EXPERIMENTAL STUDY OF AN UNDERACTUATED MANIPULATOR

Marcel Bergerman Christopher Lee Yangsheng Xu

CMU-RI-TR-95-16

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

April, 1995

This research is partially sponsored by the Brazilian National Council for Research and Development (CNPq). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of CNPq or Carnegie Mellon University.

Table of Contents

List of Figures

List of Tables

Abstract

Underactuated manipulators are a class of robotic mechanisms where passive joints are present. By controlling only the motion of the active joints, it is possible to control the entire system. Our goal is to develop control schemes using both classical nonlinear and modern learning techniques for underactuated manipulators. To examine the validity of the approaches, we developed an experimental setup known as U-ARM, or UnderActuated Robot Manipulator. In this report, we present the hardware development, dynamic parameters derivation, control software and experimental results of real-time control of U-ARM.

1 Introduction

In recent years, there has been an increased interest in the study of underactuated manipulators. These are mechanisms where not all joints are equipped with control actuators. Joints bearing actuators are called *active joints*, and the remaining ones are called *passive joints*. For control purposes, all joints are equipped with position sensors. The passive joints are equipped with brakes, so that they can be locked if needed. The passive joints' angles or displacements are not directly controllable due to the lack of an actuator. However, in case there is sufficient dynamic coupling between the active and the passive joints, control schemes can be devised to drive the passive joints using the torques available at the active ones.

In order to better understand the behavior of such systems, it is necessary to experiment with real mechanisms. Their sensitivity to dynamic disturbances is generally greater than that of fully-actuated manipulators. Control schemes based on simulations may not work when later transferred to the actual system. In order to demonstrate the validity of the robust controller we proposed for underactuated manipulators [3], we built a two-link system with one passive joint. The mechanism is called U-ARM, or UnderActuated Robot Manipulator.

In this report we present the hardware development of the U-ARM, and the derivation of the dynamic parameters of the mechanism: link lengths, centers of mass, masses and inertias. We describe the real-time control architecture written under the operating system Chimera 3.2, developed at CMU. Experimental results of control in joint space with a variable structure controller are presented to demonstrate the feasibility of the method. Future research is pointed out at the report.

2 Robot Hardware

Our first step in the experimental study of underactuated manipulators was to develop a two link mechanism with one passive joint. This is the simplest possible underactuated manipulator that can be controlled. Because the design is modular, it will be easy to extend the concept to mechanisms with more than two joints in the future.

The underactuated manipulator developed at Carnegie Mellon is pictured in Figure 1, and shown schematically in Figure 2. Both joints have incremental encoders for sensing their angular positions. The angular velocities are obtained via numerical differentiation and low-pass filtering.

The first joint is actuated via a DC motor with a gear reduction of 60:1, and the second joint is equipped with a brake.

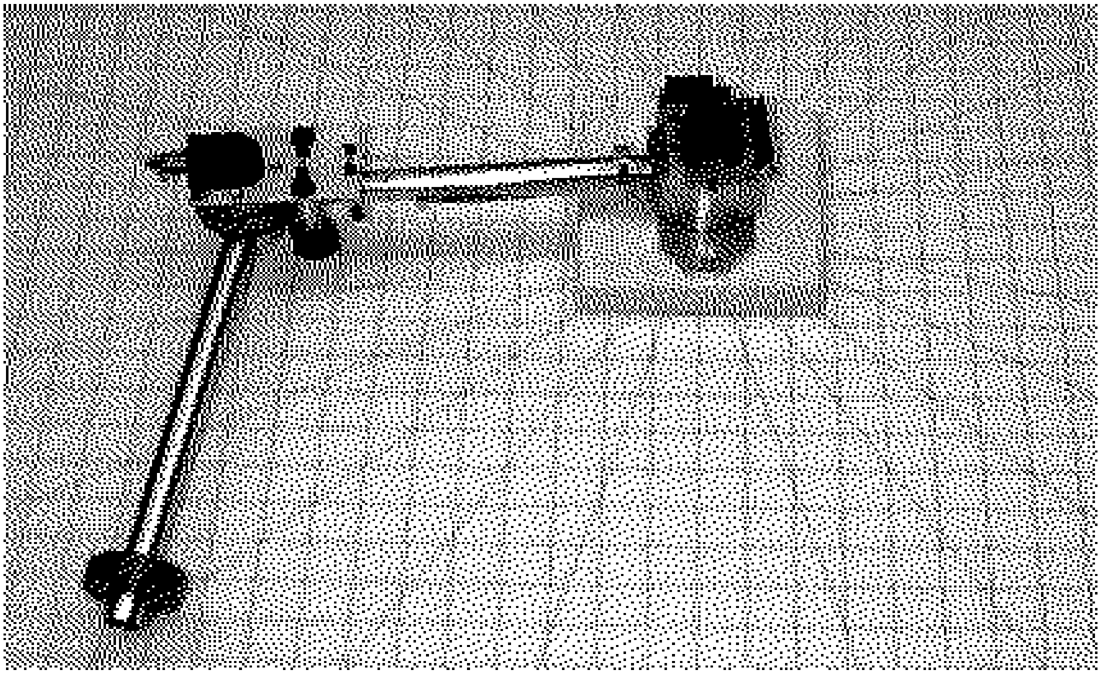

Figure 1: Underactuated manipulator U-ARM.

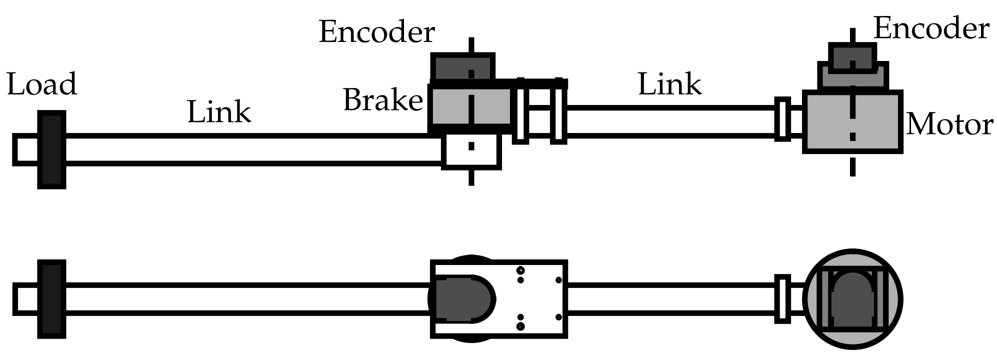

Figure 2: Schematic representation of U-ARM.

The DC motor used in the first joint is of the same type used in the Self Mobile Space Manipulator (SM^2) and the Dual-use Mobile Detachable Manipulator (DM^2), both developed in previous years at CMU. A joint limiter is incorporated in the motor interface, turning off the power to the motor when the joint angle is close to 180^o. This eliminates the need for a mechanical joint limiter.

The brake used in the second joint is an on/off brake with rated static torque 50 lb-in, operating on a 24 VDC voltage. It is manufactured by Inertia Dynamics (model number FB-22.) In order to avoid damage to the manipulator, a mechanical joint limiter was devised for the second joint. This consists of a set of two rubbermade cylinders, which give the joint an excursion of 240^{o} degrees (from -120^{o} to $+120^{o}$). The encoders mounted at each joint are manufactured by Hewllet Packard, model HEDS 5540; which generate 1024 counts for each complete joint revolution.

3 Dynamic Parameters

The dynamic parameters of the experimental robot are important for model-based control schemes, and for verification of the robustness to uncertainties in the parameters when different control methods are compared.

In order to compute each parameter, we weighed and measured each individual component of the mechanism. The total number of components is 16, as depicted in Table 1. The radius of gyration indicates the distance between the component and the respective link axis. Note that components 1 to 12 belong to link 1, while those numbered 13 to 15 belong to link 2 (the load was not considered as belonging to link 2; if it is desirable, the load's mass and inertia can be easily added to the link's ones, and the center of mass will have to be recomputed).

3.1 Links' Length

These are obtained by directly measuring the links of the manipulator. The results are given in Table 2.

3.2 Links' Mass

The masses of the links were obtained after every component in the mechanism was weighed. Adding the masses of the components belonging to each link, the results shown in Table 3 were obtained.

3.3 Links' Center of Mass

In order to compute the centers of mass, each separate component (considered as a point mass) and its radius of gyration with respect to its link axis must be considered. Let μ_i be the mass of each component as shown in Table 1, and λ_i its radius of gyration with respect to its link axis. Then the center of masses can be computed as follows:

Table 1: Component-wise design of U-ARM.

Link	Part Number	Part Name	Mass (μ_i, 10^{-3} Kg)	Radius of gyration w.r.t. link (λ_i, 10^{-3} m)
Link 1	1	Motor	543.0	0.0
	2	Support		
	3	Encoder	7.0	0.0
	4	Support	9.5	47.7
	5	Link	20.2	129.0
	6	Support	14.3	194.3
	7	Support	14.1	229.1
	8	Joint Limiter	8.3	229.1
	9	Joint Limiter	8.3	229.1
	10	Support	33.0	235.9
	11	Encoder	7.0	261.1
	12	Brake	391.7	255.5
Link 2	13	Brake	10.0	0.0
	14	Axle	20.0	0.0
	15	Link	31.1	167.8
Load	16	Load	227.0	289.0

Table 2: Links' lengths.

Link	Length (l_{i}, 10^{-3} m)
1	255.5
2	289.0

$$l_{c_1} = \sum_{i=1}^{12} \frac{\mu_i \lambda_i}{\mu_i} \qquad\qquad l_{c_2} = \sum_{i=13}^{15} \frac{\mu_i \lambda_i}{\mu_i}. \tag{1}$$

If the load is considered as belonging to link 2, then the superior index in the second summation should be changed to 16. The use of these formulas lead to the results in table 4.

Table 3: Links' masses.

Link	Mass (m_i, 10^{-3} Kg)
1	1063.4
2	61.1
2 (with load)	288.1

Table 4: Links' center of masses.

Link	Center of mass (l_{c_i}, 10^{-3} m)
1	115.8
2	85.4
2 (with load)	245.8

3.4 Links' Inertia

We also considered each component separately for the computation of the total inertia of the links. The manipulator is planar, so the inertia tensor reduces to a scalar quantity. Components 4 through 12, 15 and 16 were considered point masses localized at their radius of gyration; components 1 and 2 were considered a single cylinder; components 13 and 14 were also considered cylinders; finally, components 2 and 3 were considered parallelepipeds. Table 5 presents the individual inertias of each component of U-ARM. As it can be seen, the major contribution for the first link's inertia comes from the brake housing, mounted at the end of the link - 83% of the total. Likewise, when the load is considered to be part of link 2, it contributes with 96% of the link's inertia.

With these results, the total inertias for links 1 and 2 can be obtained as shown in Table 6.

4 Kinematics and Dynamics

In order to perform joint and Cartesian control with the U-ARM, we need both the kinematic and dynamic models for control purposes. In this section we present their derivation. We assume here that U-ARM can be modelled as a set of two rigid links rotating on an horizontal plane, as shown in Figure 3.

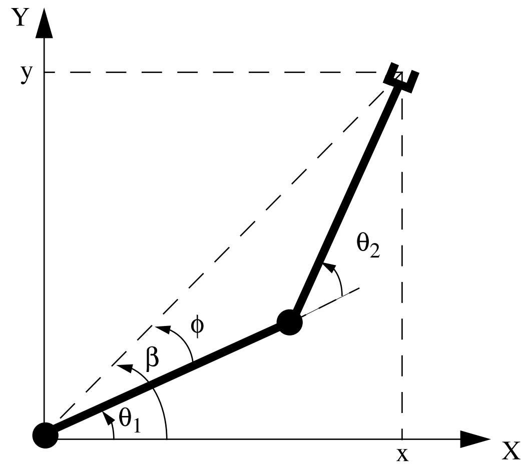

Figure 3: Rigid body model of U-ARM.

Table 5: Component-wise inertia distribution of U-ARM.

Part Number	Inertia (10^{-6} Kg m^2)
1	296.0
2	
3	1.3
4	21.6
5	336.2
6	539.9
7	740.1
8	435.6
9	435.6
10	1836.4
11	477.2
12	25570.3
13	4.1
14	3.7
15	875.7
16	18959.3

Table 6: Links' inertias.

Link	Inertia (I_i, 10^{-3} Kg m^2)
1	30.69
2	0.88
2 (with load)	19.84

4.1 Kinematic Model

From Figure 3, we can write the forward kinematic equations of the mechanism:

$$\begin{aligned} x &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \\ y &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \end{aligned} \tag{2}$$

Given the Cartesian coordinates *x* and *y*, these equations can be solved for θ_1 and θ_2. This is the so-called inverse kinematics problem, whose solution in this case can be found in [4]:

$$\begin{aligned} \theta_2 &= \operatorname{acos}\left(\frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right) \\ \theta_1 &= \beta - \phi \operatorname{sgn}(\theta_2) \end{aligned} \tag{3}$$

where the auxiliary angles β and ϕ are given by:

$$\begin{aligned} \beta &= \operatorname{atan2}(y, x) \\ \phi &= \operatorname{acos}\left(\frac{x^2 + y^2 + l_1^2 - l_2^2}{2 l_1 \sqrt{x^2 + y^2}}\right) \end{aligned} \tag{4}$$

In (4), the arccosine function must be computed so that $0 \le \phi \le 180^o$. Note that there are two possible solutions for *q*, namely, the positions known as "elbow up" — $\theta_2 < 0$ — and "elbow down" — $\theta_2 > 0$. Care must be taken when the sign of θ_2 in chosen in (3), so that the geometry of the manipulator is coherent with the solution found.

4.2 Dynamic Model

Using the Lagrangian formulation, one can write the dynamic equations of a system if the kinetic and potential energies of the system, respectively K and U, are known as a function of a set of generalized coordinates. In this case, $q = \begin{bmatrix}\theta_1 & \theta_2\end{bmatrix}^T$ is such a set of coordinates. Because the manipulator lies on an horizontal plane, the potential energy is constant and does not depend on the joint angles. So we have:

$$\begin{aligned} K(q, \dot{q}) &= \frac{1}{2}\dot{q}^T M(q)\,\dot{q} \\ U(q) &= U_0 \end{aligned} \tag{5}$$

where:

$$M(q) = \begin{bmatrix} m_1 l_{c_1}^2 + m_2\left[l_1^2 + l_{c_2}^2 + 2 l_1 l_{c_2}\cos(\theta_2)\right] + I_1 + I_2 & m_2\left[l_{c_2}^2 + l_1 l_{c_2}\cos(\theta_2)\right] + I_2 \\ m_2\left[l_{c_2}^2 + l_1 l_{c_2}\cos(\theta_2)\right] + I_2 & m_2 l_{c_2}^2 + I_2 \end{bmatrix} \tag{6}$$

The Lagrangian formulation gives the relationship between the generalized coordinates q and their respective generalized forces τ as:

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}}\right) - \frac{\partial L}{\partial q} = \tau \tag{7}$$

where $L = K - U$ is the Lagrangian of the system. Substituting (5) in (7) and carrying out the computations, we get:

$$M(q)\,\ddot{q} + C(q, \dot{q})\,\dot{q} = \tau \tag{8}$$

The matrix $C(q, \dot{q})$ is given by:

$$C = m_2 l_1 l_2 s_2 \begin{bmatrix} -2\dot{\theta}_2 & -\dot{\theta}_2 \\ \dot{\theta}_1 & 0 \end{bmatrix} \tag{9}$$

5 Dynamic Coupling

Our goal is to control the angle of U-ARM’s passive joint using only the torque available at the active joint. It is important to examine whether there is sufficient dynamic coupling between these joints, so that the passive joint can be controlled through its coupling with the active one. Low or inexistent coupling will make it impossible for the control algorithm to perform its function properly.

The dynamic coupling between the passive and the active joints of a generic n-link manipulator can be quantified by the study of the relationship between the accelerations generated at the passive joints given the available accelerations at the active joints. Considering only the inertial effects of the mechanism, we can write [2]:

$$\bar{\ddot{q}}_p = M_c \ddot{q}_a \tag{10}$$

where $\bar{\ddot{q}}_p$ represents the "virtual" accelerations of the passive joints, $\ddot{q}_a$ represents the accelerations of the active joints, and M_c is a matrix obtained from partitioning the inertia matrix of the manipulator, $M(q)$, into active and passive subsystems.

For a two-link manipulator, when the first joint is active and the second one passive, the matrix M_c is given by:

$$M_c^{ap} = -\left[1 + \frac{m_2 l_1 l_{c_2} \cos(\theta_2)}{m_2 l_{c_2}^2 + I_2}\right] \tag{11}$$

where the superscript *ap* denotes that the joints are, in this order, active and passive. In the converse case, we have:

$$M_c^{pa} = -\frac{m_2\left[l_{c_2}^2 + l_1 l_{c_2} \cos(\theta_2)\right] + I_2}{m_1 l_{c_1}^2 + m_2\left[l_1^2 + l_{c_2}^2 + 2 l_1 l_{c_2} \cos(\theta_2)\right] + I_1 + I_2} \tag{12}$$

The coupling index, a measure of the dynamic coupling, is given by the product of the singular values of M_c:

$$\rho_c = \prod_{i=1}^{\min(r,p)} \sigma_i(M_c) \tag{13}$$

Substituting the values obtained in Section 3, we obtain:

$$\begin{aligned} \rho_c^{ap} &= \left|1 + 1.00571 \cos\theta_2\right| \\ \rho_c^{pa} &= \left|\frac{1 + \cos\theta_2}{37.78 + 2\cos\theta_2}\right| \end{aligned} \tag{14}$$

Figure 4 shows the evolution of both ρ_c^{ap} and ρ_c^{pa} as a function of θ_2. Note that, because of the joint limiters on joint 2, the coupling index never reaches zero. Note also that the coupling index is much greater for the case when joint 1 is active. This is because the second link's mass and inertia are too small to drive the first joint. For this reason, we chose to build U-ARM with its first joint active.

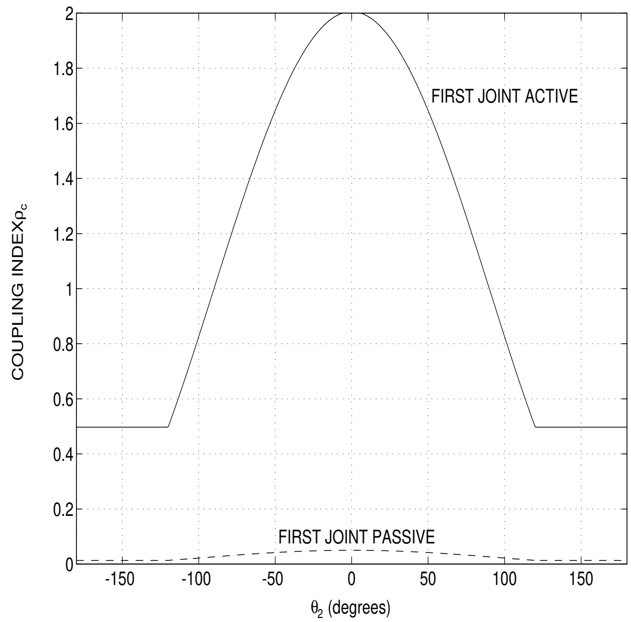

Figure 4: Coupling index of U-ARM.

6 Control Software

The U-ARM runs a software control system based on the Chimera 3.2 operating system, a multiprocessor real-time operating system which supports software written as a group of independent modules [10], [11]. The basic architecture of the control software was taken from the $(DM)^2$ robot, but the modular design of this software made the adaptation straightforward.

The software architecture consists of two levels — a high-level system which interprets commands from a host workstation and monitors the execution of these commands, and a low-level system in which the basic control system is built from communication between a set of independent real-time software modules.

The low-level real-time modules each run as one (or more) separate threads, cycle at independent frequencies, and can be allocated at runtime to any of the processors on the system without modification. Modules can communicate through a variety of means, but real-time control modules generally exchange data at the beginning and end of each execution cycle through Chimera's state variable table mechanism. For example, a trajectory-generation module like that show in Figure 5 will typically output reference joint positions to a state variable at the end of each cycle, and a controller module will input state-variables representing reference and measured joint positions, and output actuator-torque values to a state variable at the end of each cycle. The high-level system communicates with these modules through a high-speed message-passing system.

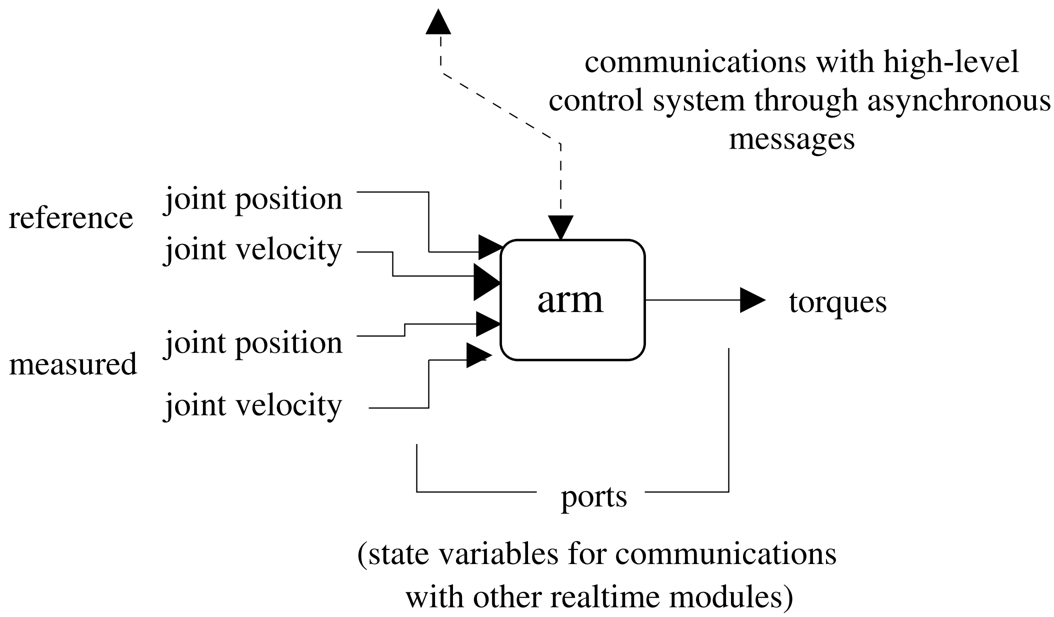

Figure 5: Trajectory-generation module's inputs and outputs.

The purpose of the low-level real-time controller modules is to provide strong controller performance while insulating higher-level modules from the details of robot's hardware configuration. The U-ARM uses a simple arrangement of these modules, shown in Figure 6. The modules which interface to the robot hardware are the encoder and actuator modules. The encoder module writes joint positions to the state variable table, and the actuator module writes torque-values from the state variable table to the joint actuator and the joint brake. A trajectory-generation module provides reference-values for the joint positions and velocities, and the controller module reads measured and reference joint positions from the state variable table and outputs torque values to the table.

Both the trajectory-generator module and the controller module operate by using a set of *control-objects* which are selected from a library of objects at runtime. Trajectory-generation objects can correspond to functions such as set-point generators, linear-interpolators, or cubic

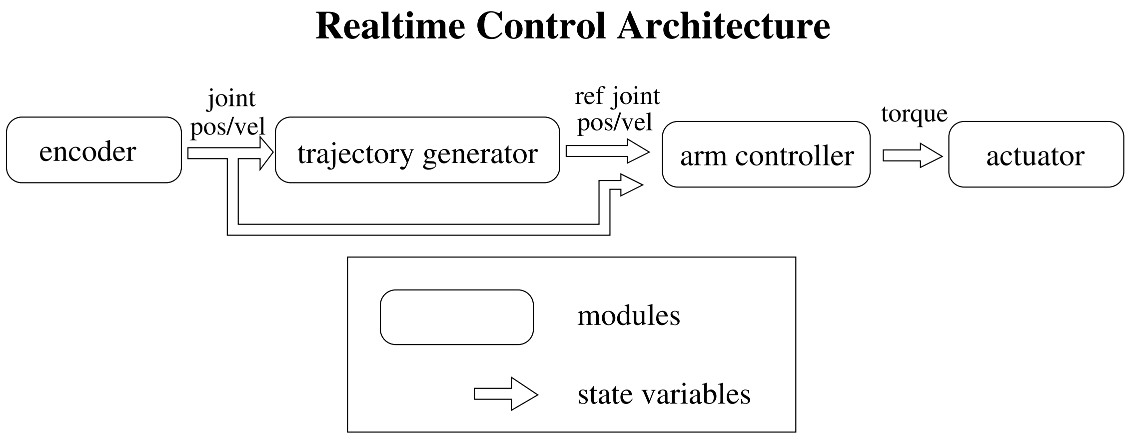

Figure 6: Combination of modules for the control of U-ARM.

spline followers. Real-time controllers can be created through combinations of control objects such as PID controllers, variable structure controllers, and torque limiters. By indicating which control objects to use, a high-level system is able to specify which trajectory-generation or control strategies to use at a particular time. Default controllers and the parameters of these controllers are specified in a configuration file which is read at the beginning of the software's execution.

Creation of new control-objects is facilitated by a code-generation tool which reads information about the controller from the developer and then uses this information to create a source-code skeleton from a template file. Filling-in the code skeleton to produce a functional control-object takes just a couple of minutes. Two lines of code are needed to interface the new object to the library of usable control-objects for a given module.

High level control within the software system is provided by a software library consisting of a text-based command-parser and a separate thread which executes these commands and checks their status at a fixed cycle rate.

This library is designed as a framework for high-level code to exist within a Chimera program. The command parser allows a user to specify tasks for the robot to perform. These commands are grouped recursively into lists of commands that are to be executed either sequentially or in parallel. This way of scheduling tasks allows for simple specification of complicated jobs consisting of subtasks which may be executed quickly and in parallel when possible, or sequentially if later tasks depend on the completion of previous tasks.

High-level commands are defined in terms of a combinations of *atomic* tasks, which together span the technical ability of the robot. These atomic commands are written as C++ classes, and are called by passing commands to the interpreter which resemble function calls in the C-language.

The command parser/executor combination is not a true programming language in its current form, although it has some constructs similar to common programming languages, and can easily be extended to handle more. The main elements which the system shares with many programming languages are the ability to specify new procedures from combinations of existing commands, and the ability to pass parameter lists to these procedures. Some basic elements of a programming language that are currently lacking in the parser/execution system are expressions with values, variables, looping structures, and conditional expressions. Because the system is written as a combination of a lex/yacc grammar and a hierarchy of C++ objects, the language may be extended cleanly by subclassing the necessary language-feature objects and linking them to modifications in the yacc grammar. In fact, the fundamental language mechanisms are implemented within the same C++ class hierarchy as the atomic robot-commands, and may be added to the system in exactly the same manner as these atomic commands.

Atomic robot-commands perform hardware actions by sending messages to the appropriate real-time modules. It is, however, undesirable for each command-macro to worry about interprocess communication with the relevant controller modules and all of the complexities, such as preserving state-information and avoiding race conditions, that this communication implies. It is also undesirable for two or more commands running synchronously to be allowed to give potentially conflicting commands to the same controller-module.

To avoid these problems, C++ objects representing *virtual hardware* were created to represent sets of hardware control-modules. Each virtual hardware object provides a limited number of handle-objects (typically one) to command-atoms that can be used to perform operations with the hardware. The virtual hardware objects are responsible for communication with the real-time modules and for maintaining state information about the robot for use by command-atoms.

This approach lets authors of command-atoms concentrate on functionality rather than interprocess communication. It also centralizes data about the entire robot within the task-execution system and provides a consistent interface for accessing that information. Limiting access to virtual objects through handles avoids situations such as two conflicting commands being sent to the base-trajectory generator at once.

Unlike the $(DM)^2$, the U-ARM has a fairly simple hardware configuration, so only one virtual hardware object was needed to represent the manipulator arm.

7 Joint Control Technique

In [1], a PID-based scheme similar to the computed torque method is used to bring the passive joints to a desired set-point via the dynamic coupling with the active ones. A similar approach was used in [8], where the objective was to bring a failed joint of a space manipulator to a desired position. In both schemes, accurate dynamic models were assumed to be known. An important theorem was established in [7], which states that no smooth control law can achieve stabilization of both active and passive joints to an equilibrium point. One must either target for an equilibrium manifold, or derive noncontinuous control laws (a result yet to be achieved.)

Studies [1], [7], [8] have shown that the control of underactuated manipulators strongly depends on an accurate dynamic model of the system. We proposed a robust control technique in [3] which consists on a variable structure controller (VSC) that forces the state trajectory to converge to a pre-defined sliding surface in the state space. We will present here the results of the application of this technique on U-ARM.

In this control scheme, the dynamic equation of the manipulator is partitioned into active and passive subsystems. Because the torque at the passive subsystem is zero, it is possible to express the acceleration of the active joint as a function of the acceleration of the passive one. Substituting this expression into the active subsystem equation, one can obtain the relationship between the joint torque and passive joint acceleration. A sliding surface is then chosen in the state space of the passive joint, and a variable structure controller designed to force the state trajectories to converge to this surface. This guarantees that the passive joint converge to its desired set-point, where it can be locked. Then, the active joint can also be controlled to its desired position, and joint control of the underactuated system is completed.

The experiments performed with U-ARM are shown in Table 7; all angles are in degrees. Figures 7 to 9 present the respective responses; the dotted lines represent the passive joint, while the full ones represent the active joint.

As can be seen from the figures, the behavior of the manipulator under the VSC control scheme is fast and stable. First the active joint brings the passive one to its desired set-point with approximately zero velocity, then it converges to its own set-point smoothly along the sliding surface. One point to note is that the control of the passive joint in experiment 3 was slower than in 1 and 2. This is due to the fact that the table where U-ARM is installed is not perfectly levelled. This creates a small but disruptive gravitational torque that pulls the manipulator to one side. Nonetheless, the VSC was robust enough to cope with this problem and still control the system.

Table 7: Joint control experiments performed.

Experiment	Initial angles	Set-point	Steady-state error
1	$\begin{bmatrix} 0 \\ -88 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.057 \\ -0.172 \end{bmatrix}$
2	$\begin{bmatrix} -37 \\ -53 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} -0.057 \\ -0.343 \end{bmatrix}$
3	$\begin{bmatrix} -50 \\ 88 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0.057 \\ 0.000 \end{bmatrix}$

Another point that deserves comment is the fact that the motor in the first joint of U-ARM has a significant static friction coefficient. This friction was not compensated for in our control software, and was considered as a disturbance to be rejected by the controller. As we saw, the controller was effective in rejecting this disturbance.

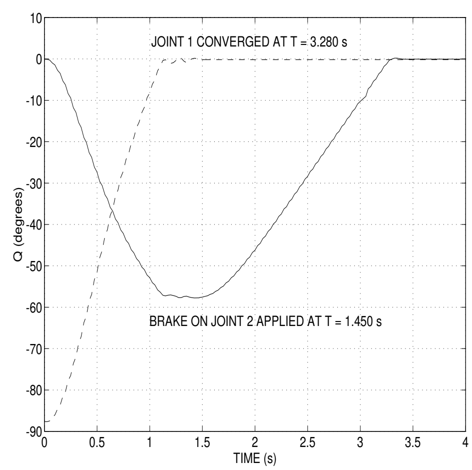

Figure 7: Joint control of the U-ARM, Experiment 1.

8 Conclusion

In this report we presented the mechanical design, control software, and robust control of an experimental underactuated manipulator. The mechanism has two joints, the first one chosen to be active on a rationale of maximized dynamic coupling. Real-time control was implemented using the real-time operating system Chimera 3.2. The experimental results demonstrated the validity of

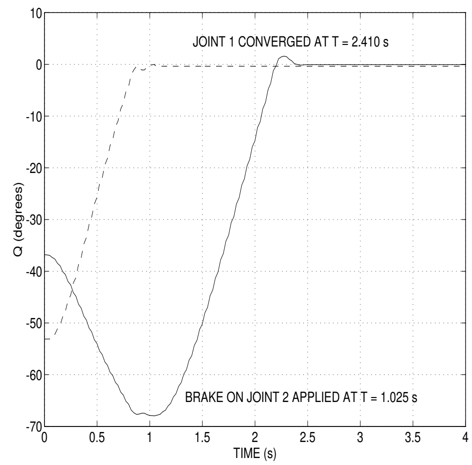

Figure 8: Joint control of the U-ARM, Experiment 2.

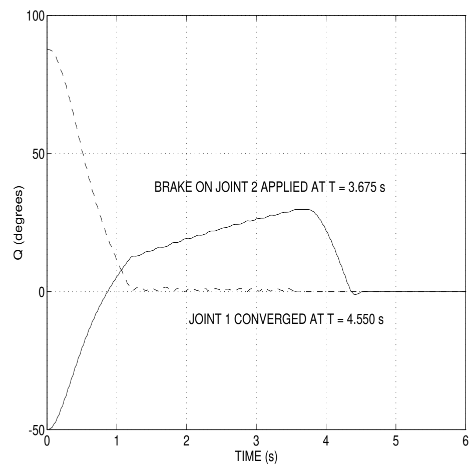

Figure 9: Joint control of the U-ARM, Experiment 3.

the proposed variable structure controller, which was able to control the mechanism despite friction in the motor joint and residual gravitational torque.

We demonstrated that the choice of the first joint as the active one maximizes the dynamic coupling between the active and the passive joint, thus making control of the passive one easier. However, if one is interested in Cartesian space control of underactuated manipulators, it may be more important to consider the actuability of the mechanism [5]. Our future research will probably include the assembly of a four-link manipulator with joints that can be chosen to be either active or passive at the beginning of each experiment. This way, more interesting experiments can be performed, and a better understanding of this class of systems can be gained.

9 Acknowledgments

This work was possible partially due to a grant from the Brazilian National Council for Research and Development (CNPq).

10 References

[1] Arai, H.; Tachi, S. "Position control of a manipulator with passive joints using dynamic coupling." *IEEE Trans. on Robotics and Automation*, vol. 7, no. 4, Aug. 1991, pp. 528-534.

[2] Bergerman, M.; Lee, C.; Xu, Y. "*Dynamic coupling of underactuated manipulators*." CMU-RI-TR-94-25, Carnegie Mellon University, Aug. 1994.

[3] Bergerman, M.; Xu, Y. "Robust control of underactuated manipulators: analysis and implementation." *Proceedings of the IEEE Systems, Man and Cybernetics Conference*, Oct. 1994, pp. 925-930.

[4] Craig, J.J. *Introduction to robotics*: mechanics and control (2nd ed.) Reading: Addison-Wesley, 1989.

[5] Lee, C.; Xu, Y. *Actuability of underactuated manipulators*. CMU-RI-TR-94-13, Carnegie Mellon University, 1994.

[6] Lee, C.; Xu, Y. Control architecture of $(DM)^2$. Submitted to the International Conference on Robotics and Automation, 1995.

[7] Oriolo, G.; Nakamura, Y. "Control of mechanical systems with second-order nonholonomic constraints: underactuated manipulators." *Proc. of the 30th Conference on Decision and Control*, Dec. 1991, pp. 2398-2403.

[8] Papadopoulos, E.; Dubowsky, S. "Failure recovery control for space robotic systems." *Proceedings of the 1991 American Control Conference*, vol. 2, 1991, pp. 1485-1490.

[9] Saito, F.; Fukuda, T.; Arai, F. "Swing and locomotion control for a two-link brachiation robot." *IEEE Control Systems Magazine*, Feb. 1994, pp. 5-12.

[10] Stewart, D.; Volpe, R.; Khosla. P.K. Integration of Real-Time Software Modules for Reconfigurable Sensor-Based Control Systems. *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*", 1992, pp. 325-332.

[11] Stewart, D.; Volpe, R.; Khosla, P.K. *Design of Dynamically Reconfigurable Real-Time Software using Port-Based Objects*. CMU-RI-TR-93-11, Carnegie Mellon University, 1993.