_{1,m}$, in place of $\dot{\theta}_{1,p}$.

Moreover, since

$$\sin \theta_{2,p} = \sqrt{(\omega a_x)^2 + (\omega a_y)^2}$$

Equation (17) can be rewritten as

$$\dot{\theta}_{1,s} = \dot{\theta}_{1,m} \sin \theta_{2,p}$$

(19)

Hence,

$$\theta_{1,s}(t) = \theta_{1,s}(t - t_s) + \dot{\theta}_{1,s} t_s$$

(20)

where, $t$ is the time and $t_s$ is the sample period. We note that $\theta_{1,s}$ is a non-conservative function, as it is now both time and path dependent.

From Long and Paul [1988] $\theta_{2,s}$ and $\theta_{3,s}$ can be written as functions of $\theta_{1,s}$

$$\theta_{2,s} = \tan^{-1} \left\{ \frac{\omega a_x C \theta_{1,s} + \omega a_y S \theta_{1,s}}{\omega a_z} \right\}$$

(21)

$$\theta_{3,s} = \tan^{-1} \left\{ \frac{\omega a_x S \theta_{1,s} - \omega a_y C \theta_{1,s}}{\sqrt{(\omega a_x C \theta_{1,s} + \omega a_y S \theta_{1,s})^2 + (\omega a_z)^2}} \right\}$$

(22)

And, $\theta_{4,s}$ has the same form as $\theta_{4,p}$ except that $\theta_{1,p}$ is replaced by $\theta_{1,s}$

$$\theta_{4,s} = \tan^{-1} \left\{ \frac{-\omega a_x S \theta_{1,s} + \omega a_y C \theta_{1,s}}{-\omega a_x S \theta_{1,s} + \omega a_y C \theta_{1,s}} \right\}$$

(23)

Of pedagogic value is the analogy between the expression for $\dot{\theta}_{1,s}$ and the quaternion representation of spatial rotations invented by Hamilton [1969]. Using Hamilton’s quaternion concept, a rotation of the vector $v$, about an arbitrary axis $k$ is given by

$$v' = q \ast v \ast q^{-1}$$

(24)

where, $v'$ is the final orientation of $v$, $q$ is the unit quaternion given by

$$q = \cos \frac{\phi}{2} + \sin \frac{\phi}{2} k,$$
\[ e \text{ represents quaternion multiplication, and } \phi \text{ is the rotation angle.} \]

When the rotation \( \phi \) is zero, we expect Equation (24) to yield \( v' = v \). Interestingly, as the rotation axis \( k \) becomes undefined when \( \phi \rightarrow 0 \), the factor of \( \sin \frac{\phi}{2} \) conveniently annihilates this term. This is analogous with the expression for \( \dot{\theta}_{1,s} \), since the \( \sin \theta_{2,p} \) annihilates the constant \( \dot{\theta}_{1,m} \).

Because of the analogy between the quaternion representation for rotations and the expression for \( \dot{\theta}_{1,s} \), we have named this 4-\( R \) wrist the Hamilton wrist. In comparison, the 3-\( R \) wrist is also known as the Euler wrist, since spatial orientations of the end-effector can be defined by the three Euler angles \( \psi - \theta - \psi \).

### 4.3 Algorithm Summary

The inverse kinematic algorithm for the 4-\( R \) wrist is as follows: (Note: the \( \tan^{-1} \) function has been replaced with the \( \text{atan2} \) function used in most programming languages)

1. Begin in primary mode with the four joint values given by

\[
\begin{align*}
\theta_{1,p} &= \text{atan2} \left( \frac{w a_z}{w a_y} \right) \\
\theta_{2,p} &= \text{atan2} \left( \frac{\sqrt{(w a_x)^2 + (w a_y)^2}}{w a_z} \right) \\
\theta_{3,p} &= 0 \\
\theta_{4,p} &= \text{atan2} \left( \frac{w n_x S \theta_{1,p} + w n_y C \theta_{1,p}}{-w \theta_x S \theta_{1,p} + w \theta_y C \theta_{1,p}} \right)
\end{align*}
\]

2. If

\[
\left| \frac{\theta_{1,p}(t) - \theta_{1,p}(t-t_s)}{t_s} \right| \geq \dot{\theta}_{1,m} \sin \theta_{2,p}
\]

then go into secondary mode with the joint values given by

\[
\begin{align*}
\theta_{1,s}(t) &= \theta_{1,s}(t-t_s) \pm \dot{\theta}_{1,m} \sin \theta_{2,p} t_s \\
\theta_{2,s} &= \text{atan2} \left( \frac{-w a_x C \theta_{1,s} + w a_y S \theta_{1,s}}{w a_z} \right) \\
\theta_{3,s} &= \text{atan2} \left( \frac{-w a_x S \theta_{1,s} - w a_y C \theta_{1,s}}{\sqrt{(w a_x C \theta_{1,s} + w a_y S \theta_{1,s})^2 + (w a_z)^2}} \right) \\
\theta_{4,s} &= \text{atan2} \left( \frac{w n_x S \theta_{1,s} + w n_y C \theta_{1,s}}{-w \theta_x S \theta_{1,s} + w \theta_y C \theta_{1,s}} \right)
\end{align*}
\]

13
3. Monitor $\theta_{1,p}$ in the background.

4. If

$$|\theta_{1,p} - \theta_{1,s}| > \tau$$

then

$$\dot{\theta}_{1,s} = -\dot{\theta}_{1,s}$$

5. Return to primary mode when

$$\frac{\dot{\theta}_{1,p}(t) - \dot{\theta}_{1,p}(t-t_0)}{t_s} \leq \dot{\theta}_{1,m} \sin \theta_{2,p} \quad \text{and} \quad \theta_{1,s} = \theta_{1,p}$$

6. Repeat steps 1-5

The algorithm presented here is simple and straightforward, and all that remains is to determine the maximum Cartesian twist-velocity, which is discussed in Section (6).

5 Wrist Jacobian and Twist-Velocities

The wrist Jacobian is formed in the $O_3$ coordinate system, as the resulting elements in the formulation are simple. Formation of the wrist Jacobian is achieved by resolving the set of joint-screws $\{S_1, S_2, S_3, S_4\}$ in the $O_3$ system. If this resolution is denoted by a superscript 3 in front the joint-screw, then

$$\begin{bmatrix}
\Omega_x \\
\Omega_y \\
\Omega_z
\end{bmatrix} =
\begin{bmatrix}
S_1 & S_2 & S_3 & S_4
\end{bmatrix}
\begin{bmatrix}
\dot{\theta}_1 \\
\dot{\theta}_2 \\
\dot{\theta}_3 \\
\dot{\theta}_4
\end{bmatrix}
= C\theta_3 S\theta_3 C\theta_3 0 0
\begin{bmatrix}
\dot{\theta}_1 \\
\dot{\theta}_2 \\
\dot{\theta}_3 \\
\dot{\theta}_4
\end{bmatrix}
\text{(26)}$$

where the wrist Jacobian, $J_w$, is given by

$$J_w =
\begin{bmatrix}
C\theta_2 S\theta_3 & C\theta_3 & 0 0 \\
S\theta_2 & 0 & -1 0 \\
C\theta_2 C\theta_3 & -S\theta_3 & 0 1
\end{bmatrix}
\text{(27)}$$
5.1 Twist-Velocities: Primary Solution

Since $\theta_{3,p} = 0$ and $\dot{\theta}_{3,p} = 0$ in the primary solution, from Equation (26) the relationship between the twist-velocities and the Cartesian rate becomes

$$3 \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ S\theta_{2,p} & 0 & 0 \\ C\theta_{2,p} & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_{1,p} \\ \dot{\theta}_{2,p} \\ \dot{\theta}_{4,p} \end{bmatrix}$$

(28)

Solving Equation (28) for $\dot{\theta}_{1,p}$, $\dot{\theta}_{2,p}$, and $\dot{\theta}_{4,p}$ we obtain

$$\dot{\theta}_{1,p} = \frac{3\Omega_y}{S\theta_{2,p}}$$

(29)

$$\dot{\theta}_{2,p} = 3\Omega_x$$

(30)

$$\dot{\theta}_{3,p} = 0$$

(31)

$$\dot{\theta}_{4,p} = 3\Omega_z - C\theta_{2,p}\dot{\theta}_{1,p}$$

(32)

From the above equations, we see that the primary solution is singular when $S\theta_{2,p} = 0$, which corresponds to $\theta_{2,p} = 0, \pm \pi$. As discussed previously, this singularity occurs when the blade lies along the polar axis. Also note that as $S\theta_{2,p}$ approaches zero, only $\dot{\theta}_{1,s}$ and $\dot{\theta}_{4,s}$ become unbounded while $\dot{\theta}_{2,s}$ and $\dot{\theta}_{3,s}$ remain well-defined.

5.2 Twist-Velocities: Secondary Solution

To show that $\dot{\theta}_{2,s}$, $\dot{\theta}_{3,s}$, and $\dot{\theta}_{4,s}$ are bounded when $\dot{\theta}_{1,s}$ is bounded, we rearrange Equation (26) as

$$3 \begin{bmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{bmatrix} - \dot{\theta}_{1,s} \begin{bmatrix} C\theta_{2,s}S\theta_{3,s} \\ S\theta_{2,s} \\ C\theta_{2,s}C\theta_{3,s} \end{bmatrix} = \begin{bmatrix} C\theta_{3,s} & 0 & 0 \\ 0 & -1 & 0 \\ -S\theta_{3,s} & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\theta}_{2,s} \\ \dot{\theta}_{3,s} \\ \dot{\theta}_{4,s} \end{bmatrix}$$

(33)

Hence, $\dot{\theta}_{2,s}$, $\dot{\theta}_{3,s}$, and $\dot{\theta}_{4,s}$ are given by

$$\dot{\theta}_{2,s} = \frac{3\Omega_z - \dot{\theta}_{1,s}C\theta_{2,s}S\theta_{3,s}}{C\theta_{3,s}}$$

(34)

$$\dot{\theta}_{3,s} = \dot{\theta}_{1,s}S\theta_{2,s} - 3\Omega_y$$

(35)

$$\dot{\theta}_{4,s} = 3\Omega_z - \dot{\theta}_{1,s}C\theta_{2,s}C\theta_{3,s} + \dot{\theta}_{2,s}S\theta_{3,s}$$

(36)
From the above equations, we see that $\dot{\theta}_{2,s}$, $\dot{\theta}_{3,s}$, and $\dot{\theta}_{4,s}$ are bounded provided that $\dot{\theta}_{1,s}$ is bounded and that $C\theta_{3,s} \neq 0$, or that $\theta_{3,s} \neq \pm \frac{\pi}{2}$. Since $\dot{\theta}_{1,s}$ is given by $\dot{\theta}_{1,m} \sin \theta_{2,s}$, $\dot{\theta}_{1,s}$ is always bounded, however, without auxiliary restrictions on the Cartesian twist-velocity, there is no insurance that $C\theta_{3,s}$ will never be zero. Hence, the singularity in the secondary mode occurs when $C\theta_{3,s} = 0$; in the next section, we discuss how to prevent this from occurring.

6 Maximum Cartesian Twist-Velocity

An upper-limit for the Cartesian twist-velocity is given naturally by the familiar triangle inequality rule of vector addition (Noble and Daniel [1969])

$$\omega_u \leq |\dot{\theta}_{1,m}S_1| + |\dot{\theta}_{2,m}S_2| + |\dot{\theta}_{3,m}S_3| + |\dot{\theta}_{4,m}S_3| = \dot{\theta}_{2,m} + \dot{\theta}_{3,m} + \dot{\theta}_{4,m}$$

(37)

where $\omega_u$ is the upper-limit for the Cartesian twist-velocity, and $\dot{\theta}_{1,m}$, $\dot{\theta}_{2,m}$, $\dot{\theta}_{3,m}$, and $\dot{\theta}_{4,m}$ are maximum joint rates.

Although an upper-limit for the Cartesian twist-velocity is given by the triangle inequality rule, an auxiliary restriction must be imposed to prevent the occurrence of a singularity in the secondary mode.

6.1 Preventing the Singularity in Secondary Mode

Recall that when $\theta_1$ is given by the primary solution, $C\theta_3 = 1$, however, in general, when $\theta_1$ is given by the secondary solution, $C\theta_3 \neq 1$. From the wrist geometry, we notice that $C\theta_3 = 0$ only when $w_a$ lies in the equatorial plane. Hence, our objective is to force $\theta_{1,s}$ to $\theta_{1,p}$ as $w_a$ approaches the equatorial plane.

Let the difference between $\theta_{1,s}$ and $\theta_{1,p}$ be given by $\theta_{1,d}$

$$\theta_{1,d} = \theta_{1,s} - \theta_{1,p}$$

(38)

In deriving a maximum value for the Cartesian twist-velocity, we use the following scenario. Let the approach vector, $w_a$, initially lie along the polar axis, where $w_a$ points in
Figure 4: Geometry for Scenario

either the north or south directions. Further, let the difference between \( \theta_{1,s} \) and \( \theta_{1,p}, \theta_{1,d} \) have its maximum value of \( \pi \). Now, rotate \( w_a \) about an axis \( k \) in the equatorial plane such that after the rotation \( w_a \) also lies in the equatorial plane; this corresponds to a rotation of \( \pi/2 \) about the axis \( k \).

Our objective is to determine a Cartesian twist-velocity such \( \theta_{1,d} \) reaches zero before \( w_a \) reaches the equatorial plane. Obviously, if the Cartesian twist-velocity is too large, \( \theta_{1,s} \) will not have enough time to catch up with \( \theta_{1,p} \) before \( w_a \) reaches the equatorial plane, and the possibility of \( C\theta_{3,s} = 0 \) will occur.

Recall that in the secondary mode that \( \dot{\theta}_1 \) is given by

\[
\frac{d\theta_1}{dt} = \dot{\theta}_{1,s} = \dot{\theta}_{1,m} \sin \theta_{2,p} \tag{39}
\]

From Figure (4) we see that the angle between the polar axis and \( w_a \) is given by \( \phi \), which is equal to both \( \omega t \) and \( \phi_{2,p} \). Hence, Equation (39) can be rewritten as

\[
\frac{d\theta_1}{dt} = \dot{\theta}_{1,m} \sin \omega t = \dot{\theta}_{1,m} \sin \phi \tag{40}
\]

Assuming a maximum value for the Cartesian twist-velocity, \( \omega_m \), multiplying Equa-
\begin{align*}
\int_{\theta_{1,0}}^{\theta_{1,p}} d\theta_1 &= \int_0^t \dot{\theta}_{1,m} \sin \omega_m t \, dt = \int_0^\phi \frac{\dot{\theta}_{1,m}}{\omega_m} \sin \phi \, d\phi \\
\theta_{1,d} &= \pi = \frac{\dot{\theta}_{1,m}}{\omega_m} (1 - \cos \omega_m t) = \frac{\dot{\theta}_{1,m}}{\omega_m} (1 - \cos \phi)
\end{align*}

The above equation can be rearranged to solve for $\omega_m$

$$
\omega_m = \frac{\dot{\theta}_{1,m}}{\pi} (1 - \cos \phi)
$$

Hence, when a rotation through the polar axis is required, Equation (43) gives a maximum value for the Cartesian twist-velocity for a given $\dot{\theta}_{1,m}$ and $\phi$.

### 7 Simulations

With suitable modifications, the solution for the 4-$R$ spherical wrist can be implemented on a Puma 260 robot manipulator. A fourth joint can be attached to the Puma's original 3-$R$ spherical wrist, as shown in Figure (5); the design of the fourth joint is based on the original design of Fisher [1984]. The controller for the wrist is a parallel architecture Robot Force/Motion Server as described by Paul and Zhang [1986].
Figures (6) - (12) show results of simulations with the 4-R wrist. In Figure (6), using the $z-y-z$ convention for expressing Euler angles, the initial orientation of $\omega a$ is given by $\phi_i = 0$, $\theta = -60$, and $\psi_i = 0$; the final orientation of $\omega a$ is achieved by rotating $\omega a$ 120 degrees about the axis $k = (0,1,0)$ ($\phi_f = 0, \theta_f = 60, \psi_f = 0$). The Cartesian twist-velocity in this simulation is given by Equation (43) with $\phi = 40$ degrees

$$\omega = \frac{\dot{\theta}_{1,m}}{\pi}(1 - \cos 40)$$

(44)

When $\omega a$ goes through the north pole, an instantaneous change of 180 degrees is required by $\theta_{1,p}$ and $\theta_{4,p}$. Physically, since this change is impossible, the 4-R wrist must go into secondary mode, where the "redundant" joint, joint 3, is activated to insure that the desired Cartesian trajectory is followed.

In each of Figures (7) - (12), $\omega a$ is at a fixed angle from the polar-axis, these being 0, 5, 20, 45, 60, and 90 degrees, respectively. With this fixed angle, $\omega a$ is then rotated around the polar-axis twice. The Cartesian twist-velocity in each of these simulations is equal to the maximum twist-velocity of joint 1, or

$$\omega = \dot{\theta}_{1,m}$$

(45)

The simulations in Figures (7) - (12) show the transition from the secondary mode to the primary mode for the given Cartesian twist-velocity. When the angle between the polar-axis and $\omega a$ increases, we see that $\dot{\theta}_{1,s}$ increases until it is equal to $\dot{\theta}_{1,p}$. We note the transition from the secondary mode to the primary mode occurs sooner if the Cartesian twist-velocity is less than the maximum twist-velocity of joint 1.

8 Conclusion

In this paper we have endeavored to present a novel inverse kinematic algorithm for a four-revolute-joint spherical wrist which provides for a complete set of spatial orientations, is non-singular, and is simple to implement with existing technology. In the process of presenting
our algorithm, we have developed the fundamentals of differential motion with n-revolute-point spherical wrists. We have also shown that when the wrist's kinematic geometry is used, in conjunction with rate-control of one of the terminal joints, that a non-conservative and continuous inverse kinematic algorithm can be obtained. The method outlined here has been applied to a 4-R arm with encouraging results. Our goal is to combine the 4-R spherical wrist with the 4-R arm to develop an 8-R manipulator that is free of internal singularities.

References


![Figure 6: Rotation Through the North Pole: \( \phi_1, \theta_1, \psi_1 = (0, -60, 0), \phi_f, \theta_f, \psi_f = (0, 60, 0) \).]
Figure 7: 720 Degree Rotation About the Polar Axis: $\phi_i \mathbf{\cdot} \theta_i \mathbf{\cdot} \psi_i = \phi_f \mathbf{\cdot} \theta_f \mathbf{\cdot} \psi_f = (0, 0, 0)$.

Figure 8: 720 Degree Rotation About the Polar Axis: $\phi_i \mathbf{\cdot} \theta_i \mathbf{\cdot} \psi_i = \phi_f \mathbf{\cdot} \theta_f \mathbf{\cdot} \psi_f = (0, 5, 0)$.

Figure 9: 720 Degree Rotation About the Polar Axis: $\phi_i \mathbf{\cdot} \theta_i \mathbf{\cdot} \psi_i = \phi_f \mathbf{\cdot} \theta_f \mathbf{\cdot} \psi_f = (0, 20, 0)$. 

22
Figure 10: 720 Degree Rotation About the Polar Axis: $\phi_i - \theta_i - \psi_i = \phi_f - \theta_f - \psi_f = (0, 45, 0)$.

Figure 11: 720 Degree Rotation About the Polar Axis: $\phi_i - \theta_i - \psi_i = \phi_f - \theta_f - \psi_f = (0, 60, 0)$.

Figure 12: 720 Degree Rotation About the Polar Axis: $\phi_i - \theta_i - \psi_i = \phi_f - \theta_f - \psi_f = (0, 90, 0)$.
A.10 Control of a Seven Joint Manipulator
A Simple Control Method for A Redundant Robot with Seven Joints

Yehong Zhang
Philip Lee

Computer and Information Science Department
University of Pennsylvania
Philadelphia, PA 19104

ABSTRACT

Most redundancy control methods require the computation of the pseudoinverse of the Jacobian matrix. The large computation load makes the real time control of a redundant manipulator impossible. This paper presents a redundancy control algorithm for avoiding singularities of the 6 DOF manipulator. It is based on the partitioning of the Jacobian matrix. The formulation is computationally simple since it only requires the inversion of a 6x6 matrix. The control is applied to a seven degree of freedom robot manipulator with a 4 joint spherical wrist built at the GRASP Lab. Two types of simulation results are presented. The first plots trajectories of the manipulator and the second is a video tape using graphical packages. Both simulation results confirm that the method can be used successfully to solve the singularity problem of the 6 DOF manipulator.

1This paper is based upon work supported by the National Science Foundation under Grant No. ECS-11879. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.
1 Introduction

Redundancy was proposed to give more degrees of freedom and flexibility to the manipulator. With this redundancy, tasks such as singularity avoidance and obstacle avoidance can be solved[5]. Besides these, redundancy can be utilized to solve a wider range of problems such as avoiding joint limits and minimizing joint torques.[7] The central idea is to manipulate the extra degrees of freedom to meet the constraints imposed such as singularity avoidance. In designing the redundant arm, various designs have been presented. For a PUMA type of robot, there are essentially 3 different types of singularities, namely the shoulder, the elbow and the wrist. Hollerbach proposed to avoid all these singularities by adding another joint to the shoulder[6]. By realizing that the wrist singularity is the most common type, Paul, Fisher[3] and Long[10] proposed a four jointed wrist. It is this wrist that was built at the Grasp Lab of the University of Pennsylvania and it’s a PUMA260 with such a wrist that is controlled in our research. In this paper, we shall present a simple singularity resolution method using this arm. The algorithm is tested using two types of simulations, regular and graphical. The results confirm the correctness of the proposed method.

2 Problems of Non-redundant Manipulators

2.1 Position and Force Control at Singular Points

The control of a manipulator involves the transformation between two spaces, the 6 dimensional joint space and the 6 dimensional Cartesian space for position and orientation[4]. The mapping between the two spaces is highly non-linear. A first order linear approximation is as follows:

\[ dx = Jd\theta \]  

(1)

Where \( J \) is the Jacobian of the mapping. Another important matrix relationship relating the Cartesian forces and joint torques also involves the Jacobian \( J \) and is given below.

\[ \tau = J^TF \]  

(2)

Where \( \tau \) is the joint torques and \( F \) the Cartesian forces at the end effector. Because of the non-linear nature of the mapping, there are degenerate
points in the configuration space. Mathematically these singularity points are located in places where $\text{det}(J) = 0$.\cite{17} In actuality, the neighborhood of a singularity forms a degenerate configuration space\cite{11}. So when the manipulator moves through this neighborhood, the inverse mapping breaks down. Equation 1 implies that finite Cartesian rates will cause nearly infinite joint rates at these singularity points. Equation 2 implies that finite joint torques will result in infinite Cartesian end effector forces. Thus, at these singularity points, position and force control breaks down. Paul and Stevenson proved the existence of singularities for a wrist with 3 joints and proposed that a manipulator with 6 joints is really only a five degree of freedom mechanism\cite{11}. Hollerbach proved the case for a general 6 d.o.f robot\cite{6}. Though the ability to deal with singularity is important, but it should not be the primary concern of a redundant robot manipulator. A redundant manipulator is attractive because it has the potential to deal with problems other than singularity avoidance.

2.2 Constraint Induced Problems

Besides singularity points, there are other problems with the general 6 d.o.f manipulators caused by physical constraints of the manipulator. These constraints include joint torque limits, joint limits and other form of geometric constraints such as obstacles. A manipulator moving in a space with many obstacles not only has to find a end effector path that is obstacle free but also has to make sure that the links stay clear of the obstacles. This requires much more flexibility and manipulability than is currently available in a 6 d.o.f manipulator. And the more constraints there are, the more degrees of freedom are demanded of the manipulator. These constraints, together with the singularity constraints limit the work space of the manipulator. To have a manipulator work in a realistic environment in an intelligent manner requires that it overcome these constraints.
3 Current Redundancy Control Methods

3.1 Existing Methods

Current redundancy control strategies are Jacobian based and the mathematical formulation is the linearization of the non-linear transformation between the Cartesian space and manipulator joint space.[9]. With a redundant arm, Equation 1 assumes the same form except that now $d\theta$ is 7x1 and $J$ is 6x7. By inverting this equation, we obtain:

$$d\theta = J^+ dx + (I - J^+J)k$$

(3)

Here $J^+$ is the pseudoinverse of $J$. $(I - JJ^+)$ represents the null space of $J$. $k$ is an arbitrary 7x1 vector since $(I - J^+J)k$ gets mapped to zero under $J$. It is this arbitrary vector $k$ that allows us to utilize the redundancy of the robot[8]. Most of the current approaches are focused on finding the right $k$ to avoid singularity points and stay away from physical limits of the arm.

Yoshikawa first proposed the division of tasks into subtasks of different priorities to avoid singularity and obstacles[5]. Hollerbach choose $k$ such that joint torques are minimized[7]. Others have tried the minimum norm solution and avoidance of joint limits. The common characteristic of all these methods is that they are all different ways of selecting $k$ to meet a particular criterion. And they all involve very heavy computational loads. The computational load mainly comes from two parts, the computation of the pseudoinverse and the computation of the $k$ vector. The large computational load makes the real time control of a redundant manipulator very difficult. There are other methods under research which achieve redundancy control with a great deal less computation than the Jacobian based methods. Paul and Webster proposed a heuristic approach where the psedoinverses of certain locations are named and stored and control in space is achieved by interpolating between the named positions.[16] The advantage is that computational load is reduced greatly and the robot is controlled in a very intelligent manner. Problems remain such as the organization of the heuristic rules, the number of positions needed to cover the whole workspace and the area of convergence for each named psedoinverse. Fisher, Long and Paul also developed a kinematic solution, without using Jacobian and its inverse, to control a robot arm with a four joint wrist[10].
In the last two sections of this chapter, we will propose a new scheme for redundancy control. The method is a simple and straightforward approach that involves only the inversion of a 6 by 6 matrix. It works particularly well for a manipulator with 7 joints as will be shown by the simulation results. But before moving on to a detailed discussion of the methods, a topological analysis is conducted to reach some general conclusions with regards to redundancy control and singularity.

3.2 Topological Insights on Manipulator Singularities

Robot manipulation centers around the forward and inverse mappings between two different spaces, namely the n dimensional joint space and the 6 d.o.f Cartesian space. By recognizing the fact that the joint space is the product of n circles and thus represents a n dimensional torus and the Cartesian space is the product of $R^3$ and $S^2$, many general useful conclusions[4] can be derived regardless the value of n. The first thing to recognize is that the torus and the sphere are totally different topologically. The fact that their fundamental groups are different shows that they are different manifold. This fact has profound implications in the choices of mapping between these two spaces. Most important of all, it says that there is no global homeomorphism between these two manifolds. This essentially means the there is no global mapping between the two manifolds which is continuous, bijective and whose inverse mapping is also continuous. Note that it doesn't say anything about local mappings. As a matter of fact, local mappings could be very well behaved. Let's take the 6 linked robot for an example. Let's suppose Jacobian matrix is used for the mapping between the two manifolds. When the mapping is good, it is continuous and bijective and so is its inverse and so Jacobian control can be used successfully. But as predicted above, there are bound to be points where the mapping breaks down. In this case, the mapping is not bijective anymore but instead is one to many. This is what we call kinematic singularity. Thus, the existence of singularities can be predicted by using topological arguments as well as by linear algebra arguments. Another implication of the topological analysis is that no matter how many joints there are on a robot, singularities will always be there. As a matter of fact, the more joints there are, the larger the dimension of the singularity space[2]. Yet another implication of the topological analysis is that a global homeomorphic inverse kinematic mapping does not exist. This does
not mean as some believe that there is no continuous global inverse kinematic mapping. It simply means that a globally continuous and bijective inverse kinematic mapping does not exist. This implies that the conservation of path repeatability can not coexist with the existence of a globally continuous inverse kinematic mapping. In the case of a four joint wrist, path repeatability might not be as important as in the case of a four joint positioning arm and globally continuous inverse kinematic mappings can be found quite easily.

4 A Redundancy Control Method for Singularity Avoidance

From the previous discussion, the computational load for singularity avoidance using a redundant arm is quite large. We believe a highly redundant arm should be able to deal with singularities naturally and easily. This serves as the motivation for the following approach to redundancy control.

4.1 Partitioned Jacobian Approach

We will assume that our task is the general positioning and orienting of the end effector of the arm and the number of joints n is greater than 6. The differential relationship is given as follows.

\[ dx = Jd\theta \] (4)

where \( J \) is a 6xn matrix and \( d\theta \) is a nx1 column matrix. Now consider all the possible 6x6 submatrices formed by combining different columns of the Jacobian matrix. Suppose there is a way to select one of the 6x6 submatrices which is reasonably far away from any singularity, then we can rearrange columns of the Jacobian matrix and the corresponding elements of the \( d\theta \) matrix. The above equation can then be written as

\[ dx = [J_6, J_{n-6}][d\theta_p, ..., d\theta_{p+5}, d\theta_{p+6}, ..., d\theta_{p+n-1}]^T \] (5)

Here \( J_6 \) is the selected submatrix and \( J_{n-6} \) is the rest of the Jacobian matrix. Because of the rearrangement in the Jacobian matrix, the \( d\theta \) column would have to change accordingly as well and subscript \( p \) represents the
particular permutation caused by this rearrangement of the columns of the Jacobian matrix. The above equation can be written in another way.

\[ dx - J_{n-6}[d\theta_{p+6}, \ldots, d\theta_{p+n-1}]^T = J_6[d\theta_p, \ldots, d\theta_{p+5}]^T \]  \hspace{1cm} (6)

Since \( J_6 \) is a non-singular square matrix, we will denote its inverse by \( J_6^{-1} \). We have

\[ [d\theta_p, \ldots, d\theta_{p+5}]^T = J_6^{-1}(dx - J_{n-6}[d\theta_{p+6}, \ldots, d\theta_{p+n-1}]^T) \]  \hspace{1cm} (7)

By imposing up to \( n - 6 \) conditions on the elements of \([d\theta_{p+6}, \ldots, d\theta_{p+n-1}]^T\), we can solve for the elements of \([d\theta_p, \ldots, d\theta_{p+5}]^T\). This will allow us to calculate the joint trajectories for all the joints. Note that the conditions imposed on the joint rates have to be such that the joint trajectories are continuous.

To better understand the method, let’s take a 7 jointed robot arm as an example. In this case the Jacobian matrix will be 6x7. The number of possible 6x6 matrices through column permutation is 7. Let’s again denote such a submatrix by \( J_6 \). Since the determinant of \( J_6J_6^T \) is a good measure of how far away \( J_6 \) is from its singularity points[17], we select a \( J_6 \) matrix by comparing all 7 determinant values and choosing the largest one. Suppose \( J_i \) is the \( i \)th column of the Jacobian matrix and is not chosen, then the equation can be written in the following fashion.

\[ dx - J_i d\theta_i = J_6 d\bar{\theta} \]  \hspace{1cm} (8)

Here \( d\bar{\theta} \) is the permuted 6x1 column matrix. A simple condition on joint \( i \) rate would be a constant rate motion. So set \( d\theta_i \) to a constant velocity \( C_0 \) and invert \( J_6 \). We get

\[ d\bar{\theta} = J_6^{-1}(dx - J_i C_0) \]  \hspace{1cm} (9)

4.2 Experimentations

Many experiments are conducted using the above approach. The redundant manipulator is a PUMA260 with an extra joint attached to the wrist.
4.2.1 Kinematic Analysis

The kinematic diagram of the redundant manipulator is given below in Fig.1. Given the above kinematic diagram, the joint and link parameters are given by Table1[13]. The homogeneous transforms for the link coordinates are given by the usual Denavit and Hartenberg matrix. Knowing the homogeneous transforms, the Jacobian matrix relating the end effector Cartesian velocity with the joint velocities is found to be as follows[12]:

\[
J = \begin{pmatrix}
J_{11} & 0 \\
J_{21} & J_{22}
\end{pmatrix}
\]

where \( J_{11}, J_{21}, J_{22} \) are 3x3 and 3x4 submatrices of the Jacobian.

Note that the top right 3 by 4 sub-matrix is a null matrix. This is to be expected since the 4 wrist link axes intersect at a single point and thus have no contribution to the Cartesian velocities along the Cartesian frame axes. This simplifies our task tremendously as will be shown below.

4.2.2 Set Point Generation

It is proven that singularity occurs when the last 4 columns become linearly dependent. So we will select 3 out of those four columns such that the 3 selected columns are linearly independent. Because of the null submatrix at the upper right hand corner, the problem reduces to comparing all possible 3x3 submatrices of \( J_{22} \) and choosing the one that is the best according to some criterion. Since the determinant of a matrix gives a good measure of how far away a matrix is from singularity points, we are going to select the 3 columns which give a matrix whose determinant is the largest. Of course, this is by no means the only or the best criterion.

Let's denote the columns in the \( J_{22} \) matrix in the following way:

\[
J_{22} = (j_i \ j_j \ j_k \ j_i)
\]

where the subscripts denote the corresponding joint angles as well.

Suppose \( j_i, j_j, j_k \) is judged to be the best linearly independant subset of \( J_{22} \). Then the Jacobian equation can be written as follows:

\[
[d\theta_1, d\theta_2, d\theta_3, d\theta_i, d\theta_j, d\theta_k]^T = J_e^{-1}(dx - j_k * d\theta_k)
\] (10)
The computational load associated with the evaluation of the 3x3 determinants is small so this selection process is repeated every sampling period. The joint corresponding to the column not chosen is supposed to move with its previous velocity. This is done to ensure the continuity of the joint velocity trajectory. The simulation results are given below.

4.3 Simulation Results

4.3.1 Regular Simulation Results

The task is to move the manipulator from P1 to P2 where P2 is a singular configuration. The two positions and their Jacobian matrices are given by:

\[
P1 = \begin{pmatrix}
-0.29 & -0.30 & -0.91 & 5.00 \\
0.51 & -0.85 & 0.12 & -4.00 \\
-0.81 & -0.43 & 0.40 & 3.00 \\
0.00 & 0.00 & 0.00 & 1.00
\end{pmatrix}.
\]

\[
Jac1 = \begin{pmatrix}
1.37 & -3.17 & 0.88 & 0.00 & 0.00 & 0.00 \\
-5.47 & -1.22 & 0.91 & 0.00 & 0.00 & 0.00 \\
-3.01 & 4.74 & 2.72 & 0.00 & 0.00 & 0.00 \\
-0.81 & -0.51 & -0.51 & -0.81 & -0.59 & -0.72 \\
-0.43 & 0.85 & 0.85 & -0.43 & 0.60 & -0.70 \\
0.40 & -0.12 & -0.12 & 0.40 & -0.54 & 0.00
\end{pmatrix}.
\]

\[
P2 = \begin{pmatrix}
1.00 & 0.00 & 0.00 & 5.00 \\
0.00 & 0.00 & 1.00 & -4.00 \\
0.00 & -1.00 & 0.00 & 3.00 \\
0.00 & 0.00 & 0.00 & 1.00
\end{pmatrix}.
\]

\[
Jac2 = \begin{pmatrix}
4.02 & -3.01 & -3.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.03 & -4.99 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 \\
4.99 & 0.04 & 0.01 & 0.00 & 0.00 & 0.00 & 0.00 \\
-0.00 & -0.00 & -0.00 & -0.00 & 0.00 & -0.87 & 0.00 \\
-1.00 & -0.01 & -0.01 & -1.00 & -0.01 & -0.49 & 0.00 \\
0.01 & -1.00 & -1.00 & 0.01 & -1.00 & 0.00 & 1.00
\end{pmatrix}.
\]

The second Jacobian matrix indicates that the manipulator is at a singular configuration since the fifth and seven rows are almost parallel to each
other. Using the algorithm outlined in the previous section, the joint trajectories are obtained and presented in Fig. 2. Note the joint trajectories for the first three positioning joints remain zero since the $p$ vector for the beginning and ending transforms are the same.

### 4.3.2 Graphical Simulation Using an Iris Workstation

Often redundancy control methods are confined to paper because a redundant manipulator does not exist. The reason for this is that it may be difficult to justify the necessary spending required to build the links. However, the joint trajectories of a seven-degree-of freedom puma can be simulated using graphical software tools and therefore the viability of a control method can be tested.

For our purpose, a highly interactive, mouse driven program called Jack[1] is used. Jack is loaded on an IRIS, a graphics workstation from Silicon Graphics, which can perform near real time update of an image. Jack provides a flexible and general user interface for manipulating complex articulated structures. Jack is not a complete system in itself, but a toolkit of routines and operators for displaying and manipulating geometric figures. Jack provides input and control for applications involving lighting and image rendering, figure modeling, dynamic analysis, and keyframe and constraint-based animation and simulation.[15]

The geometric objects used by Jack are represented by a system called Peabody. Peabody is the name of a language for describing the figure as well as the internal data structure for representing them. Peabody, which is a graph-structured representation for articulated geometric objects, represents figures composed of segments connected by joints. Segments are connected by joints at two attachment points call sites. Peabody objects may be described in text files using the peabody language, which resembles a data structure definitions in a programming language. [14] Below is part of the file that was used to describe the Puma.

```plaintext
figure puma{
  segment link0 {
    site dist ->location = xyz(90deg,90deg,0) * trans(1,0,.5);
    ....
  }
}
```
segment link1 {
    site prox ->location = xyz(0,0,0) * trans(0,0,0);
    ...
}

joint spin_joint {
    connect link0.dist to link1.prox;
    ...
}

Jack is much like an editor for constructing and manipulating peabody objects. It provides a means of manipulating the peabody files without having to edit them explicitly. Most of the operations in Jack uses the mouse, both to pick commands from menus and to specify geometric transformations. Parameters and values may also be entered directly from the keyboard. Whenever possible, the syntax of these values follows the peabody language, reinforcing the idea that Jack is an interactive peabody editor. [1]

The video, which accompanies this paper, was created by basic animation techniques which are available in Jack. Joint positions were generated by the control scheme described in the paper’s earlier sections and then were played back with the animation facilities in Jack.

Animation, in general, is generated by first creating a set of key-frames. Key-frames are a set of key joint positions which defines an entire joint motion. The joint positions between key-frames are then computed by interpolating the joint values with the key-frame’s joint values as data points. In general, as a joint motion increases in complexity, the number of key-frame to accurately produce the correct motion increases. The playback time of an animation is a parameter. This is useful since the Iris has near-real time capabilities when animating a wire-frame figure. After the Puma robot’s motion has been animated and a proper view has been selected the wire-frame figure is rendered. Rendering is the process of converting a wire-frame representation into a solid representation. A solid model offers the advantage of better visualization of the robots motion and to quickly check for possible collisions.

The trajectory given to the graphical simulator is a trajectory whose end
point is a singular configuration. The robot is shown to move smoothly into that configuration as expected.

5 Conclusion

The above method works well for a 7 DOF robot manipulator because the number of possible 6x6 submatrices are fairly small. In the case of our GRASP redundant arm, the problem was even more simplified by the fact that all four wrist joints intersect at the same point in space. That reduces the problem to evaluating only 3x3 submatrices of J22 and there are only 4 of those. In a arm with more than 7 DOF’s, the number of possible submatrices become very large and selecting a reasonable matrix becomes much more difficult. We believe that the formulation still works except that a new selection algorithm needs to be found.

Almost all current redundancy control methods become cumbersome when the number of joints of a robot manipulator increases beyond seven. Even though a seven DOF arm can be proven to avoid singularities in its workspace, it is not a very good arm simply because that is about all that it can do. So there is a need for more redundant mechanisms and the control of these mechanisms is yet to be solved.
References


<table>
<thead>
<tr>
<th>Link</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Link</td>
<td>c  a  d</td>
</tr>
<tr>
<td>1</td>
<td>90 0 0</td>
</tr>
<tr>
<td>2</td>
<td>0 a2 d2</td>
</tr>
<tr>
<td>3</td>
<td>-90 0 0</td>
</tr>
<tr>
<td>4</td>
<td>90 0 d4</td>
</tr>
<tr>
<td>5</td>
<td>-90 0 0</td>
</tr>
<tr>
<td>6</td>
<td>-90 0 0</td>
</tr>
<tr>
<td>7</td>
<td>0 0 0</td>
</tr>
</tbody>
</table>

14
A.11 Visual Servoing of Robot Manipulators
VIDEO-RATE VISUAL SERVOING FOR ROBOTS

Peter I. Corke 1
Richard P. Paul 2

1. Abstract
This paper presents some preliminary experimental results in robotic visual servoing, utilizing a newly available hardware region-growing and moment-generation unit. A Unix-based workstation in conjunction with special purpose video processing hardware has been used to visually close the robot position loop at video field rate, 60Hz.

The architecture and capabilities of the system are discussed. Performance of the closed-loop position control is investigated analytically and via step response tests, and experimental results are presented. Initial results are for 2 dimensional servoing, but extensions to 3 dimensional positioning are covered along with methods for monocular distance determination.

Finally some limitations of the approach and areas for further work are covered.

2. Introduction
This paper presents some preliminary experimental results in robotic visual servoing, utilizing an off-the-shelf hardware region-growing and moment-generation unit. This unit can process a binary scene, containing many regions, at video data rates and yields fundamental geometric parameters about each region. Such a device opens up many applications that were not previously possible due to the computational complexity involved.

The authors' aim was to develop a system capable of:

- at least frame rate (30Hz) analysis of video data
- functioning with a non-trivial scene
- determining object position as well as distance
- following a moving or stationary target
- functioning with a moving camera

A priori knowledge of the target object's shape and dimensions is assumed. A non-trivial scene is one with a complex background (not a black backdrop) and in which illumination levels may change. In this paper a distinction is made between tracking, and visual servoing. Much of the tracking literature assumes slow motion, or stop-start motion. Visual servoing however treats the camera-processing subsystem as a sensor with its own inherent dynamics, which the closed-loop position control must take into account.

There are many approaches to segmenting a complex scene so as to locate a possibly moving object. A general discussion of motion detection schemes is presented in[1]. Perhaps the simplest approach is image differencing[2] in which the difference between consecutive frames eliminates all detail from the scene except the edges of moving objects. This approach will fail if the velocity of the object is too low to differentiate motion energy from sensor noise, or if the camera is moving, and some hybrid strategy is needed to initially locate a static object.

1 Research Scientist, CSIRO Division of Manufacturing Technology, Melbourne, Australia.
2 Professor, University of Pennsylvania, Philadelphia.
A more sophisticated approach is optical flow analysis[3] which yields a dense velocity vector field from spatio-temporal changes in pixel intensities. This analysis is computationally expensive, particularly the smoothing operation[4] and requires special processing hardware to implement in real time. Other problems include interpretation of the velocity field for general camera motion with rotations[5] and sensitivity to camera noise[6]. The derived velocity field must still be segmented to separate background from foreground. A related technique is based on feature matching, where features are generally corner points[7,8], edges[9] or regions. The change in position of various feature points between scenes can be used to estimate the velocity of those points. However features such as corners or lines are a sparse representation of the scene, and it is difficult to relate those features to the objects of interest in the environment without complex world models.

Sanderson[10] discusses visual servoing of robots and proposes a control scheme based on features extracted from the scene such as face areas of polyhedra, but gives no detail of the vision processing required to obtain those features. Kabuka describes two segmentationless approaches. One[11] uses an adaptive control based on parameters derived from the entire scene, while[12] is based on Fourier phase difference between consecutive frames.

Image segmentation is only the first step towards object tracking, the next problem is to identify the position of the regions within the scene, and must be done whether the input image represents intensity, velocity or range data. One approach to object location is to use a pyramid decomposition[13] so as to reduce the complexity of the search problem. Others have used moment generation hardware to compute the centroid of an object within the scene at video data rates[14, 15, 16].

This paper discusses a region feature approach; a hardware unit performs segmentation of the intensity image to identify region features. The regions are then further investigated in software to identify the target, and tracking commands are then generated from its position. This approach is very general and works regardless of the state of motion of camera and target. The task of segmenting a complex scene in real-time is not trivial, but is nevertheless the basis of this work.

There is relatively little literature on visual servo implementations for robot position control. Many papers present only simulation results. Approaches to date could be classified as either stop-start (not dealt with here) or continuous. Hill and Park[17] describe a closed-loop position controller for a Unimate robot. They use binary image processing for speed, but also propose the use of structured lighting to reduce computational burden and to provide depth determination. Coulon and Nougaret[18] describe a digital video processing system for determining the location of one object within a processing window, and use this information for closed loop position control of an XY mechanism. They report a settling time of around 0.2s to a step demand. Kabuka[11] describes a two axis camera platform and image processor controlled using an IBM-PC/XT, and reports a minimum time of 30s to center on an object. Maklhin[19] discusses aspects of accuracy, tracking speed, and stability for a Unimate based visual servo system.

A number of related areas also require real-time visual object tracking. Gilbert[20] discusses automatic object tracking cameras used for rocket tracking. Andersson[21] describes a ping-pong playing robot that uses a real-time vision system to estimate ball trajectory for a subsequent paddle positioning planning algorithm. While the system described is not a closed-loop position controller many of the principles and problems are the same.
3. The experimental setup

The experimental setup is shown schematically in Figure 1. It comprises three major subsystems: robot control, image processing, and coordination.

3.1. Robot control

The robot control subsystem runs on a MicroVAX II connected to a Unimate Puma 560 robot and controller. Software on the MicroVAX can directly command robot joint angles using the RCI interface of RCCL[22]. The MicroVAX in this application is programmed to be a cartesian rate server; that is, it accepts cartesian velocity commands (in the tool reference frame) over an Ethernet using Unix datagram facilities. The server program consists of two asynchronous processes, one that accepts cartesian rate command packets from the network, and another that communicates robot joint angle measurements and commands with the Unimate controller via the RCI software interface. The robot servo interval is currently 14 or 28\text{ms}[22]. The manipulator’s inverse Jacobian is computed every sample period using the method of Paul and Zhang[23]. The desired robot joint angles are simply calculated by

![Experimental setup diagram](image-url)

Figure 1. Experimental setup.
\[ \theta_{d_{i+1}} = \theta_d + J^{-1}dx_{\text{client}} \]

where \( dx_{\text{client}} \) is the cartesian rate from the motion client. The control algorithm makes no use of the observed joint angle \( \theta_{obs} \) since the robot is currently moving in free space, and it is the job of the Unimate servo subsystem to achieve desired joint angles.

Cartesian rate data is transferred in Sun’s external data representation, XDR, using UDP protocol. XDR is needed since the client process, running on a Sun-3, has a different data byte orientation and floating point representation to the MicroVAX. Benchmark tests indicate that writing a command packet on the client takes 1100\( \mu \)s of elapsed time (kernel plus context switch time) using UDP, versus 2500\( \mu \)s using TCP/IP. Although UDP is an inherently unreliable protocol, and does not guarantee sequential delivery of packets, in the simple laboratory network environment these presented no problems. The server will accept datagrams from a number of different clients such as the visual tracker or a graphical teachpendant emulator.

3.2. Image processing

3.2.1. Segmentation

Robust segmentation of a scene remains one of the great problems of machine vision. Haralick[24] provides a survey of techniques applicable to static images. Unfortunately many of the algorithms proposed for segmentation are iterative and thus not suitable for real-time applications. This being the case most real-time implementations use contrived situations with dark backgrounds and simply threshold the video data. Much work has been done on automated approaches to threshold selection[25, 26].

Many suggest that incorporating edge information into the segmentation process increases robustness. Although 2D histograms of edge and intensity information have been used to extract targets from noisy FLIR\(^3\) imagery in real time[27, 28, 29], many approaches still use simple thresholding.

An adaptive approach to segmentation is currently being investigated, using the capability of this hardware region-grower to perform many trial segmentations per second. One of the principle problems is how to rate the success of a segmentation so that an automatic system may modify or adapt its parameters[30].

3.2.2. Architecture for preprocessing

A functional representation of the image processing subsystem is shown in Figure 2, and is based on Datacube\(^4\) pipeline processing modules. As mentioned above, one design aim was that the system be able to function robustly with respect to different scenes, non-black background and changing lightlevels. The architecture described has attempted to meet these requirements within the constraints of modules available.

The Datacube family of video processing modules are VMEbus boards that perform various operations on digital video data. The inter-module video data paths are patch cables installed by the user. The boards are controlled by a host computer via the VME bus. The video data paths run at 10Mpixels/s and are known as MAXBUS\(^4\). Horizontal and vertical timing is established

---

\(^3\) Forward looking infrared

---

January 11, 1989
by a separate timing bus linking all boards. All images are 512 x 512 pixels.

The two VFIR convolution modules generate horizontal and vertical Sobel gradient components

\[ I_X = G_X * I_{in} \]
\[ I_Y = G_Y * I_{in} \]

where \( G_X \) and \( G_Y \) are respectively the horizontal and vertical Sobel convolution kernels. A lookup table is used to compute the gradient magnitude, from which a gradient threshold, \( T_{edge} \), is subtracted

\[ I_{edge} = \sqrt{I_X^2 + I_Y^2} - T_{edge} \]

All arithmetic is performed in 8 bits 2’s complement. The original video stream is delayed to
synchronize it with the thresholded edge image, is offset by an intensity threshold $T_{\text{image}}$, and mapped through a lookup table which binarizes the pixels on the basis of both intensity and gradient magnitude. This provides some advantage in cases where regions of different intensity, but on the same side of the intensity threshold, separated, by distinct intensity gradients need to be resolved as separate regions.

Binary median filtering on a 3x3 neighbourhood is used to eliminate single pixel noise regions which may overwhelm the region-growing unit.

The Featuremax board is used to histogram edge and scene intensities periodically. The histogram data it computes is used by the host to select the thresholds $T_{\text{edge}}$ and $T_{\text{image}}$.

For frame-rate processing the interlaced video data must be deinterlaced using double-buffered framestores in the Framestore module. Although only frame rate processing is discussed[31], an alternative approach which has been successfully tested is to process the interlaced data as two consecutive frames of half vertical resolution. The field processing approach has the advantage of eliminating the one frame time delay involved in deinterlacing, at the expense of twice as many regions to process per second.

3.2.3. APA-512

The APA-512[31] is a hardware unit designed to accelerate the computation of area parameters of objects in a scene. It was conceived and prototyped by the CSIRO Division of Manufacturing Technology, Melbourne, Australia, in 1982-4, and is now manufactured by Vision Systems Ltd. of Adelaide, Australia. The first author was a member of the design team.

The unit has some similarities to other hardware implementations of binary image processing systems[15, 32, 16]. Andersson's unit[14] computes moments of grey-scale data so as to improve accuracy when dealing with a quickly moving object. However it has no capability to

![APA-512 block diagram](image)

Figure 3. APA-512 block diagram.
detect and generate moments for multiple regions, and multiple regions will, if present, be merged into one moment set.

The APA binarizes incoming video data and performs a single pass connectivity (simple-linkage region growing) analysis. The connectivity unit commands a bank of 8 ALUs which update the region parameters (referred to as seeds) stored in seed memory. The ALUs are implemented by custom gate arrays. The seed memory is dual ported to the host VMEbus so that seed parameters of completed regions may be read.

The APA performs very effective data reduction, reducing a 10Mpixel/s stream of grey-scale video data via a MAXBUS interface, to a stream of tokens representing objects in the scene. The host processor screens the tokens according to their parameters, and thus finds the objects of interest.

For each region the following parameters are computed:

- $\Sigma_i$, number of pixels (zeroth moment)
- $\Sigma x$, $\Sigma y$ (first moments)
- $\Sigma x^2$, $\Sigma y^2$, $\Sigma xy$ (second moments)
- minimum and maximum x and y values for the region
- perimeter length
- a perimeter point
- region color (0 or 1)
- window edge contact

From these fundamental parameters, a number of commonly used area parameters such as

- area
- centroid location
- circularity
- major and minor equivalent ellipse axis lengths
- object orientation (angle between major axis and horizontal)

may be calculated by the host processor. The perimeter point is the coordinate of one pixel on the region’s perimeter, and is used for those subsequent operations that require traversal of the perimeter. The edge contact flag, when set, indicates that the region touches the edge of the processing window and may be partially out of the image, in this case the parameters would not represent the complete object.

Perimeter is computed by a sophisticated scheme that examines a 3x3 window around each perimeter point and produces an appropriate perimeter length contribution depending upon the slope of the perimeter at that point. Experiments reveal a perimeter error of less than 2% with this scheme.

The APA-512 computes these parameters for each of up to 256 current regions within the scene. Processing of the data is done in raster scan fashion, and as the end of a region is detected the region label is placed in a queue and the host is notified by an interrupt or a pollable status flag. The host may read the region parameters and then return the region label to the APA for reuse later in the frame, thus allowing processing of more than 256 objects within one frame. This feature is essential for processing non-trivial scenes which can contain several hundred regions of which only a few are of interest. Maximum processing time is one video frame time.

An additional feature of the APA is its ability to return region hierarchy information. When a region is complete the APA may be polled to recover the labels of already completed regions which were topologically contained within that region. This makes it possible to count the
number of holes within an object, and compute the area of enclosed holes or internal perimeter.

3.3. Coordination

The visual servoing is coordinated by a Sun-3/260 workstation running Sun's version of Unix, SunOS. Much of the processing involves sending commands to the various hardware image processing modules and this must be synchronized with the video vertical blanking interval, which is only 1.2ms wide for RS-170 format video data. To achieve this reliably special device drivers are used, since a Unix process cannot guarantee the necessary response time. One driver was provided for the Datacube boards, and another was written for the APA.

The main computational burden is involved in post-processing regions detected by the APA. Small regions due to noise are screened out quickly and the region labels are returned to the APA. Larger regions are then screened according to object feature descriptions, which contain specifications for color (0 or 1), and permissible ranges for size, and shape. If a matching object is found then the manipulator's velocity in the 2D plane is set as a function of the displacement of the object's centroid from the center of the image.

4. Dynamic performance

As mentioned above, due to the high image processing rates attainable with this architecture, dynamic effects within the closed-loop system become significant.

A block diagram of the control loop is shown in Figure 4, and where \( y \) is the robot position and \( u \) is the target position. The error \( e = y - u \) is computed in pixel space. The open-loop transfer function of the frame rate system is

\[
V(z)D(z)\left(\frac{z}{z-1}\right) R(z)
\]

where the pole at \( z = 1 \) is due to the integrating effect of the cartesian rate server.

Experiments reveal that delay is the dominant dynamic effect in this system, as has been found by others[17, 19, 10]. The frame rate tracker exhibits 4 frames times of delay, whilst the field rate version exhibits 3 field times of delay. This can be accounted for by raster scan latencies, processing time, deinterlacing for the frame rate version, inter processor communications and robot motion time. The vision processing transfer function is modelled as \( V(z) = \frac{K_{obs}}{z^3} \), where

![Control system diagram (frame rate tracker).](image-url)
$K_{obs}$ is the observation gain in mm/pixel and is related to the calibration gains discussed below. We will ignore mechanical and servo dynamics of the robot and model it as a single sample time delay, $R(z) = \frac{1}{z}$. Thus the open-loop transfer function for the frame rate tracker is

$$\frac{K_{obs}}{z^3(z-1)}$$

This system is of type 1, and will thus exhibit steady state error for a constant velocity target. The simple compensator,

$$D(z) = K \frac{z-a}{z-1}$$

introduces another pole at $z = 1$ to produce a type 2 system, while the zero is used to position the closed-loop poles.

The achievable closed-loop bandwidth of a discrete system is typically less than one fifth of the sample rate, or approximately 30rad/s in this case.

4.1. Step response tests

The step response of the system was measured by a test rig with two spatially separated LEDs, only one of which was illuminated at a time. A manual switch controlled the LEDs and joint trajectory logging code in the robot server was activated by a change in the LEDs as sensed via an A/D converter channel.

Figure 5. Measured closed-loop step response.
The compensator zero was located at $z = 0.95$, and a loop gain of 0.12 was chosen. This yields dominant closed loop poles at $z = 0.915\pm0.0619j$, which yields a time to first peak of 1.3s with 1.5% overshoot. Figure 5 is the measured response of joint 2 of the robot, to a step applied at time 0. It can be seen that the measured response shows considerable overshoot, 36%, which is due to the action of the zero in the compensator.

The simple classical compensator is very sensitive to gain and compensator zero position, and performance is very far short of the potential bandwidth. Further work with state-feedback control and predictive state estimators is planned.

4.2. Control behaviour

The controller described works well in practice, and has the advantage of being computationally inexpensive. The visual tracking process does not produce cartesian rate commands at a constant interval, occasionally skipping video frames when Unix schedules other processes to run. In practice this causes very little problem since the robot carries on at its previous velocity until another command is received. This strategy could also be used to follow a target moving, with low acceleration, that becomes temporarily obscured or merged with the background.

5. Depth determination

A camera calibration is required to relate real world dimensions to those sensed by the camera in pixel space. Due to the aspect ratio of standard TV screens an image of a circle will appear as a vertically elongated ellipse in pixel space, thus two calibration constants $K_x$ and $K_y$ are determined. These are defined such that at a distance $u$ one pixel represents a rectangle of width $K_x u$ and height $K_y u$.

Many techniques for depth determination exist[33], and could perhaps be categorized as static, using cues such as lighting or texture, and triangulation based where multiple views from moving cameras or stereo camera pairs are interpreted. The technique outlined here is a simple static technique based on a priori knowledge of the target object's area.

The area of a pixel is

$$A_{pixel} = K_x K_y u^2$$

If the object of known area $A$, is determined by the APA to have $N$ pixels then an estimate of the objects distance is

$$x_{est} = \sqrt{\frac{A}{K_x K_y N}}$$

This simple technique works well in experiments, although factors such as focus and threshold may be expected to introduce errors. Further work using a continuous sequence of views from a moving camera has been commenced.

6. Further work and discussion

The initial objective relating to non-trivial scenes has been only partially met. The system works well on a scene containing many regions but good contrast between target and background is still required.

Motion blur effect in the camera has not been a problem due to the slow moving objects currently used. Experiments have shown that interlaced frames are substantially distorted due to the delay between exposure times of the component fields, but this problem is eliminated when
working at field rate. Motion blur effects have been investigated in detail by Anderson[21]

The development and execution of the visual tracker has been carried out under Unix on a workstation computer. Unix is an excellent environment for software development, and on a graphics workstation becomes a very powerful tool indeed. Unix is much maligned for its poor real-time performance, but in reality does an acceptable job. The particular machine used here with a 25MHz processor and 8Mbyte of memory, and with a single user did not page or swap processes. The only processes competing for the processor were the application and the periodic update daemon. Unix allows for rapid prototyping of software, and by placing the most response critical code in device drivers an acceptable level of performance was obtained for experimental purposes.

This work, and others such as[21] show that video standards and conventional cameras are now a limiting factor in high performance video sensing applications. Video technology is currently tied to the needs of the television industry where conflicting requirements such as small bandwidth and flicker have to lead to such devices as interlaced scanning. However when the end user is a machine vision system the requirements are very different. Fortunately, a new generation of cameras with both high resolution and high speed non-interlaced data formats are becoming available.

Areas for further work include:

1. Determining the trajectory of target objects so that some attempt to follow them through occlusion or background clutter can be made.

2. Investigation of more sophisticated control algorithms to improve the position response time of the closed-loop system.

3. Investigate adaptive segmentation based on feedback of segmentation results from the APA so as to modify edge and scene thresholds.

7. Conclusion

This paper has described a high performance image processing system capable of providing robot positioning commands at rates of up to 60Hz, and demonstrated closed-loop position control. The image processing subsection is comprised entirely of off-the-shelf components, notably a hardware region growing and moment generation unit.

The region oriented approach used here enables the system to identify the target object whether or not there is relative motion between object and camera.

8. References


9. Acknowledgments

This research was conducted at the General Robotics and Sensory Perception (GRASP) Laboratory, University of Pennsylvania.

The authors are grateful to Vision Systems Ltd, Adelaide, Australia, for lending me the APA-512 unit upon which this project is based. Professor Kwangyoen Wohn of the University of Pennsylvania provided advice and comments. Dr Bob Brown, Chief of the CSIRO Division of Manufacturing Technology, Melbourne Australia, made this visit to University of Pennsylvania possible.
A.12 Stability of Robot Systems with Random Sample Rates
On the Stability of Robotic Systems with Random Communication Rates*

H. Kobayashi††, X. Yun, R. P. Paul
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104-6389

Abstract

This paper studies control problems of sampled data systems which are subject to random sample rate variations and delays. Due to the rapid growth of the use of computers more and more systems are controlled digitally. Complex systems such as space telerobotic systems require the integration of a number of sub-systems at different hierarchical levels. While many sub-systems may run on a single processor, some sub-systems require their own processor or processors. The sub-systems are integrated into functioning systems through communications. Communications between processes sharing a single processor are also subject to random delays due to memory management and interrupt latency. Communications between processors involve random delays due to network access and to data collisions. Furthermore, all control processes involve delays due to causal factors in measuring devices and to signal processing.

Traditionally, sampling rates are chosen to meet the worst case communication delay. Such a strategy is wasteful as the processors are then idle a great proportion of the time; sample rates are not as high as possible resulting in poor performance or in the over specification of control processors; there is the possibility of missing data no matter how low the sample rate is picked.

Randomly sampled systems have been studied since later 1950's, however, results on this subject are very limited and they are not applicable to practical systems. This paper studies asymptotical stability with probability one for randomly sampled multi-dimensional linear systems. A sufficient condition for the stability is obtained. This condition is so simple that it can be applied to practical systems. A design procedure is also shown.

1 Introduction

Many complex systems today involve the integration of a number of different subsystems at various hierarchical levels. Examples of hierarchical subsystems are, for example, in the case of spacecraft:

Level 1 – Assignment of systems to tasks;
Level 2 – Assignment of subsystems to task systems, such as the shuttle manipulator, one of more cameras, an astronaut on EVA;

*This material is based upon work supported by the National Science Foundation under Grant No. ECS-11879. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.
†Dept. of Precision Eng., Meiji Univ., Japan. He is currently visiting at Dept. of Computer and Information Science, Univ. of Penn.
Level 3 – Control of individual subsystems, cameras comprised of pan tilt, zoom, focus, feature tracking, exception warning; or control of machine tools comprised of spindle, table, tool changer, gauge;

Level 4 – Control of elements, control of manipulator joints, end-effector force measurement, machine tool spindle drive, elevator motor drive, submaring plane control.

These systems all comprise many components which may be ranked hierarchically. Many of the components are now computer controlled and are integrated by means of digital busses or networks. The integration of these components into functioning feedback systems, camera – force-sensor – manipulator roll sensor, pitch sensor – main propulsion – planes, implies in addition to data communications, communication rates.

In the case of communication networks, the data rates of point to point communication busses are well known. The rates at which a computer can respond to communication data interrupts requests add a variance to the data rates. In the case of shared networks, such as Ethernet, data collisions add considerable variance to the data rate frequently exceeding the data rate itself. However, these shared networks are very attractive from both the reliability and flexibility standpoints.

Because of their flexibility in programming and speed in computing, digital computers are now regularly employed as integral components of dynamic feedback control systems. They are easily programmed to realize desired compensators. Due to the discrete nature of digital computers, variables in dynamic systems are sampled and quantized before sending to the computers. The well established discrete time system theory (e.g., [8]) provides methods to analyze the behavior of sampled data systems, based on the assumption that the sampling rates are fixed and the same, and the sampling operations on different channels of the systems are synchronized. If the sampling rates are fixed but different on different channels, known as multi-rate sampling, the system analyses are simple if the sampling rates have integral ratios [6, 10].

Due to random delays in measurement devices, signal processing, interrupt latency, priority scheduling, conditional branching, network communications, etc., sampling rates vary randomly in many systems, and the system performance could be expected to be improved if a theory supporting random sampling rates was used. Systems with random sampling processes are called randomly sampled systems. The behavior of a randomly sampled system is, presumably, related to the statistical properties of the random sampling processes as well as system parameters. Randomly sampled systems have been studied by Kalman [11], Leneman [16], Kushner and Tobias [15], Agnien and Jury [2], and others. One of the major motivations for studying randomly sampled systems in late 1950's and early 1960's was the introduction of digital computers in control systems. However as the speed of computers improved dramatically, time delays caused by computers became practically negligible in simple single processor controlled systems compared to other delays, and research on randomly sampled systems came to an end. Nowadays, development of computer controlled systems has reached beyond the stage of single processor control. Many subsystems are integrated into large systems. Furthermore, many complex dynamic systems impose demanding computation requirement. For example, computation time becomes a bottleneck in the implementation of dynamic control algorithms of multi-joint robot manipulators. Delay caused by computation and communication is no longer a negligible factor.

Early researchers in the area of randomly sampled systems primarily considered stability conditions of the systems. Their work is briefly summarized below. Kalman carried out a comprehensive study of sampling systems [11]. He classified sampling into six categories: conventional sampling, nonsynchronous sampling, multiple-order sampling, multi-rate sampling, noninstantaneous sampling, and random sampling. For randomly sampled systems, Kalman showed that if the second moment of the output of an autonomous system is stable, the second moment of the output remains bounded when a bounded input is applied to the system. Based on his state space method [13], Kalman [12] also discussed the regulator problem and stability of a linear system described by independent random functions. This class of systems include randomly sampled systems. Thus the stability conditions obtained for this class of systems are applicable for randomly sampled systems. Kushner and Tobias [15] studied an autonomous linear system with linear and nonlinear feedback. Using a stochastic Lyapunov function, criteria for stability with probability one
and s-th moment stability ($s > 0$) were obtained for scalar linear systems, and criteria for stability with probability one and second moment stability were obtained for multi-dimensional linear systems. Agniel and Jury [2] investigated asymptotic stability with probability one of a linear system with a saturating type nonlinear component. A computational procedure was provided to determine the largest stability sector of the nonlinearity for asymptotic stability with probability one. Using a stochastic Lyapunov function, Agniel and Jury in another paper [1] gave a condition for the asymptotic stability with probability one and the second moment asymptotic stability for single-input single-output multi-dimensional linear systems. They also showed that if an autonomous system exhibits asymptotic stability with probability one, the system is almost surely bounded input–bounded output. Leneman [16] studied a single-input single-output first order linear system with feedback. He derived the second moment of the output for the cases with and without input. The input is a stationary stochastic process independent of the sampling process. Consequently, a condition for the second moment stability was given. Assuming the independence of the sampling times and the signals, Dannenberg and Melsa [7] took the expectation of a linear system equation, obtaining a system equation of expectation of the states and outputs. The first moment stability analysis is similar to that of deterministic sampled-data systems. An example of a spacecraft control problem was given, in which it is assumed that there is a probability of missing messages. The problem of random sampling of a random signal was studied by Bergen [4] and Leneman [17]. Their focus was on deriving expressions of the spectral density of a random signal after a random sampling.

This paper studies the stability of randomly sampled systems in relation to the random sampling processes. Though Kalman [11] and Kushner [15] have obtained necessary and sufficient conditions for the stability in the second moment, it is not so easy to apply these conditions to practical systems. This paper studies asymptotic stability with probability one and gives a necessary and sufficient condition for one-dimensional systems and a sufficient condition for multi-dimensional systems. These conditions are easy to verify for given sampling distributions and are thus applicable to practical systems.

In the next section, the asymptotical stability with probability one is defined. A sufficient condition is given for multi-dimensional linear time-invariant randomly sampled systems which is also necessary for one-dimensional systems. A design procedure to determine feedback gains is obtained in Section 3. If we use a nonlinear compensator such as a computed torque controller for a robotic control system, then we would have a set of simple two-dimensional linear systems. In Section 4, the stability of such two-dimensional systems is considered and the design procedure is shown for a Bernoulli distribution, a uniform distribution and a mixed uniform distribution.

## 2 Stability

Consider following linear time-invariant control system.

\[ \dot{z}(t) = Az(t) + Bu(t), \]

where $z$ is an $n$-dimensional state vector, $u$ an $r$-dimensional control vector, and $A$ and $B$ are $n \times n$ and $n \times r$ matrices, respectively. For this system, we apply a constant state feedback input

\[ u(t) = Kz(t_k), \]

from $t = t_k$ to $t = t_{k+1}(= t_k + \Delta_k)$, where $K$ is an $r \times n$ matrix. Then $z(t_{k+1})$ is given as follows.

\[ z(t_{k+1}) = (\Phi(\Delta_k) + \Psi(\Delta_k)K)z(t_k), \]

where

\[ \Phi(\Delta_k) = \exp(A\Delta_k), \text{ and } \Psi(\Delta_k) = \int_0^{\Delta_k} \exp(A\tau)dB. \]
Sampling interval $\Delta_k$ is assumed to be subject to some probability distribution function $F(\Delta)$ or distribution density function $f(\Delta)$ and $\Delta_i$ and $\Delta_j (i \neq j)$ are statistically independent of each other. For simplicity, we write Eq. (3) as follows

$$x_{k+1} = \Gamma(\Delta_k)x_k.$$ (4)

In this paper, we use the following matrix norm which is compatible with usual Euclid norm for vectors:

$$||\Gamma|| = \{\sigma(\Gamma^*\Gamma)\}^{1/2},$$ (5)

where $\Gamma^*$ is the conjugate transformed matrix and $\sigma(\Gamma)$ denotes the maximum eigenvalues of the matrix $\Gamma$. Note, however, that while the stability of the system (1) or (3) is invariant under a similarity transformation of the state variables, the matrix norm depends on the transformation, namely in general

$$||\Gamma|| \neq ||T^{-1}\Gamma T||.$$

The stability of randomly sampled control system Eq. (4) is defined as follows.

**Definition 1 (Stability)** The randomly sampled data system Eq. (4) is asymptotically stable with probability one if

$$\text{Prob}\{\lim_{k \to \infty} ||x_k|| = 0\} = 1$$

for any initial state $x_0$.

Now we define the following notation:

$E[\omega]$ : Expectation of random variable $\omega$,

$V[\omega]$ : Variance of random variable $\omega$,

and assume that

$$E[(\log(||\Gamma(\Delta)||))^2] < \infty.$$ (6)

Then a sufficient condition of the asymptotical stability is given in the next proposition.

**Proposition 1 (Sufficient Condition)** Randomly sampled control system (1) is asymptotically stable with probability one if there exists a non-singular matrix $T$ such that

$$E = E[\log(||T^{-1}\Gamma(\Delta)T||)] < 0,$$ (7)

We also have

$$\text{Prob}[||T^{-1}x_k|| < ||T^{-1}x_0|| \exp\{k(E + \epsilon)\}] > 1 - \frac{V}{k\epsilon^2},$$ (8)

for any $\epsilon > 0$, where $V = V[\log(||T^{-1}\Gamma(\Delta)T||)]$

< proof > Assuming $x_0 \neq 0$ without loss of generality, from Eq. (4) we have

$$\log(||T^{-1}x_k||) = \sum_{i=0}^{k-1} \log(||T^{-1}\Gamma(\Delta_i)T||).$$

Then the proposition is easily proved by the statistical independence of $\Delta_i$'s and Thebyshov's inequality.

< end of proof >
We note that for one-dimensional systems the condition stated in the above proposition is necessary and sufficient for the asymptotic stability with probability one [14]. If the sampling interval is constant, the condition in Prop. 1 is also necessary for the asymptotic stability of multi-dimensional systems.

Now we define
\[
\gamma(\Delta) = \log(||T^{-1}T(\Delta)||), \quad \text{and} \quad g(\Delta) = \int_0^\Delta \gamma(\tau) d\tau,
\]
then we have the following proposition.

**Proposition 2**

i. If the sampling rate $\Delta$ is subject to a Bernoulli distribution where $\Delta = \alpha$ with probability $p$ and $\Delta = \beta$ with probability $q = 1 - p$, then the system is asymptotically stable with probability one, if
\[
p\gamma(\alpha) + q\gamma(\beta) < 0.
\]

ii. If the sampling rate $\Delta$ is subject to a uniform distribution $U[\alpha, \beta]$, then the system is asymptotically stable with probability one, if
\[
g(\alpha) < g(\beta).
\]

iii. If the sampling rate $\Delta$ is subject to $U[\alpha, \beta]$ with probability $\varepsilon$ and to $U[\mu, \nu]$ with probability $1 - \varepsilon$, then the system is asymptotically stable with probability one, if
\[
\varepsilon \frac{g(\beta) - g(\alpha)}{\beta - \alpha} + (1 - \varepsilon) \frac{g(\nu) - g(\mu)}{\nu - \mu} < 0.
\]

The proof is straightforward, so we omit it here.

**3 Design Procedure**

Next we discuss a design procedure of a feedback gain $K$ and a matrix $T$ in the following. Now, assume that system
\[
\dot{x}(t) = Ax(t) + Bu(t)
\]
is controllable, then it is well known that the discretized system
\[
x_{k+1} = \Phi(\Delta_k)x_k + \Psi(\Delta_k)u_k
\]
is also controllable for almost all sampling interval $\Delta_k$ [5]. Then we can assign poles $\{\lambda_i, i = 1, 2, \ldots, n\}$ to system (11) if poles $\{\lambda_i\}$ are symmetric with respect to the real axis. Here, we apply Hikita’s pole assignment algorithm[9] to the randomly sampled control systems.

[Algorithm]

1. **For given $\{\lambda_i\}$, find $r$-dimensional vectors $v_i$, $i = 1, 2, \ldots, n$, which makes matrix $T(\Delta) = [v_1 : \cdots : v_n]$ non-singular. Vector $v_i$'s are given as follows where $\Phi = \Phi(\Delta)$ and $\Psi = \Psi(\Delta)$.

   - if $\lambda_i$ is a real number, then
     \[
     v_i = (\Phi - \lambda_i I_n)^{-1}\Psi v_i.
     \]
• if \( \lambda_i \) and \( \lambda_{i+1} \) are conjugate complex numbers \( \alpha_i \pm j\beta_i \), then

\[
v_i = V_{1i}\xi_i - V_{2i}\xi_{i+1}, \text{ and } v_{i+1} = V_{1i}\xi_i + V_{2i}\xi_{i+1},
\]

where

\[
V_{1i} = \{(\Phi - \alpha_i I_n)^2 + \beta_i^2 I_n\}^{-1}(\Phi - \alpha_i I_n)\Psi, \text{ and } V_{2i} = \{(\Phi - \alpha_i I_n)^2 + \beta_i^2 I_n\}^{-1}\beta_i\Psi.
\]

step (ii) Feedback gain \( K \) is given as follows.

\[
K(\hat{\Delta}) = -[\xi_1 : \cdots : \xi_n]T(\hat{\Delta})^{-1}.
\]

step (iv) Check the stability using Proposition 1 or 2. If not stable, return step (i) and try another \( \{\lambda_i\} \) and/or \( \Delta \).

It is easy to show that for this \( T(\hat{\Delta}) \) and \( K(\hat{\Delta}) \), we have

\[
||T^{-1}(\hat{\Delta})\Gamma(\hat{\Delta})T(\hat{\Delta})|| = \max_i(|\lambda_i|).
\]

Hence we can use matrices \( T(\hat{\Delta}) \) and \( K(\hat{\Delta}) \) to calculate \( \gamma(\Delta) \) and \( g(\Delta) \). In the next section, we use notations \( \gamma(\Delta, \hat{\Delta}) \) and \( g(\Delta, \hat{\Delta}) \) for \( \gamma(\Delta) \) and \( g(\Delta) \), respectively, to show the dependence of the functions on \( \Delta \) clearly.

4 Two Dimensional Systems

In this section, we consider control of robot manipulators. We view a robot manipulator as a component of a large system, such as a space station. The robot controller communicates with the other components of the system to achieve cooperative actions. Communication between components is considered to have a longer delay than that within a component. We assume that the robot controller has an inner feedback loop which compensates the nonlinearity of manipulator dynamics and operates independently of the other part of the system. The robot dynamic system together with the inner feedback loop becomes a linear system. It is feasible to treat the robot manipulator subsystem as a linear system when integrating and communicating with the other components. For example, if we use the nonlinear feedback controller developed in [3], we have \( n \) (=DOF of manipulator) decoupled two-dimensional linear systems

\[
\dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)
\]

where \( x(t) = (e_i(t), \dot{e}_i(t)) \) is the error vector for the \( i \)-th component of outputs and \( u(t) \) is the corresponding input for this component of outputs. If the task is specified in joint space (the joint space control), the \( i \)-th component of output is simply the displacement of the \( i \)-th joint and the error vector is composed of the joint position error and joint velocity error.

We now study the asymptotical stability of this system under the random sampling rate. The corresponding discrete time system is easily obtained for a sampling interval \( \Delta \) as follows.

\[
x_{k+1} = \begin{bmatrix} 1 & \Delta_k \\ 0 & 1 \end{bmatrix} x_k + \begin{bmatrix} \Delta_k^2/2 \\ \Delta_k \end{bmatrix} u_k.
\]

We apply the algorithm given above to this system directly. Then we have the following proposition.
Proposition 3 (PD Controller) Assume that \( \{\lambda_i\} = \{\lambda_1, \lambda_2\} \) where \( \lambda_1 \neq \lambda_2 \), then we have

\[
K(\hat{\Delta}) = ((\lambda_1 + \lambda_2 - \lambda_1 \lambda_2 - 1)/\hat{\Delta}^2, (\lambda_1 + \lambda_2 + \lambda_1 \lambda_2 - 2)/(2\hat{\Delta}))
\]

and

\[
\gamma(\Delta, \hat{\Delta}) = \gamma(\theta, 1),
\]

where \( \theta = \Delta/\hat{\Delta} \).

The proof is obtained by direct calculation. This proposition implies that the function \( \gamma(\Delta, \hat{\Delta}) \) is the same as the function \( \gamma(\theta, 1) \) if we use \( K(\hat{\Delta}) = (k_p/\hat{\Delta}^2, k_v/\hat{\Delta}) \) instead of \( K(1) = (k_p, k_v) \). Therefore we have \( g(\Delta, \hat{\Delta}) = \hat{\Delta}g(\theta, 1) \) for the same \( K(\hat{\Delta}) \). This fact is very useful to design the feedback gain. This will be shown by examples.

Fig. 1 shows \( \gamma(\theta, 1) \) and \( g(\theta, 1) \) for \( \lambda_1 = 0.4 \) and \( \lambda_2 = 0.7 \), where we have

\[
K(1) = -(0.18, 0.81), \quad T(1) = \begin{bmatrix} -0.759 & -0.934 \\ 0.651 & 0.333 \end{bmatrix},
\]

and \( \xi_i \) was used to make the norm of column vectors of \( T \) matrix be equal to one.

Example 1 (Bernoulli Distribution) Let's assume that the sampling interval is subject to Bernoulli distribution, i.e. \( \Delta = \alpha \) with probability \( p \) and \( \Delta = \beta \) with probability \( q \), where \( \alpha < \beta \), \( 0 \leq p \leq 1 \), and \( q = 1 - p \). The sufficient stability condition is given as follows.

\[
p\gamma(\alpha/\hat{\Delta}, 1) + q\gamma(\beta/\hat{\Delta}, 1) < 0.
\]

Note that if \( \hat{\Delta} \geq \beta/1.96(= \hat{\Delta}^*) \) then the system is asymptotically stable for any \( \alpha \) because \( \gamma(\theta, 1) < 0 \) for any \( \theta \leq 1.96 \). But we are generally interested in the smallest \( \hat{\Delta} \) because it gives us the fastest response.

Fig. 1 shows that the function \( \gamma(\theta, 1) \) reaches the minimum value \(-0.417\) at \( \theta = 1.35 \). Let \( \theta^* \) be the point which satisfies the following equation.

\[
\gamma(\theta^*, 1) = \frac{p}{q} \times 0.417.
\]
Then it is clear that \( \hat{\Delta} \) must be greater than \( \hat{\Delta}_{\text{min}}(= \beta/\theta^*) \) for Eq. (19).

A suitable value of \( \hat{\Delta} \) can be found from the range \( \hat{\Delta}_{\text{min}} < \hat{\Delta} < \hat{\Delta}^* \) by a trial-and-error method using Fig. 1 or Table 1 which gives pairs of \( \theta_1, \theta_2 \) such that \( \gamma(\theta_1, 1) = \gamma(\theta_2, 1) \).

(i) Calculate \( a = -(q/p)\gamma(\beta/\hat{\Delta}, 1) \).

(ii) Find \( \{\theta_1, \theta_2\} \) such that \( \gamma(\theta_1, 1) = \gamma(\theta_2, 1) \leq a \) using Fig. 1 or Table 1.

(iii) Check \( \theta_1 < \alpha/\hat{\Delta} < \theta_2 \). If so, calculate \( K(\hat{\Delta}) \). If not so, go back to step (i) with another \( \hat{\Delta} \).

For example, if \( \alpha = 10 \text{ msec}, \beta = 30 \text{ msec}, \) and \( p = 0.75 \), then \( \theta^* \) is about 3.64 and \( \hat{\Delta}_{\text{min}} = 8.24 \text{ msec} \), while \( \hat{\Delta}^* = 15.3 \text{ msec} \). If we select \( \hat{\Delta} = 11 \text{ msec} \) then \( \frac{1}{2}\gamma(\beta/\hat{\Delta}, 1) = -0.278 \) and \( \alpha/\hat{\Delta} = 0.91 \). Therefore we can try the 6-th row of Table 1, and we have \( \theta_1 = 0.84 < 0.91 < \theta_2 = 1.68 \). Hence the system is asymptotically stable for \( K = -(1488, 73.64) \).

Example 2 (Uniform Distribution) Now assume that \( \Delta \) is subject to a uniform distribution \( U[\alpha, \beta] \).

The sufficient condition of the asymptotic stability with probability one is given as follows:

\[
g(\alpha/\hat{\Delta}, 1) > g(\beta/\hat{\Delta}, 1).
\]

The function \( g(\theta, 1) \) has its minimum value at \( \theta = 1.96 \). Now we define \( \hat{\Delta}^* = \beta/19.6 \) and \( \hat{\Delta}_{\text{min}} = \beta/2.89 \). If \( \hat{\Delta} \geq \hat{\Delta}^* \), then the above sufficient condition is satisfied for any \( \alpha \). Therefore the system is asymptotically stable if \( \hat{\Delta} \geq \hat{\Delta}^* \). On the other hand, if \( \hat{\Delta} \leq \hat{\Delta}_{\text{min}} \), then the above condition is not satisfied for any \( \alpha \).

Table 1 also gives pairs of \( \{\theta_3, \theta_4\} \) and the ratio \( \theta_3/\theta_4 \) such that \( g(\theta_3, 1) = g(\theta_4, 1) \). If there is a pair \( \{\theta_3, \theta_4\} \) such that \( \alpha/\beta > \theta_3/\theta_4 \), then the system is asymptotically stable for the \( K(\hat{\Delta}) \) where \( \hat{\Delta} = \alpha/\theta_3 \).

Therefor we can determine \( \alpha \) easily using this table as follows:

(i) Calculate \( a = \alpha/\beta \).

(ii) Find a pair \( \{\theta_3, \theta_4\} \) in the Table 1 such that \( \alpha > \theta_3/\theta_4 \).

(iii) Calculate \( \hat{\Delta} = \alpha/\theta_3 \) and \( K(\hat{\Delta}) \).

Now assume that \( \alpha = 10 \text{ msec} \) and \( \beta = 30 \text{ msec} \), then we have \( \hat{\Delta}^* = 15.3 \text{ msec}, \hat{\Delta}_{\text{min}} = 10.38 \text{ msec} \), and \( \alpha/\beta = 1/3 > 0.273 \) in the Table 1. Therefore we can use \( \alpha/\hat{\Delta} = 0.75 \) and \( \hat{\Delta} = 13.33 \text{ msec} \). Hence the system is asymptotically stable with \( K = -(1065, 62.31) \) if \( \beta < 36.7 \text{ msec} \). Table 2 shows the IAE (Integration of Absolute value of the Error) for fifty random streams with the initial condition \( x(0) = (1.0, 0)^T \).

The table shows that when \( \beta \geq 40 \text{ msec}, \) the STD (STandard Deviation) and the maximum values of IAE for the velocity error \( \dot{e}_1(t) \) become very large compared to the cases where \( \beta \leq 35 \text{ msec} \). This means that the system is still stable but there is a large vibration in the response for \( \Delta \geq 40 \text{ msec} \).

Example 3 (Mixed Uniform Distribution) Next we assume that \( \Delta \) is subject to a uniform distribution \( U[\alpha, \beta] \) with probability \( \epsilon \) and to \( U[\mu, \nu] \) with probability \( 1 - \epsilon \). The sufficient condition is given as follows:

\[
E = \epsilon^g(\beta/\hat{\Delta}, 1) - g(\alpha/\hat{\Delta}, 1) + (1 - \epsilon)^g(\nu/\hat{\Delta}, 1) - g(\mu/\hat{\Delta}, 1) < 0.
\]

Though the selection of \( \hat{\Delta} \) becomes a little difficult, we can use the following procedure to estimate an appropriate \( \hat{\Delta} \):

(i) Define \( \alpha = (\alpha + \beta)/2.0, \beta = (\mu + \nu)/2.0, p = \epsilon, \) and \( q = 1 - p \).
Figure 2: Simulations for Bernoulli Distribution, Uniform Distribution and Mixed Uniform Distribution

(ii) Determine $\Delta$ using the procedure in Exam. 1 for $\alpha = \bar{\alpha}$ and $\beta = \bar{\beta}$.

(iii) Check the condition. If satisfied, calculate $K(\Delta)$. If not, try another value for $\Delta$.

Now assume that $\Delta$ is subject to $U[5 \text{ msec}, 15 \text{ msec}]$ with probability $\epsilon = 0.75$ and to $U[20 \text{ msec}, 40 \text{ msec}]$ with probability 0.25. Then we have $\bar{\alpha} = 10\text{ msec}$, $\bar{\beta} = 30\text{ msec}$, $p = 0.75$, and $q = 0.25$. If we use $\Delta = 11\text{ msec}$ from the result of Exam. 1, then we have $E = -0.04 < 0$. Therefore the system is asymptotically stable for the same $K = -(1488, 73.64)$.

Fig. 2 shows the simulations of $x(t)$ for three cases discussed above where $x(0) = (1.0, 0)^T$.

It is easily shown that even if we use a PID controller

$$ z_{k+1} = z_k + [1 : 0]x_k, \text{ and } u_k = K_1 z_k + K_2 x_k, $$

or a PD controller with one step delay

$$ u_k = K(\Phi(\Delta)x_{k-1} + \Psi(\Delta)u_{k-1}), $$

instead of the PD controller given in Prop. 3, we have the similar proposition. Therefore we can determine $\Delta$ easily.

5 Conclusions

In this paper, the stability of randomly sampled linear control systems was discussed and the following results were obtained.

1. A sufficient condition for the asymptotical stability in a norm with probability one was obtained for multi-dimensional systems.

2. For a simple two-dimensional system with PD controllers, a design procedure was shown which was easily applicable to systems with PID controllers or PD controllers with one step delay.

The results given in this paper are also easily applicable to the robotic control systems where computed torque controllers or PD controllers with a feedforward term are used at the random sampling rate. The results will be shown in the near future [14].
Table 1: $\theta_1$, $\theta_2$, $\theta_3$, and $\theta_4$

<table>
<thead>
<tr>
<th>$\gamma(\theta_1,1) = \gamma(\theta_2,1)$</th>
<th>$g(\theta_3,1) = g(\theta_4,1)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\gamma(\theta,1)$</td>
<td>$\theta_1$</td>
</tr>
<tr>
<td>0.00</td>
<td>0.60</td>
</tr>
<tr>
<td>-0.05</td>
<td>0.18</td>
</tr>
<tr>
<td>-0.10</td>
<td>0.33</td>
</tr>
<tr>
<td>-0.15</td>
<td>0.46</td>
</tr>
<tr>
<td>-0.20</td>
<td>0.58</td>
</tr>
<tr>
<td>-0.25</td>
<td>0.71</td>
</tr>
<tr>
<td>-0.30</td>
<td>0.84</td>
</tr>
<tr>
<td>-0.35</td>
<td>0.98</td>
</tr>
<tr>
<td>-0.40</td>
<td>1.18</td>
</tr>
</tbody>
</table>

Table 2: IAE for $\mathcal{U}[10 \text{ msec}, \beta \text{ msec}]$

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>$e_i(t)$</th>
<th>$\dot{e}_i(t)$</th>
<th>$e_\epsilon(t)$</th>
<th>$\dot{e}_\epsilon(t)$</th>
<th>$e_i(t)$</th>
<th>$\dot{e}_i(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0.0531</td>
<td>0.9969</td>
<td>0.0022</td>
<td>0.0011</td>
<td>0.0572</td>
<td>1.0020</td>
</tr>
<tr>
<td>30</td>
<td>0.0511</td>
<td>1.0339</td>
<td>0.0043</td>
<td>0.0200</td>
<td>0.0561</td>
<td>1.1310</td>
</tr>
<tr>
<td>35</td>
<td>0.0498</td>
<td>1.0198</td>
<td>0.0060</td>
<td>0.0515</td>
<td>0.0589</td>
<td>1.2370</td>
</tr>
<tr>
<td>40</td>
<td>0.0509</td>
<td>1.2418</td>
<td>0.0077</td>
<td>0.4305</td>
<td>0.0717</td>
<td>2.7051</td>
</tr>
<tr>
<td>45</td>
<td>0.0709</td>
<td>2.6492</td>
<td>0.0638</td>
<td>4.4592</td>
<td>0.4354</td>
<td>30.276</td>
</tr>
</tbody>
</table>
References


