A PC-based Workstation for Robotic Discectomy

Claudio Casadei[†], Paolo Fiorini[‡], Sandra Martelli[†]
Marco Montanari[†] and Alberto Morri[†]

[†]Laboratorio di Biomeccanica
Istituti Ortopedici Rizzoli
Bologna, Italia 40136

[‡]Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Abstract

This paper describes a PC-based controller for robot-assisted minimally invasive surgery. The development is motivated by the need of reducing the exposure of operating room personnel to X-rays during surgical procedures such as percutaneous discectomy. The approach taken consists of upgrading the hardware architecture and the control software of a clean-room PUMA 260 manipulator, and by developing a new vision-based operator interface. The original PUMA controller is interfaced to a PC for trajectory generation and force/torque data acquisition. Kinematic and communication functions are ported to the PC from the RCCL package. The operator interface is used for robot calibration and motion commands. In this implementation, position control and force/torque data acquisition are executed every 28 ms, thus preserving the original robot performance. Preliminary tests have shown the validity of this approach and readiness of the system for more realistic experiments in the operating room.

1 Introduction

Lower-back injuries are a frequent pathology affecting patients of all ages and occupations. It is estimated, for example, that more than 400,000 lower-back operations are performed every year in the US alone. It, is therefore very important to devise new, minimally invasive procedures that could reduce the treatment cost and the hospitalization time of these injuries. Some of these procedures would benefit from the use of robotic manipulators, but very few commercial robots have the performance needed by surgical operations, and none has a cost compatible with the budget of a research laboratory.

An appealing solution is to enhance the capabilities of a general purpose robot by increasing its computation power and by adding advanced control and new safety features. This route has been pursued from the beginning of commercial robotics [10, 1], and has led to important results that benefit from the capabilities of Unix workstations [5, 6, 4]. The increased computation power of economic personal computers (PC) allows now to develop similar robot controller in the PC environment. However, the solutions presented so far lack generality or simplicity of use. For example, a PC-based system connected to a robot controller via a serial interface is presented in [8]. Using this system, the operator can select a specific control law, set the controller gains, and compile the control law into a program to be downloaded to the robot controller. In [3] a PC-based controller for a dexterous manipulator is described, which integrates in a single off-the-shelf PC robot control and operator interface functions. Unfortunately these controllers can only be used with the high cost manipulators for which they were developed, since they rely on proprietary information from the robot manufacturer.

An approach to develop a PC-based robotic controller at a lower cost and for a larger user base than the above systems, is to refurbish an older PUMA manipulator by upgrading some of its hardware and software. So far however, this approach has received little attention because of the lack of detailed technical information from the manufacturer. Since some tech-

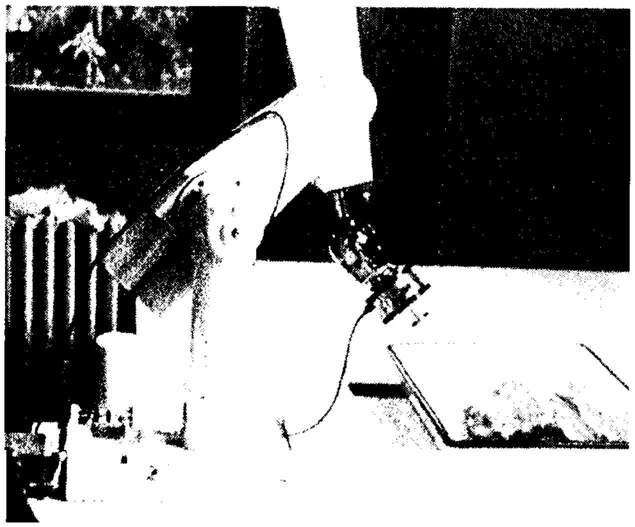


Figure 1: The PC-controlled surgical PUMA.

nical information about PUMA manipulators is found in the documentation of the Robot Control C Library (RCCL) [4], we based the development of this PC-based controller for PUMA's on the RCCL package, whose well written and extensively tested programs can avoid code duplication and reduce testing time. Since RCCL cannot be used directly in a PC environment due to its Unix origin, some reverse engineering is necessary to port RCCL functions and data structures to the PC environment.

This paper documents this process, in the hope of providing a useful methodology to other researchers involved in similar developments. To the best of our knowledge, this system is the first PC-based controller for PUMA robots developed using RCCL and integrating in a single commercial PC both robot control and operator interface functions.

The paper is Organized as follows. In the following Section we give the medical motivation for this work. In Section 3, we describe the hardware and software architecture of the system. In Section 4, we summarize the features of RCCL ported to the PC environment. Section 5 describes our current experiments using a vision-based operator interface. Finally, in Section 6 we draw some conclusions from this work and discuss the directions Of our future research and development.

2 Medical Motivation

Minimally invasive surgical procedures are routinely used for lower-back injuries, but a few specific injuries still require regular surgery or extensive X-ray monitoring. This is the case, for example, of lumbar radiculopathy due to herniated disks, which can be effectively treated using percutaneous discectomy, a minimally invasive surgery that requires the precise positioning of a guide needle against the lesion. Because of the position of the lesion, the surgeon does not see the area reached by the needle, and he/she must rely on fluoroscopic images and patient feedback to avoid damaging nerves and blood vessels. Figure 2 is a typical fluoroscopic image, showing the guide needle positioned on the herniated disk.

Percutaneous discectomy is an appealing procedure since it eliminates the need for entry into the spinal canal. It consists of the removal of the prolapsed tissue by inserting the surgical tools across a small skin incision until they reach the herniated disk. The tools are guided by the needle initially positioned by the surgeon. Percutaneous discectomy offers a number of advantages, including the avoidance of epidural bleeding and perineural fibrosis, the elimination of damage to the articular facets, and the preservation of spinal stability. Unfortunately, percutaneous discectomy is performed under fluoroscopy, thus exposing the medical staff to large amounts of radiation.

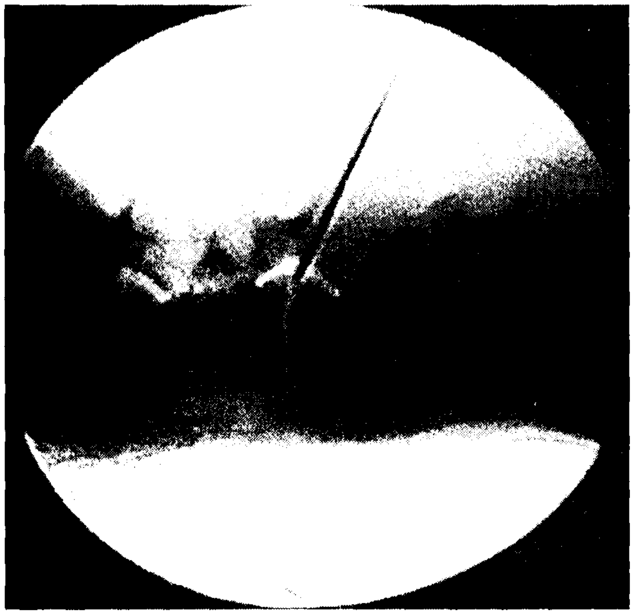


Figure 2: A fluoroscopic image of a discectomy.

To overcome the problem of radiation exposure, a semi-automatic procedure is proposed in [7], using a PUMA 260 manipulator to assist the surgeon during the needle insertion. A robotic arm can be used successfully in this procedure because of the simplicity of the surgical gesture required. In fact, the insertion of the guide needle consists of a simple linear trajectory, whose length and orientation are chosen on CAT scans during pre-operative planning. The proposed robot-aided procedure consists of three phases. In the first phase, the surgeon positions the tip of the needle mounted on the robot wrist at the starting point of the insertion. During the second phase, the manipulator autonomously adjusts the orientation of the needle to the planned value, and initiates the motion towards the prolaxed tissue. This motion is monitored by the surgeon in a location protected by the X-rays. Finally, when the target disk is reached, the needle is unlocked and the manipulator removed, leaving room for the surgeon to complete the procedure.

To perform this procedure successfully, several enhancements are needed to the standard PUMA controller, in particular in the areas of operator interface, operative planning and kinematic control. It is necessary to integrate the analysis of the CAT scans and of the fluoroscope images with the trajectory planning of the robot. Furthermore, kinematic singularity avoidance and force control must be added to the controller to increase operational safety. These features cannot be implemented using the standard PUMA motion controller, and a more powerful, PC-based controller must be developed for this application.

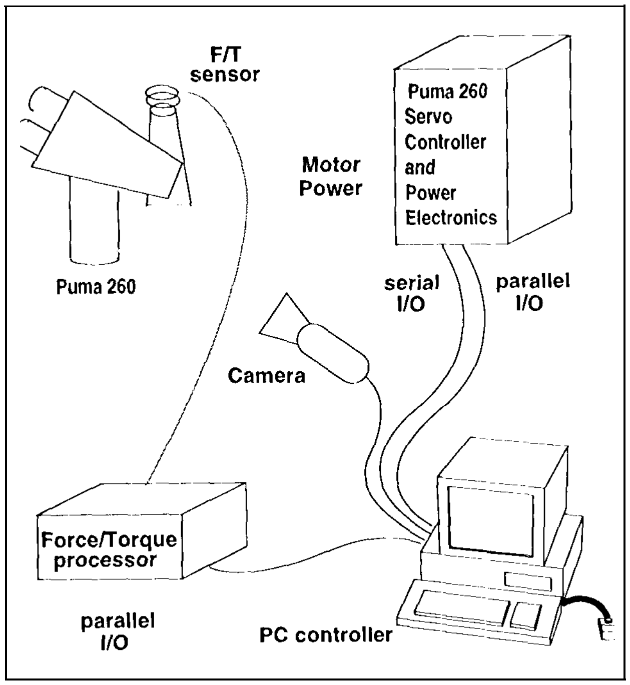


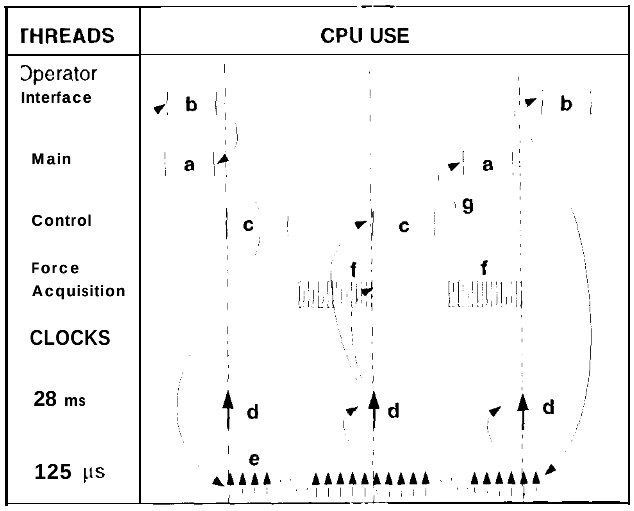
Figure 3: Schematic layout of the PC controller.

3 System Description

This section briefly describes the main components of the new PC-based controller. Its hardware design is driven by the need of interfacing standard laboratory equipment to the PUMA family of manipulators. This feature simplifies maintenance and eases the adoption of the robotic workstation by the medical community.

3.1 Hardware

The hardware used in the development of the PC-based surgical workstation consists of a PUMA 260 manipulator, a personal computer, an Assurance Tecl - nology Inc. (ATI) G-axis Force-Torque (FT) sensor, and a solid state TV camera, as shown schematically in Figure 3. The PUMA is equipped with its original LSI-1 1 controller enhanced by a DRV-1 1 board to carry out a parallel communication with the PC. The installation and settings of this board are described in [5]. The serial port of the PUMA controller is used for the initial set-up and for downloading the soft ware resident in the PUM A controller. The parallel port of the PUMA is used during normal operation, to exchange data between the PC and the PUMA servo controllers. The PC is equipped with a 1 00 MHz Pentium Intel microprocessor, 16 M Bytes of RAM, serial and parallel input/output ports, and a frame grabber for image acquisition. The frame grabber is capable of several input formats, to accept images from different operating room equipment, and it includes a video driver replacing the PC video driver to display both



Figu 4: Diagram of the software timing

camera images and the computer output on the PC monitor. During normal operation in the Operating Room, the camera will be replaced by fluoroscopic device, providing images of the needle insertion.

3.2 Software

The PC software is implemented using the DOS operating system. The advantages of this choice are in the large amount of software available, in the significant design expertise available, and in the compatibility with software packages used during pre-surgery planning.

The central element of the PC software is an interrupt handler activated by the real-time clock of the programmable interval-counter in the CMOS memory chip of the PC. This hardware interrupt dots not interfere with the PC clock and is normally associated to an empty interrupt handler. An interval of 125 μs is chosen to ensure a good stability of the control loop and a prompt acquisition of the sensory data.

To simplify the design and the synchronization of the different modules in the program, we used a commercial real-ti[llc operating system called Embedded DOS [9] which provides re-entrant system calls, and functions such as threads, semaphores and protected segments. However, it is worth noticing that this real-time support is not essential for the basic functions of the controller. In fact, we are working on a version of this controller that will not use the commercial package and that will be released in the public domain.

The 1 'C software consists of the following threads: (*i*) A *main* program handling all housekeeping functions to initialize the environment, decode the motion commands, start the real-time clock and the control thread and, at the end of the trajectory, to saw the trajectory data to an ASCII file on disk.

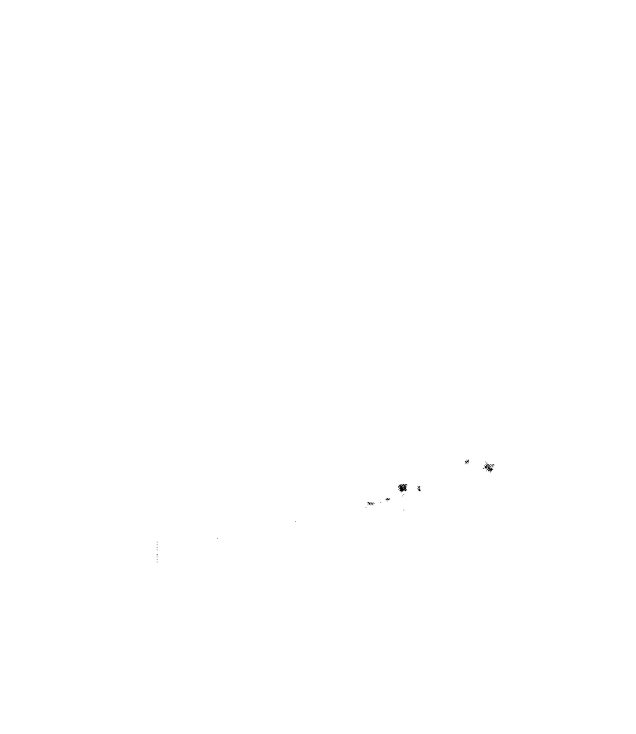


Figure 5: Visual calibration and target acquisition.

Figure 6: Final position of the robot arm.

(*ii*) A *graphical interface* for robot calibration and motion command input. Commands can be specified in joint or Cartesian relative motions, or by selecting the trajectory end point on the image of the robot.
(*iii*) The *control* task activated by the interrupt handler, performing the trajectory interpolation, the velocity limit check, and the communication with the PUMA controller.
(*iv*) The *sensor* acquisition activated by the interrupt handler at a higher frequency than the control task, since force/torque measurements must be completed within a control cycle.

The interaction among the tasks is schematically represented in Figure 4, which also snows the timing of the control functions. The real-time clock runs at a 125 μs interval during the execution of the trajectory, and increments the counter for the 28 ms control algorithm. In Figure 4, the execution flow of the program is shown by the arrows linking the various functions and representing causal relations. The controller software starts with the main task (o), followed by the operator interface (b). When a new motion command is given, the docks (d) and (e) are started, together with the the control (c) and the FT acquisition (f) threads. At (g), when the trajectory is finished, the main turns off the docks, saw the data to a file and enables the operator's interface, which displays the current position of the arm.

The FT sensor processor acquires the eight strain-gauge values in approximately 2 ms, and transmits them to the PC in 250 μs. The acquisition/transmission cycle thus provides new FT data every cycle! of the control algorithm.

4 RCCL port to the PC

The Robot Control C Library (RCCL) consists of two main components: a *C* language *extension* for *robot* motion planning and control, and the Real-Time Control Interface (RCI) package. The former includes programs and data structures for kinematic and planning computation. The latter handles the real-time aspects of robot control and communication. We take advantage of the modularity and quality of RCCL/RCI programs by using some of its components into the software of the PC-based controller.

We use several elements of RCI: the communication program *down*, the PUMA resident software *moper*, and the functions and the data structures to communicate with moper. *Down* overwrites the VAL language with moper. Moper is the software in the LSI-11 computer that handles the data exchange between the joint servo controllers and the PC. It is worth noticing that moper works properly only on a controller that has been freshly initialized by VAL. Since the real-time clock on the PC provides an accurate timing, here moper is used in *slave* mode. The low level communication between the PC software and moper is carried out, using RCI puma-input () and puma_output () routines, which read from, and write data to the structures defined in RCI_RBT, i.e. JLS, HOW, CHG, KIN [4]. The real-time support, task synchronization, and inter-task communication on the PC is carried out by the real-time software described in Section 3.2.

The control thread described in Section 3.2 includes the following RCI functions organized in this order:

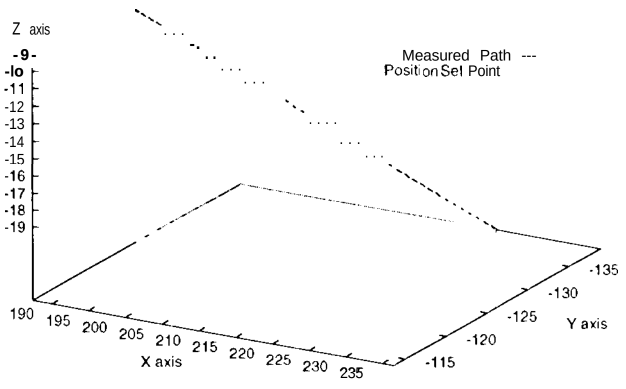


Figure 7: Plot of the position control experiment.

1. `priSignal ()` signals moper the arrival of a new set-point;
2. priRead () reads the current encoders values;
3. puma-input () converts the encoders into joint angles and does a safety check on the input data;
4. `cycleDrive ()` generates the trajectory interpolation;
5. puma_output () converts the setpoints into encoders data and dots a safety check on the output data;
6. priWrite send the setpoints to moper.

The kinematic computations on the PC are carried out using RCCL functions and data structures. The trajectory generator consists of a cycloidal interpolation which suits the application needs better than the original RCCL trajectory generator.

Before starting a motion, the trajectory is computed off-line to check for possible singularities and to verify that joint velocities are within the safety limits.

5 Experimental Results

In this section, we report on our current tests with the PUMA 260 manipulator and the PC-based controller. The system start-up is still quite involved and includes the following steps:

(i) Power up the PUMA controller and load VAL to the PUMA controller,

(ii) Calibrate the robot using VAL and acquire the PUMA joint angles,

(*iii*) Download moper to the PUMA controller,

(*iv*) Start the! PC-based controller, initialize PC and moper data structures with the robot joint current values

An approximate calibration of the visual interface is then carried out by the operator by selecting on the computer screen the base and the tip of the surgical needle, as shown in Figure 5. The PC then acknowledges the axis selection by drawing a line on the needle in the camera picture. By knowing the robot orientation with respect to the camera and the length of the needle, it is possible to compute the scale factor of

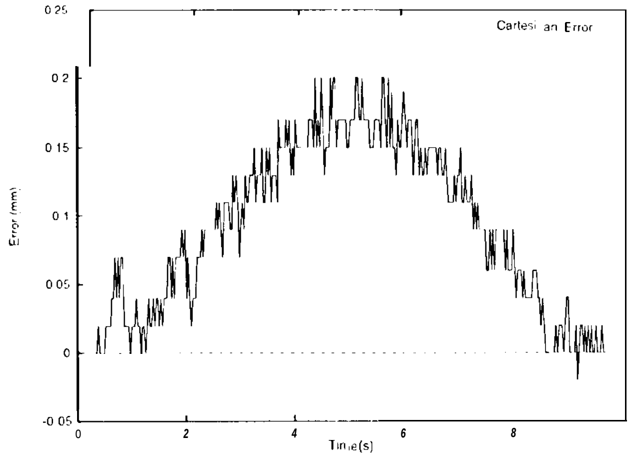


Figure 8: Plot of the tracking error.

the image and use the mouse to define a linear trajectory for the robot. The user specifies the end-point of motion for the robot by clicking with the mouse on a point on the needle axis. With the current set,-uIJ, the screen resolution corresponds to 1 mm/pixel. The start motion command is then given by typing the corresponding text string.

A typical experiment is shown in Figures 5 and 6. Figure 5 shows the cross-hair marking the selection of the needle tip and of the trajectory end point. Figure 6 shows the end of the trajectory with the manipulator reaching the cross-hair marking the target point. The command given using the visual interface corresponds to a 5 cm motion on the XZ plane, along the line defined by the current orientation of the surgical needle, and with a velocity of 10 mm/s. The initial position of the arm is ($x = 187.39\,mm, y = -126.24\,mm, z = -9.81\,mm$), and the orientation remains constant during the motion. The final position of the arm is ($x = 236.56\,mm, y = -126.24\,mm, z = -18.76\,mm$).

Figure 7 snows the plot of the Cartesian setpoint and of the robot arm motion during the trajectory of Figure G. The purpose of these tests is primarily to evaluate the effects of the trajectory generation and communication protocol on the tracking performance of the arm. This plot shows that the arm tracks the cycloidal Cartesian motion trajectories accurately, and demonstrates the good tracking performance of the PC-based controller. The maximum tracking error for the experiment shown in Figures 5 and 6 is equal to 0.2 mm, and is typical of trajectories 10 to 15 cm long, traversed with the velocities of 3 to 10 mm/s, used in these surgical procedures.

During the experiments, data are processed and presented to the operator using a set of MatLab procedures [2]. In the current implementation, the control Cycle of the PC-based controller is limited to 28.0 ms by the update time of the PUMA controller.

Since the computation and communication cycle on the PC takes approximately 8 ms, the controller dots not introduce any degradation to the robot arm performance. By making the appropriate changes to the parameters of the PUMA joint controllers, the control cycle of the PC-based controller can in theory be reduced to 8 ms, thus significantly improving system performance.

6 Conclusions

A robotic workstation for simple surgical procedures is described in this paper. The system is PC-based and uses a PUMA 260 manipulator, which is well suited for simple tasks requiring precise position control and force monitoring, and a small system envelope for easy transport and installation in the operating room. The proposed system responds to the need of the medical community for a PC-based high-performance motion controller for PUMA manipulators that can support an open architecture with different input devices, such as X-ray machines and image analysis programs.

The use of a robotic device in the surgical environment will have a number of significant advantages for the practicing surgeon. For example, it will provide quantitative force information to the surgical procedure, it will ensure that, in the case of the discectomy, the guide needle will reach the target in a single motion, and it will provide a higher level of repeatability than is currently achievable.

The experiments carried out so far show that the PC-based controller is able to handle successfully the control of PUMA arm and acquire force and torque data. Furthermore, the implementation of the controller software shows that robot control can take advantage of the large body of public domain software available, such as RCCL for kinematic analysis and PUMA communication.

In the future, we plan to emphasize the integration aspects of the design, to simplify the robot operation, and the precision tests during realistic operating room experiments. We will also improve the overall safety of the system, by developing appropriate calibration procedures, force monitoring when using the flexible surgical needle, kinematic analysis of trajectory singularities, and external sensors.

7 Acknowledgment

The research described in this paper has been carried out at the Istituti Ortopedici Rizzoli. Part of the work has been done at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. The authors are grateful to Dr. Ivan Loreti and Dr. Maurilio Marcacci for their definition of the surgical requirements, and to Dr. John Lloyd for his support with RCCL.

References

[1] Anonymous. Controlling the PUMA robot arm without using VAL. Technical report, Unimation Inc.

[2] P. I. Corke. A robotics toolbox for MATLAB. *IEEE Robotics and Automation Magazine*, pages 24 32, March 1996.

[3] P. Fiorini, H. Seraji, and M. Long. A Pc-based controller for dexterous manipulators. IEEE *Robotics and Automation Magazine*, pages 1-9, September 1997.

[4] V. Hayward and J. Lloyd. RCCL user's guide. Technical report, McGill Research Centre for Intelligent Machines, Montreal, Quebec, Canada, 1992.

[5] John Lloyd. RCCL/RCI hardware installation notes. Technical report, McGill Research Centre for Intelligent Machines, Montreal, Quebec, Canada, 1992.

[6] John Lloyd. RCCL/RCI startup and installation guide. Technical report, McGill Research Centre for Intelligent Machines, Montreal, Quebec, Canada, 1992.

[7] S. Martelli, M. Di Silvestre, P. Dario, M. Fadda, M. Marcacci, and S. Giannini. A Proposal for a semi-automatic procedure for percutaneous discectomy. In *International Conference on Advanced Mechatronics (ICA M'93)*, Tokyo, Japan, November 1-3 1993.

[8] J. Mills et al. Development of a robot control test platform. *IEEE Robotics and Automation Magazine*, 2(4):21 28, December 1995.

[9] General Software. Embedded dos user's manual. 1996.

[10] R. Vistnes. Breaking away from VAL, or how to use your PUMA without using VAL. Technical report, Unimation Inc., 1981.