TECHNICAL RESEARCH REPORT

Experimenting with Hybrid Control

by R.W. Brockett, D. Hristu

CDCSS T.R. 2000-3
(ISR T.R. 2000-16)

The Center for Dynamics and Control of Smart Structures (CDCSS) is a joint Harvard University, Boston University, University of Maryland center, supported by the Army Research Office under the ODDR&E MURI97 Program Grant No. DAAG55-97-1-0114 (through Harvard University). This document is a technical report in the CDCSS series originating at the University of Maryland.

Web site http://www.isr.umd.edu/CDCSS/cdcss.html

Experimenting with Hybrid Control

Roger W. Brockett Dimitrios Hristu

Introduction

The wide availability of experimental and numerical environments for breathing life into the various control theories found in textbooks has greatly changed the educational experience of typical students of automatic control. Nonlinear balance beams, inverted pendulums, and distributed parameter thermal systems are now widely available for hands-on experimentation. Many students react quite positively to this additional dose of realism. Because the models selected for such experiments are usually accurately described by relatively simple differential equations, the laboratory experience reinforces both the textbook analysis and the value of numerical simulation.

At the same time, there is a growing realization among educators and employers that students of automatic control should be encouraged to think of the subject in broader terms. The systems approach should embrace communication requirements, signal processing, data logging, etc. all the way up to and including the level of complexity suggested by the phrase, *enterprise control*. Designing a control experiment that is illustrative and instructional in this broader sense presents a number of challenges beyond those discussed above. The systems under consideration must be very flexible. Of course the hardware must continue to be reliable and relatively easy to understand at an intuitive level. They should also reflect the complexity of purpose and the possibility of multi-modal operation that one expects to find in complex systems. With these qualities in mind, we have assembled and extensively exercised an experimental hybrid control system for use in an instructional/research laboratory at Harvard. Our goal with this paper is to describe for others the structure of the system and to present a sample of the experiments that were facilitated by it.

An important feature of the facility we describe is that it uses several types of sensing modalities including position sensing, tactile sensing and more conventional vision sensing. It can interact with objects of different complexity and is subject to communication constraints arising in a completely natural and generic way. In constructing it we have used off-the-shelf components wherever possible and made choices with an eye toward flexibility and reliability.

Electro-mechanical Hardware

The manipulator pictured in Fig. 1 is a two-fingered hand and was developed in the Harvard Robotics Lab. The manipulator consists of a pair of fingers, made as "mirror images" of one another, mounted $5.4cm$ apart over a desktop. Each finger has two rotational degrees of freedom, allowing for planar motion parallel to the desktop. A third degree of freedom allows for vertical motion of the entire finger. The proximal and distal links have lengths of $12.9cm$ and $5.6cm$ respectively. A fingertip-like tactile sensor is attached to a mounting plate at the end of each distal link. The total length of the distal link and fingertip assembly is approximately $11.8cm$. The finger workspace is approximately $10cm$ tall with elliptical base whose major and minor axes that are $15cm$ and $12cm$ long. That workspace is imaged by a camera that is mounted above the manipulator. The images obtained from the overhead camera are similar to the one shown in Fig. 1 and can be used to track objects during manipulation tasks.

Finger joints are driven by brushless DC motors, located behind the fingers. The maximum continuous torque output for each motor is $25oz-in$ ($100oz-in$ peak) with a motor constant of $10.8oz-in/A$. Each motor contains integrated A/D and D/A electronics as well as shaft encoders (2000 counts/rev) for position and velocity measurements. An on-board microprocessor implements a PID controller that can servo on the shaft position or velocity with programmable feedback and feed-forward gains. This internal controller accepts user-supplied commands that are used to determine the motor current at a rate of $4KHz$. The motors include a DC power connector and an RS-232 port. Communication with the motors is achieved by transmitting ASCII text using a special-purpose instruction set (see Fig. 2). All motors are connected in a "daisy-chain" configuration, sharing a common serial cable that

connects to the RS-232 port of the computer which is used to control the manipulator.

Possible commands include position and velocity setpoints, sensing of the current position or velocity as well as specifying coefficient values for the local PID controller. Currently, the rate of communication with the motors is $9600bps$. Measured in terms of instructions, the communication rate will of course dependent on the number of characters sent to each motor. If a position setpoint and measurement request are transmitted to each of the four motors (total of approximately 60 characters) then the effective communication rate is approximately $20Hz$. Each motor is followed by 10/1 gear reduction. The distal link of each finger is connected to its gear box with a chain drive, while the proximal link is connected directly to its gear box. This results in a parallel linkage mechanism for each finger, so that the rotations of the distal and proximal links are decoupled.

The manipulator is controlled by a central digital computer which coordinates data gathering and commands the four actuators. By making use of joint angle, tactile sensing and object position information, the manipulator can locate, grasp and move objects on the desktop. The manipulator can be viewed as a hybrid system, combining continuous-time rigid body dynamics and event-driven transitions, to be controlled with hybrid, language-driven actuators. These ideas will be discussed in more detail as we proceed to describe the operation of the manipulator and report on some of the research activities facilitated by our experimental setup.

A Vision-based Deformable Tactile Sensor

Over the last two decades, tactile sensing research has focused on the development of technology and devices that attempt to endow robots with some of the dexterity that humans possess. Everyday experience as well as analysis of the kinematics of manipulation and grasping [1, 2] suggest that contact forces and locations are the most important geometric parameters for manipulation and it is precisely those parameters that most tactile sensors are designed to measure.

The limitations of rigid fingertips in the precise and algorithmic study of manipulation have been discussed in many works [3],[4],[5], some dating back more than a decade. One dis-

advantage of conventional tactile sensors is that they operate solely as force-sensing devices. That is, they measure the pressure distribution over their surface but provide little or no information on possible deformations of the surface itself. With few exceptions [6],[7], tactile arrays are typically mounted against a rigid backing and covered with a thin rubber layer to provide friction. Rigidity limits the degree to which such sensors can be used in manipulation tasks [3]. Despite that fact, much of the work in dexterous manipulation has continued to use the "point-contact" model for finger-object interactions. In fact, most of the existing tactile sensing technologies, including tactile arrays ([8],[9] and others) and vision-based tactile sensors [10],[11],[12] are not adaptable to deformable fingertips. The work in [4], [13] explored different ways of constructing non-rigid fingertips, using foam, rubber, powder and gel were investigated. The gel-filled membrane showed best overall performance in terms of attenuation of impact forces, conformability, strain dissipation, and reality factors. The compliant fingertips described next most closely resemble the gel-filled fingertip used in [13].

Figure 3 shows the deformable tactile sensor that has been developed in the Harvard Robotics Lab as a result of a fifteen year collaborative effort. A more detailed description of the sensor and its operation can be found in [14]. The sensor consists of a metal housing and a roughly elliptical latex membrane which provides an area of contact. A clear, fluid-like gel fills the membrane, sealed from the rest of the assembly by a transparent window. A grid of dots is drawn at precisely computed locations on the inner surface of the membrane. A metal fingernail serves to provide support for the membrane when the latter is being deformed by contact. The fingertip is approximately $6.2cm$ long and has a diameter of $2cm$ at its base. A schematic is shown in Fig. 4. The sensor's metal housing holds a camera with a diameter of $7.5mm$ and a fiber optic cable that illuminates the interior surface of the membrane. The camera is connected to an image acquisition board which captures images of the grid of dots on the membrane. Typical images are shown in Fig. 5. The image size that was used was 192×120 pixels. The sensor has mechanical properties that are well suited to manipulation. In particular, the use of a fluid-supported membrane [3] allows local deformations (caused by contact with an object) to be distributed throughout the enclosed volume, because of the constant pressure of the fluid inside. This is in contrast to materials that obey Hooke's law

(i.e. rubber-covered rigid fingertips) and allows the fingerpad to locally "wrap around" the object at a contact (see Fig. 6). Mechanically, the sensor acts much like a human fingertip and has been found to be very effective in providing grasp stability.

An Inverse Problem: Membrane Reconstruction

A particularly challenging problem involves the use of images of the grid of dots drawn on the inner surface of the membrane, to recover three-dimensional shape of the membrane. This can be accomplished by a "reconstruction algorithm" which we will briefly describe here. Additional details can be found in [15],[14].

Consider the grid of dots drawn on the membrane. The undeformed locations of the dots on the membrane are known a priori. When the fingertip comes in contact with the environment, the membrane deforms and the camera observes a change in the projections of the grid of dots onto the image plane (as in Fig. 5-b). A set of imaging operations (see Fig. 7) provide us with the 2-D projection of the deformations of the dots. Projective geometry tells us that there exist an infinity of solutions for the new three-dimensional coordinates of the dots. Under deformation, the portion of the membrane which is not in contact will assume a shape that minimizes its elastic energy. In addition, the volume enclosed by the membrane remains constant. These constraints, together with some genericity assumptions on the grid of dots are sufficient to obtain a solution for the three-dimensional coordinates of the grid.

A reconstruction example is shown in Fig. 8, corresponding to a human fingertip lightly touching the membrane. The reconstruction algorithm uses images such as the ones in Fig. 5 to produce a three-dimensional approximation to the membrane surface in the form of a 13×13 mesh that corresponds to a $4cm^2$ area on the fingerpad. The coordinates of the grid are measured with respect to a Euclidean frame whose origin at the center of the CCD array in the camera and whose z-axis is perpendicular to that array. "Crossed" points represent the undeformed location of the grid. The line segment through the grid is drawn through the centroid of the area of contact.

Tactile Sensor Performance

By computing the membrane displacement along the inward-pointing normal for each point on the grid, we can identify the points which are part of a contact. Figure 9-a shows typical results obtained with this method when a pencil tip is pressed lightly against the fingerpad. The graph shows a peak forming around the area of contact from which we can determine that the pencil was pressed about $1mm$ into the membrane. The area of contact included 14 grid points with their centroid at $(-4.5mm, -3.7mm, 22.2mm)$ measured in a coordinate frame located at the end of the distal link. The same method can be used to simultaneously detect multiple areas of contact (Fig. 9-b).

Typically, the reconstructed grid is used to estimate the maximum deformation and contact coordinates. The minimum inward displacement that can be detected is $0.5mm$. For small deformations of the membrane, the sensor can localize contact with a maximum error of $1.9mm$, equal to one half of the distance between neighboring dots on the membrane surface. The availability of an approximation to the fingertip surface allows one to estimate the local curvature of objects that come into contact with the sensor.

The reconstruction algorithm involves a significant amount of computation and image processing. On a dual $400MHz$ Pentium PC the maximum rate of performing this reconstruction is $15Hz$ using a 5×5 grid of dots on the membrane and a 13×13 interpolated grid to approximate the fingerpad surface. This rate is lower than what can be achieved with traditional tactile sensors, however the deformable sensor provides a much richer description of a contact.

Additional experimental results involving our tactile sensor can be found in [5]. Results on the use of the sensor in manipulation experiments are presented in [16]. Other applications being explored include the miniaturization of the sensor and use as a laparoscopic device in minimally-invasive surgery.

Visual Tracking

Together with tactile information, knowledge of the object's position is important during manipulation. An overhead camera captures images of the manipulator's workspace. Those

images are processed in order to detect and track objects in that workspace. Tracking software was developed to compute the position and orientation of the object relative to an inertial frame fixed on the desktop. To make it easier for the tracking algorithm to locate objects, a pair of black dots is mounted against a light background on the object. Alternatively, the tracking algorithm can locate any dark-colored object against the light background of the desktop, using simple image thresholding. Images obtained from the overhead camera are similar to that of Fig. 1.

In that case only position information is computed. The overhead camera provides 192×120 images which correspond to an area of $36cm \times 28cm$ on the desktop. Consequently, each pixel images a $1.9mm \times 2.3mm$ area.

Object tracking is made more efficient by using an estimate of the object's velocity – computed from past tracking data – to avoid searching the entire scene. If $p_k, v_k, a_k \in \mathbb{R}^2$ are the position, velocity and acceleration of the object, then the estimate $\hat{p}_{k+1}$ of the next (expected) object position is computed according to:

$$\begin{aligned} \hat{p}_{k+1} &= p_k + v_k \Delta t + \frac{1}{2} a_k \Delta t^2 \\ v_{k+1} &= \frac{p_{k+1} - p_k}{\Delta t} \\ a_{k+1} &= \frac{v_{k+1} - v_k}{\Delta t} \end{aligned} \tag{1}$$

and the tracking algorithm searches a small region around $\hat{p}_{k+1}$ in order to find the object and determine p_{k+1}. The maximum tracking rate is $60Hz$ on a dual $400MHz$ PC, currently limited by the maximum rate at which the camera can capture video frames. Figure 10 shows the position data obtained during a manipulation task, superimposed on a snapshot of the manipulator during the task.

Learning to Manipulate

Humans use their hands in many everyday tasks with remarkable dexterity and are able to integrate visual and tactile signals in order to manipulate various objects with precision, speed and efficiency. The development of robotic hands with similar capabilities is aimed at

producing mechanical and control systems that will be able to perform many of the tasks that are beyond the capabilities of conventional robot grippers. However, robotic fingers add significantly to the complexity of a manipulator, both in mechanical design as well as in control and coordination. Multi-fingered manipulation is difficult in part because it involves rolling contact between the object and finger surfaces. This introduces a nonholonomic constraint into the kinematics of the object-hand system. As a result, the planning of finger motions depends on the geometric evolution of the object-finger contact(s) [17],[18],[1],[2] . The contact evolution is itself dependent on the kinematic configuration of the hand, the object trajectory and most importantly the object's geometry which is often only approximately known.

Learning can be an effective approach to manipulation because it allows the handling of a large class of objects using the same algorithm, rather than having to specify model parameters for every object that one would like to manipulate. In addition, precise surface models may be difficult to obtain for all but the simplest of object geometries. A control system that learns would make it possible for the user to abstract from the particulars of system components and instead focus on its desired behavior only, while the internal details of system dynamics, constraints, etc, remain "hidden". Towards that end, consider then the following prototype problem:

Problem Statement 1 *Given an object and a feasible trajectory which is parameterized by time, find the actuator commands that result in the object following that trajectory as closely as possible.*

In this context, a "feasible" trajectory is one that does not require the fingers to travel outside their workspace or through any singular kinematic configurations.

To solve instances of Prob. 1, we choose to decompose it into an equivalent pair of problems:

Problem Statement 2 *Given an object and a desired feasible trajectory, find the joint trajectories that correspond to the object following the desired trajectory.*

Problem Statement 3 *Find the actuator commands that result in the joints tracking a set of desired trajectories "as closely as possible".*

Kinematic Exploration of an Object Trajectory

The first step towards solving an instance of Problem 1 involves "lifting" the object trajectory to the space of joint angles. We accomplish this by performing a "kinematic exploration" of the desired object trajectory, as well as by sampling the evolution of the model-dependent effects of rolling and fingertip compliance along the trajectory. The use of feedback and learning while exploring the kinematics of a particular manipulation task allows us to avoid model-based methods [2] that may be computationally expensive and difficult to implement.

Before we proceed, it is useful to review the kinematics of the finger/object system. Consider a set of k fingers, each with m degree of freedom, making contact with an object. Let $\theta, \tau \in \mathbb{R}^{m \cdot k}$ be the vector of the manipulator's joint angles and torques. Also, let $v \in \mathbb{R}^6$ be the instantaneous object velocity, and $f \in \mathbb{R}^6$ be the total wrench acting about an inertial frame fixed on the object. The kinematics of rolling contact can be summarized in the following well-known set equations [1]:

$$\begin{aligned} G^T v &= J(\theta)\dot{\theta} \\ \tau &= J^T(\theta) w \\ f &= G w \end{aligned} \tag{2}$$

where J is the manipulator Jacobian and G is the so-called grasp matrix. The quantity w denotes the vector of forces acting at the object/finger contacts. The grasp matrix is determined by the coordinates of the object/finger contacts, which are also necessary to specify J. Of course, the evolution of Eq. 2 during a manipulation task depends on the geometry of the object and fingertips. This dependence is made explicit quite elegantly in [2]. We note that the stability of a grasping configuration can be determined from the rank of G. In addition, the last of Eq. 2 can be used to select the internal forces w applied to the object, by exerting fingertip forces that are in the nullspace of G.

The following algorithm (similar to that in [19]) was used to learn the kinematics of manipulating an unknown object along a desired locus of points:

0. Sample spatially the desired object trajectory, using a finite number of "setpoints".

1. Determine a desired incremental motion for the object towards the next object trajectory setpoint.

2. Sense the position/orientation of the object, the coordinates of each fingertip/object contact, the force at each contact and all joint angles.

3. Compute the fingertip forces necessary to maintain a sufficiently tight grasp on the object. If an adjustment to the grasp is necessary, apply it and repeat Step 1.

4. Solve the hand-object kinematics of Eq. 2 for the incremental change in joint angles based on the desired change in object position/orientation.

5. Apply the change in joint angles to the fingers.

6. If the desired object setpoint has been reached, store the desired joint positions and velocities (computed from Eq. 2 for a given desired object velocity) for this setpoint. Otherwise repeat from Step 2.

7. Repeat from Step 1, until the object has attained all trajectory setpoints.

Figure 11 shows a typical trajectory obtained using the above algorithm. For this task, a 2″-diameter spherical object was used. The apparent "noise" in the actual trajectory is due to the resolution limit of the overhead camera which was used to measure the position of the object.

Software Organization

In the present setting, software development required a significant amount of time, taking approximately 50% of the person-years invested in this project. For this reason, we think it is appropriate to give some description of how the software was organized, and to discuss its possible reuse.

Over the last decade, there has emerged a engineering approach to the linguistic description for motion control tasks [20],[21],[22],[23]. The actuators used to drive the HRL manipulator are particularly suited for the study of motion description languages because they are

language-driven devices. This feature allows us to explore questions of motion planning and optimal description of a motion task in the given language. Using the actuators' instruction set together with basic sensing operations, we implemented a set of control primitives (akin to the "atoms" and "behaviors" of MDL [24]). These are the building blocks in terms of which grasping and manipulation tasks can be described. Our purpose was to implement the beginnings of a motion language for use with multi-fingered manipulators. This language should contain, at its highest levels, primitives for grasping and manipulation tasks, so that the user needs to specify little or no information about the object, other than its desired trajectory.

There is of course a substantial amount of literature on software organization, with no shortage of works emphasizing the power and conceptual advantages of thinking in terms of objects, classes and reusable modules. We refer the reader to standard references on that subject (e.g. [25]). However, when it comes to the control of electro-mechanical systems, standard object-oriented methods will take on a somewhat special form because the software must interact with one-of-a-kind hardware and instruction sets. In addition, there are few precedents that could dictate how software should be organized. We chose to organize software modules according to their interaction with the hardware and with the outside world.

Software/Hardware Interaction

The use of off-the-shelf hardware implies that the instruction sets are essentially fixed. In our case there is a given command set for the actuators and another for the image acquisition board. Those sets are to be treated as assembly code, and we designed low-level routines which interface to them. The following is a partial list of the available hardware-dependent primitives.

Sensing: The controller receives joint, tactile or object position data by transmitting a request to the appropriate sensor. Joint angles are measured by querying the corresponding motors. Polling a tactile sensor provides a three-dimensional vector for the contact location (i.e. the centroid of the area of contact), the maximum deformation depth as well as a vector for the surface normal at the point of maximum deformation.

Actuation: The controller can transmit a joint position or velocity command to a specific motor. The PID controller residing in that motor uses the command as a setpoint and regulates motor current. The controller can modify the parameters of the PID loop and can specify upper limits for the angular velocity and acceleration of the motor.

Feed-forward control: The controller transmits a pre-defined sequence of control inputs to the motors (in real time).

Hardware-independent primitives

Of course, we would like motion description languages to have as much portability as does "traditional" C++. One could argue that currently there is almost no portability for motion control programs. On the other hand, the HRL manipulator has given us a chance to see what such a software environment might be like. The following high-level primitives abstract from the specifics of the hardware and provide the basic functionality necessary for manipulation.

We note that each of the software modules described in this work must be supplemented by algorithms that are correct in detail. As the reader knows, this constitutes an important but tedious process which is beyond the scope of this paper. In the interest of space we omit many of the details. However, we do want to avoid gross oversimplification sometimes found in AI texts and wish to call attention to the fact that behind each of the following primitives there is a certain amount of mathematical analysis. Primitives are arranged according to the physical complexity of the corresponding task.

Find Object: Involves image processing, segmentation and template-based search. This primitive initiates a routine that obtains an image of the desktop from the overhead camera and searches that image for an object. Currently, an object is detected either by its contrast to the light-colored desktop or by a pair of fiducial dots mounted on its surface.

Grasp Object: The mathematics here have to do with finger kinematics and jacobians, as well as the grasp matrix (Eq. 2) and its rank [17]. After an object has been located, the controller moves the fingertips closer to the object using inverse kinematics. Usually,

no information is available on the object's geometry, therefore the controller proceeds by moving the fingertips slowly towards the known center of the object while the tactile sensors are monitored for signs of contact. When contact has occurred, tactile data is used to check for grasp stability. This is done by examining the singular values of the grasp matrix. Finally, the fingertips are moved along the surface normal at the contact(s) in order to tighten the grasp as desired. The fingertip forces necessary to apply a desired internal force to the object can be computed by solving an optimization problem whose data includes a basis for the null space of the grasp matrix [1].

Catch Object: This primitive uses the manipulator's inverse kinematics and jacobian, together with a simple difference equation (Eq. 1) which estimates the location of a moving object in the images acquired from the overhead camera. The controller tracks a moving object on the desktop (e.g. a rolling ball) and guides the fingers to follow the object at a fixed pre-specified distance on either side. When the object's velocity decreases below a specified threshold, the fingers are moved quickly towards the object until contact occurs.

Point-to-point manipulation: enables the manipulator to move an object from one equilibrium configuration to another. Once the object has been securely grasped, the controller uses joint, tactile and visual feedback together with the kinematics of the manipulator, in order to compute the incremental joint motions required to bring about a desired incremental object motion by means of Eq. 2 (see also [17],[18],[19]).

Teleoperation: The user provides a desired position for the object using a pointing device (such as a computer mouse) while the controller uses grasping and point-to-point manipulation primitives to move the object according to the user's commands.

Kinematic exploration: The analysis here makes use of the manipulator's jacobian and kinematics (Eq. 2) in a feedback control loop. Given a desired object trajectory (for an object in the manipulator's workspace), the controller uses low-level and point-to-point manipulation primitives in order to move the object to each of a sequence of configurations, all lying on the desired trajectory.

Software/Communication Interaction

Until recently, it had been common to "decouple" the communication aspects from the underlying dynamics of a control system, as this simplified the analysis and generally worked well for classical models. In the past few years however, advances in electronics, communications and network technologies have enabled the development of large-scale, complex systems. This has led to the need for re-examination of some of the fundamental assumptions involving the effects of communication on control. Smart structures, communication networks, robots and formations of autonomous vehicles, are all examples of systems which incorporate an unprecedented number of components, all of which must operate in a coordinated manner. These systems are *distributed*, in the sense that their sensors, actuators and computing elements communicate via a shared medium, be it a radio frequency, a computer bus or pins on a VLSI device. Restrictions on access to this medium often have a profound effect on the performance of the overall system, pointing out the necessity of analytical tools that bridge communication and control.

As the complexity of a system increases, simultaneous communication among components becomes an unrealistic assumption. Instead, sensors and actuators must "share the attention" of the controller. Lack of *time to communicate* is therefore an important constraint compared to lack of *computational power*. The HRL manipulator was used to explore aspects of this "coupling" between control and communication.

The manipulator uses actuators that share the computer's serial port so that the computer can communicate with only *one motor at a time*. Therefore the coordination of actuation and sensing operations requires that the controller chooses which motors and sensors to exchange information with at a particular time. The block diagram in Fig. 12 illustrates the control loops and communication constraints that govern the operation of the manipulator. The communication constraints governing the operation of the manipulator become important when one requires the manipulator to follow a desired trajectory in real-time. The controller cannot supply continuous inputs to the actuators nor can it update all actuators simultaneously. In the following we introduce a simple model that can be used to analyze this situation, which

captures characteristics of a rather general class of hybrid control systems.

A Model for Limited communication Control

Consider a continuous-time, linear, time-invariant system which is controlled by a digital computer (Fig. 13).

- The controller cannot provide continuous inputs to the LTI system; instead, commands are sent to the system every Δ units of time, via a zero-order hold.
- The dimension of the communication bus (b) that carries controller-generated inputs may be smaller than the input dimension of the LTI plant (m). As a result, the controller must choose which of the input signals to update at every cycle.

We are interested in control tasks of finite duration, therefore we consider controller-generated sequences of fixed length N. To fix notation, let $\ell^b(N)$, denote sequences of length N whose elements are in $\mathbb{R}^b$. The plant is defined by the triple $(B, A, C) \in \mathbb{R}^{m\times n} \times \mathbb{R}^{n\times n} \times \mathbb{R}^{n\times p}$, with $G(s) = C(sI - A)^{-1}B$. Finally, $L_2^p[0, T]$ denotes the space of square-integrable functions with finite support, taking values in $\mathbb{R}^p$.

The controller can select which input is to be updated at a particular time. This leads to the idea of a "communication sequence" [26], which can be understood as the order of operations for the switch(es) connecting the controller to the various parts of the system. We will use the notation

$$\sigma = \{(\sigma(0), \sigma(1), ..., \sigma(N-1)) : \sigma(i) \in \{0, 1\}^m\}$$

to represent a communication sequence. The vectors $\sigma(i)$ of a communication sequence σ are to be interpreted as indicators of which of the elements of the system input $v(t)$ are to be updated by the controller at $t = i\Delta$, $i = 0, \ldots, N-1$. Because only b inputs can be communicated at one time, we must impose a feasibility condition, namely that $\sum_j \sigma_j(i) \leq b$ for $i = 0, ...N-1$. It is worth noting that the definition of a communication sequence allows us to quantify the amount of "attention" that the controller pays to each input and output of a computer-controlled system.

This model for computer-controlled systems has been used to understand optimal control problems involving output tracking [27] and stabilization [28] in the presence of communication constraints. Here we discuss some of the ideas involving the input-output characteristics of computer-controlled systems.

Input-Output behavior

We ignore for the moment any constraints having to do with the size of the communication bus (i.e. all m inputs can be updated at once) and consider the resulting sampled-data system as a linear, time-varying map that takes sequences to continuous-time outputs: $\Lambda_{(G,\Delta,N)}(t) : \ell^m(N) \longrightarrow L_2^p[0, N\Delta]$ This map is completely determined by the parameters of the underlying LTI plant, the controller's sampling period and the duration of the task:

$$y(t) = \Lambda_{(G(s),\Delta,N)}(t)u = \sum_{k=0}^{N-1} \phi_\Delta(t - k\Delta)u(k)$$

where

$$\phi_\Delta(t) = \begin{cases} \int_0^{min(t,\Delta)} Ce^{A(t-\tau)}Bd\tau & t \geq 0 \\ 0 & t < 0 \end{cases}$$

Now, fix a communication sequence σ according to which inputs are to be transmitted. The multiplexing action of σ can be expressed as a linear operator $M(\sigma)$

$$M : \ell^b(N) \longrightarrow \ell^m(N)$$

The operator M is determined by the elements of σ. In fact, if we identify elements of $\ell^b(N)$ with vectors in $\mathbb{R}^{N \cdot b}$ then M can be written as an $Nm \times Nb$ matrix with binary entries. The key to constructing M is to notice that consecutive elements $(\sigma'(k), \sigma'(k+1)$ for some $k)$ of a sequence $\sigma' \in Range(M)$, differ in at most b places (as dictated by $\sigma(k)$). The closed-form expression for M is rather cumbersome and will not be given here (see [27]). To summarize, for a fixed communication sequence, the overall input-output map is given by:

$$y = \Lambda M u$$

where we have suppressed the dependence of Λ and M on the system parameters.

An Invertibility Experiment

We now proceed to apply our analysis of computer-controlled systems to a class of problems involving real-time trajectory tracking.

Suppose that we would like to manipulate an object along the (planar) real-time trajectory:

$$\begin{aligned} x_d(t) &= 4.2\cos(t) \\ y_d(t) &= 4.5 + 1.7\sin(2t), \qquad t \in [0,1] sec \end{aligned} \tag{3}$$

For the sake of simplicity we choose to restrict object and finger motions to the horizontal plane. Of course each finger is capable of motion in the vertical direction, so that one could consider object trajectories that exercise all three degrees of freedom in each finger. Using our algorithms for kinematic exploration, we were able to compute the joint trajectories that would result in the object following the desired path (see Fig 14). However, these "nominal" joint trajectories cannot be communicated to the actuators because of the existence of communication constraints between actuators and controller. Instead, any joint input must be piecewise constant (i.e. an input is set and cannot be changed until some time later). How are we then to compute the "best" set of such inputs?

If we approximate the manipulator with a LTI system then the question is one of "inverting" that system to find the input that will produce a desired output [29]. We must make a distinction here between the Moore-Penrose inverse, which is what we look for in this section, compared to the approximate inverse corresponding to an operator whose range space is dense. Turning to our model for computer-controlled systems, it is obvious that such systems are not invertible because of the existence of communication constraints. Therefore, the best one can hope for is to approximate a desired output "as closely as possible". If we choose the norms induced by the usual inner products in $\ell^m(N)$ and $L_2^p[0,T]$:

$$\langle u, v \rangle_{\ell^m(N)} = \sum_{k=0}^{N-1} u^T(k)v(k), \qquad \langle y, z \rangle_{L_2^p[0,T]} = \int_0^T y^T(t)z(t)dt$$

then we trajectory-following can be posed as a least-squares matching problem:

Problem Statement 4 *Given $y_d \in L_2^p[0,T]$ find $u \in \ell^b(N)$ that minimizes*

$$||y_d - \Lambda M u||_L^2 \tag{4}$$

It can be shown [27] that if the original LTI system has normal rank (i.e. $limsup_{|s|=1} rank(C(sI - A)^{-1}B) = m$ and if the controller communicates with every input at least once over the course of the task, then the operator ΛM is $1-1$ and thus has a generalized (Moore-Penrose) inverse given by: We conclude that ΛM is 1-to-1 and has a generalized-inverse given by

$$(\Lambda M)^{\#} = (M^T \Lambda^* \Lambda M)^{-1} M^T \Lambda^* \tag{5}$$

where Λ^* is the adjoint operator of Λ, given by

$$\begin{aligned} \Lambda^*_{(G,\Delta,N)} &: L_2^p[0, N\Delta] \longrightarrow \ell^m(N) \\ (\Lambda^* y)(j) &= \int_0^{N\Delta} \phi_\Delta^T(t - j\Delta) y(t) dt \quad j = 0, \ldots, N-1 \end{aligned}$$

for $y \in L_2^p[0, N\Delta]$.

Finally, the solution to our trajectory-following problem is given by

$$u_* = (M^T \Lambda^* \Lambda M)^{-1} M^T \Lambda^* (y_d - y_{ic})$$

where y_{ic} is the effect of the initial conditions of $G(s)$.

Equation 5 can be considered an analogue of the well-known formula for the left pseudo-inverse of an operator $K : \mathbb{R}^n \to \mathbb{R}^m$, with $m > n$, $rank(K) = n$:

$$K^{\#} = (K^T K)^{-1} K^T$$

Trajectory-Following Experiments

We modeled the manipulator as a four-input LTI computer-controlled system with a communication bus of dimension 1. Using kinematic exploration, we obtained the nominal joint trajectories required for the object to follow the path given by Eq. 3 (shown in Fig. 14). A communication sequence was selected and the generalized inverse of Eq. 5 was used to compute the optimal input sequence u_* (a total of 20 samples to be sent to the motor per second). That sequence was transmitted to the actuators and the actual object trajectory was recorded. Finally, the L_2 tracking error, $y_d - \Lambda M u_*$, was evaluated.

Uniform attention

We selected a communication sequence corresponding to "uniform attention":

$$\sigma = (e_1, e_2, e_3, e_4, e_1, \ldots, e_4)$$

where e_i denotes the standard basis vector in $\mathbb{R}^4$. In this setting the finger joins are labeled as follows: 1 - left proximal, 2 - left distal, 3 - right proximal, 4 - right distal. Fig. 15 shows the input signals applied to the actuators and the resulting object trajectory. There is good agreement between the desired and actual curves. The L_2 tracking error (Eq. 4) was 5.5.

Averaging

To obtain a basis for comparison, we computed a sequence of control inputs by "averaging", the nominal (but infeasible) actuator inputs of Fig. 14. For example, if a control sample were to be sent at $t = t_k$ and then at $t = t_{k+1}$, then set $u_k = \frac{1}{t_{k+1}-t_k} \int_{t_k}^{t_{k+1}} u(t)dt$. Figure 16 shows the input/output pair corresponding to that approach. Tracking performance was quite poor, with an L_2 error of 12.1

Non-uniform attention

The figure-8-tracking experiment was performed again, this time using a communication sequence that devotes $10\%, 35\%, 15\%$ and 40% to inputs $1, 2, 3$ and 4 respectively:

$$\sigma = (e_3, e_4, e_1, e_2, e_4, e_4, e_2, e_4, e_1, e_2, e_4, e_2, e_3, e_4, e_2, e_4, e_2, e_3, e_4, e_2)$$

using again the standard basis vectors in $\mathbb{R}^4$ to specify which input gets updated at a particular time. (inputs 2 and 4) are updated more frequently than proximal joints. We arrived at this choice of communication sequence by observing the desired joint trajectories (in Fig. 14). For each time interval of length $\Delta = 0.05sec$, we allocated communication cycles using as a guide the amount of rotation required by each joint over was slightly improved compared to what was achieved attention. The L_2 error was 3.2.

Conclusions and New Vistas

We have described an experimental facility designed to study aspects of multi-modal intelligent control and reported on some of the related research activities. The purpose of the manipulator is to provide a reliable, versatile control system that will be useful both as an instructional aid and as a research tool. As the field of intelligent control gains maturity, new issues are emerging, related to our understanding of larger, multi-component systems and to the interactions among control, communication, complexity, networks and signal processing. Our goal in developing a hybrid system of this type was to create a "system of systems" which captures such interactions and in which several sensing and actuation modalities have to be brought together into the control decisions. The manipulator's robustness is reflected in the large number of graduate and undergraduate students which have used the system over the past fifteen years. Although the manipulator has undergone several refinements over time, the use of off-the-shelf components whenever possible has contributed to its long life and upgradeability. Some of the research efforts for which the manipulator has proved useful involved mechanical design, computer vision, real-time programming, limited communication control, robotic manipulation and tactile sensing.

Current plans for improving the mechanical characteristics of the manipulator include replacing the transmission chain and gears with a cable drive in order to decrease joint compliance and backlash. Estimating object shape from overhead images could prove valuable in planning effective grasps. In addition, the rich set of data provided by the manipulator's fingertips could be used to sense slip or in combination with overhead images for shape estimation. More broadly, it is important to understand how sensors and actuators might "best" share the attention of the (centralized) controller. The manipulator allows us to explore problems of this type, which are of interest to a broad spectrum of researchers. In fact, issues related to attention, including the interaction of control and communication, are of particular significance to modern engineering systems where the presence of a networks or distributed topology is becoming increasingly common. We believe that the experimental system presented in this work together with others like it can contribute to the ongoing investigation of these questions as well as to the education of future engineers. To date, the HRL manipulator

has helped shape five doctoral theses and fifteen undergraduate research projects at Harvard. It has influenced our teaching work on a wide variety of topics as suggested by the references. We expect that the manipulator will continue to evolve and we hope that other researchers will find it to be a useful paradigm for intelligent control.

References

[1] J. Kerr and B. Roth, "Analysis of multifingered hands," *International Journal of Robotics Research*, vol. 4, no. 4, pp. 3–17, Winter 1986.

[2] D. J. Montana, "The kinematics of contact and grasp," *International Journal of Robotics Research*, vol. 7, no. 3, pp. 17–32, 1988.

[3] R. W. Brockett, "Robotic hand with rheological surfaces," in *Proc. of the 1985 IEEE International Conference on Robotics and Automation*, 1985, pp. 942–946.

[4] K.B. Shimoga and A.A. Goldenberg, "Soft robotic fingertips part I: A comparison of construction materials," *Int. Journal of Robotics Research*, vol. 15, no. 4, pp. 320–334, 1996.

[5] D. Hristu, N. J. Ferrier, and R. W. Brockett, "The performance of a deformable-membrane tactile sensor: basic results on geometrically-defined tasks," in *Proc. of the 2000 IEEE Int'l Conference on Robotics and Automation*, April 2000.

[6] E. J. Nicolson and R.S. Fearing, "Sensing capabilities of linear elastic cylindrical fingers," in *Proc. of the RSJ/IEEE International Conference on Intelligent Robots and Systems*, 1993, vol. 1, pp. 178–85.

[7] R.A. Russell, "Compliant-skin tactile sensor," in *Proc. of the 1987 IEEE International Conference for Robotics and Automation*, June 1987, pp. 2221–233.

[8] W.D. Hillis, "Active touch sensing," *International Journal of Robotics Research*, vol. 1, no. 2, pp. 33–44, 1982.

[9] R. S. Fearing, "Tactile sensing mechanisms," *International Journal of Robotics Research*, vol. 9, no. 3, pp. 3–23, 1990.

[10] J. Rebman and K.A. Morris, "A tactile sensor with electrooptic transduction," in *Robot Tactile Sensors*, A. Pugh, Ed., vol. 2, pp. 145–155. IFS Publications, Springer-Verlag, Berlin, 1986.

[11] S. Begej, "Planar and finger-shaped optical tactile sensors for robotic manipulation," *IEEE Transactions on Robotics and Automation*, vol. 4, no. 5, pp. 472–484, 1988.

[12] H. Maekawa, K. Tanie, K. Komoriya, M. Kaneko, C. Horiguchi, and T. Sugawara, "Development of a finger-shaped tactile sensor and its evaluation by active touch," in *Proc. of the 1992 IEEE International Conference for Robotics and Automation*, June 1992, pp. 2221–233.

[13] K.B. Shimoga and A.A. Goldenberg, "Soft robotic fingertips part II: Modeling and impedance regulation," *Int. Journal of Robotics Research*, vol. 15, no. 4, pp. 335–350, 1996.

[14] N. Ferrier, K. Morgansen, and D. Hristu, "Implementation of membrane shape reconstruction," Tech. Rep. 97-1, Harvard Robotics Lab, Harvard University, 1997.

[15] N. J. Ferrier and R.W. Brockett, "Reconstructing the shape of a deformable membrane from image data," *Int. Journal of Robotics Research*, (to appear).

[16] D. Hristu, *Optimal Control with Limited Communication*, Ph.D. thesis, Harvard University, Div. of Engineering and Applied Sciences, 1999.

[17] J. Kerr, *An Analysis of Multifingered Hands*, Ph.D. thesis, Stanford University Dept. of Mechanical Engineering, 1984.

[18] D. J. Montana, *Tactile Sensing and the Kinematics of Contact*, Ph.D. thesis, Harvard University, Division of Applied Sciences, 1986.

[19] H. Maekawa, K. Tanie, and K. Komoriya, "Tactile sensor based manipulation of an unknown object by a multifingered hand with rolling contact," in *IEEE International Conference for Robotics and Automation*, June 1995, pp. 743–750.

[20] R. W. Brockett, "On the computer control of movement," in *Proceedings of the 1988 IEEE Conference on Robotics and Automation*, April 1988, pp. 534–540.

[21] R. W. Brockett, "Formal languages for motion description and map making," in *Robotics*, pp. 181–93. American Mathematical Society, 1990.

[22] R. M. Murray, D. C. Deno, K. S. J. Pister, and S. S. Sastry, "Control primitives for robot systems," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 1, pp. 183–193, Jan. 1992.

[23] V. Manikonda, P. S. Krishnaprasad, and J. Hendler, "Languages, behaviors, hybrid architectures and motion control," in *Mathematical Control Theory*, J.C. Willems J. Baillieul, Ed., pp. 199–226. Springer, 1998.

[24] R.W. Brockett, "Language driven hybrid systems," in *Proc. of the 33rd IEEE Conference on Decision and Control*, 1994, pp. 4210–4214.

[25] E. Gamma, R. Helm, R. Johnson, J. Vlissides, and G. Booch, *Design patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1995.

[26] R. W. Brockett, "Stabilization of motor networks," in *Proc. of the 34th IEEE Conference on Decision and Control*, Dec. 1995, pp. 1484–8.

[27] D. Hristu, "Generalized inverses for finite-horizon tracking," in *Proc. of the 38 IEEE Conference on Decision and Control*, Dec. 1999, vol. 2, pp. 1397–402.

[28] D. Hristu and K. Morgansen, "Limited communication control," *Systems and Control Letters*, vol. 37, no. 4, pp. 193–205, July 1999.

[29] R. W. Brockett, *The invertibility of dynamic systems with application to control*, Ph.D. thesis, Case Western Reserve University, 1964.

List of Figures

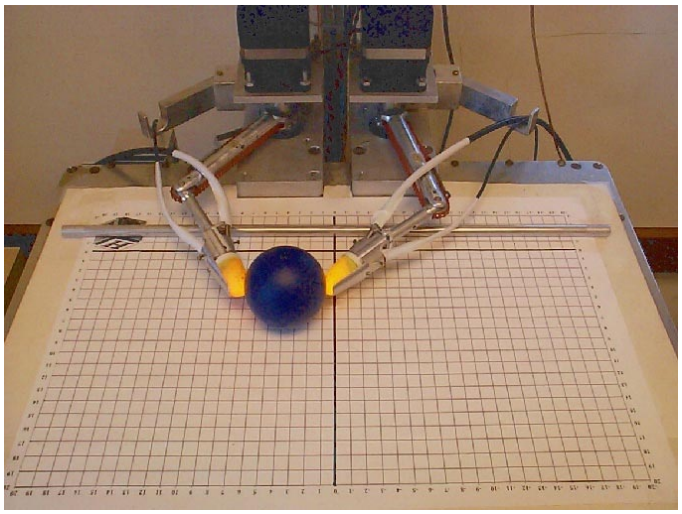


Figure 1: The HRL planar manipulator.

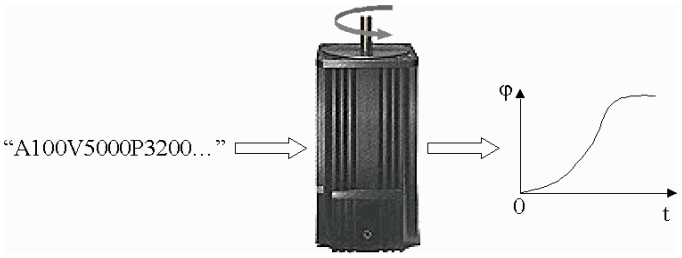


Figure 2: Language-driven motors.

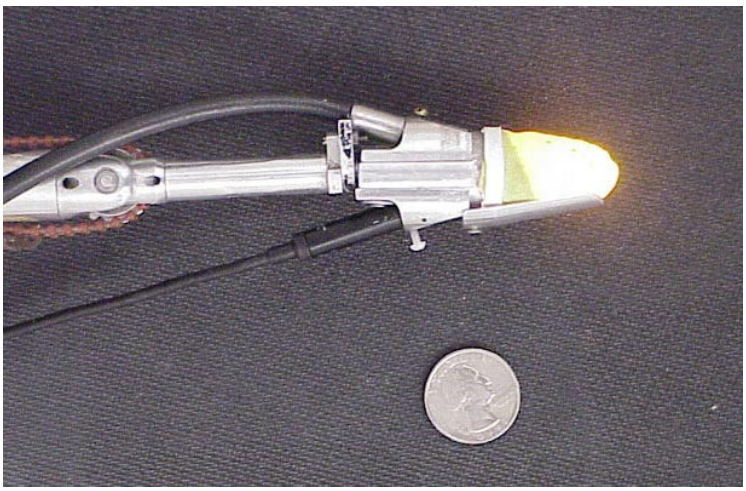


Figure 3: The tactile sensor.

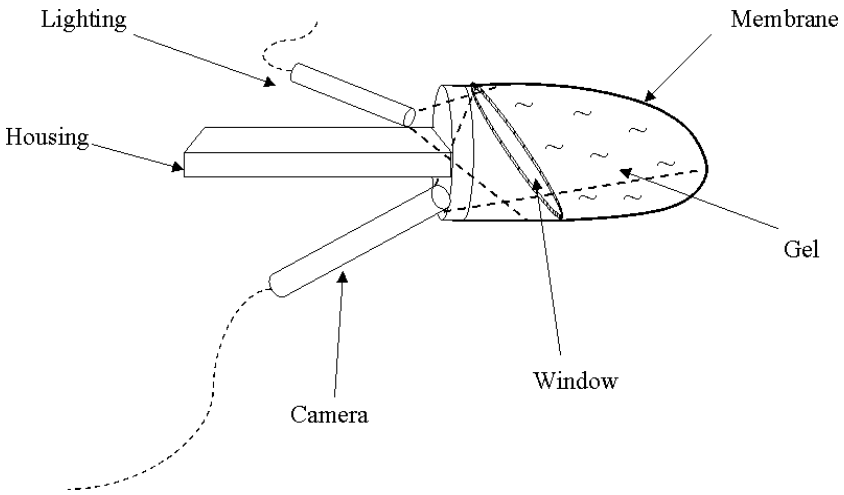


Figure 4: Tactile sensor schematic.

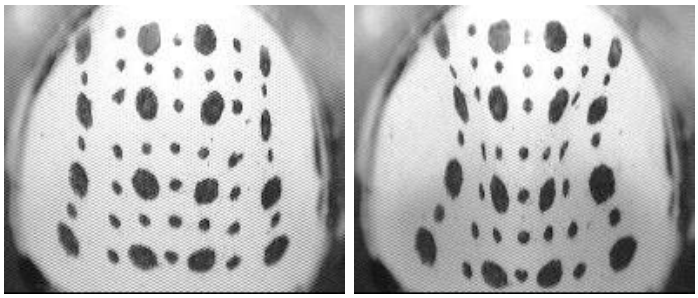
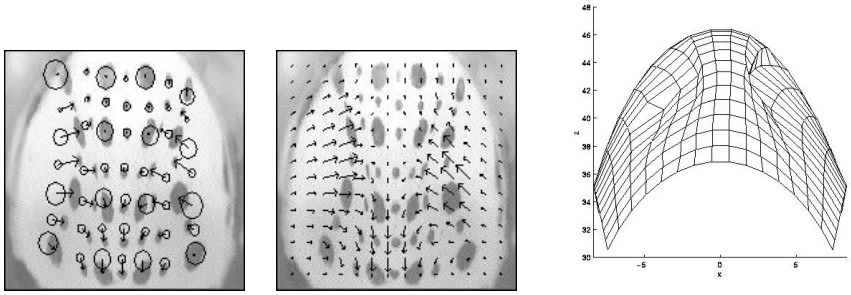


Figure 5: Camera view of membrane: (a) undeformed (b) in contact with an object.

Figure 6: Grasping with a deformable fingertip



(a) (b) (c)

Figure 7: Fingertip operation: (a) Image data of the displacement of the pattern of dots is used to interpolate a flow field, (b). The image flow field, along with other constraints enable reconstruction of the 3D shape of the deformed membrane (c).

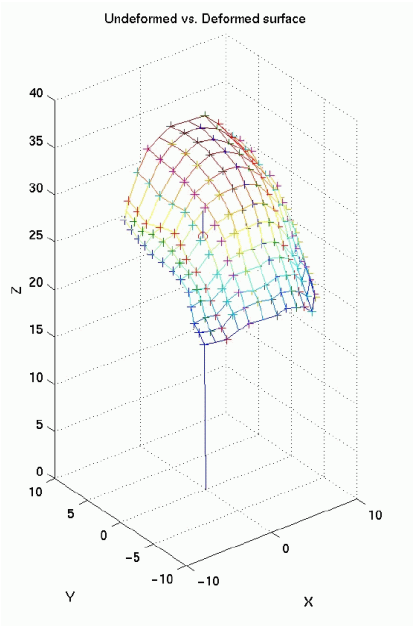


Figure 8: Tactile sensing example.

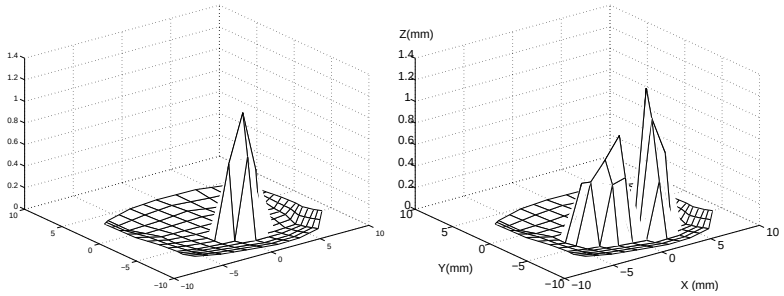


Figure 9: Contact detection.

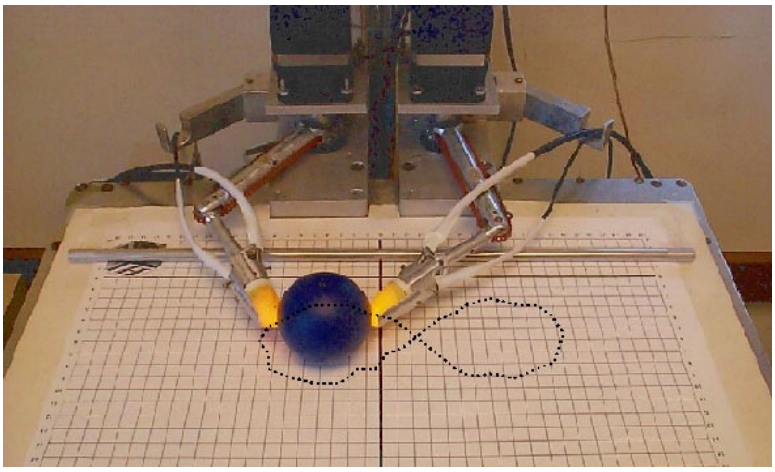


Figure 10: Tracking an object on the desktop.

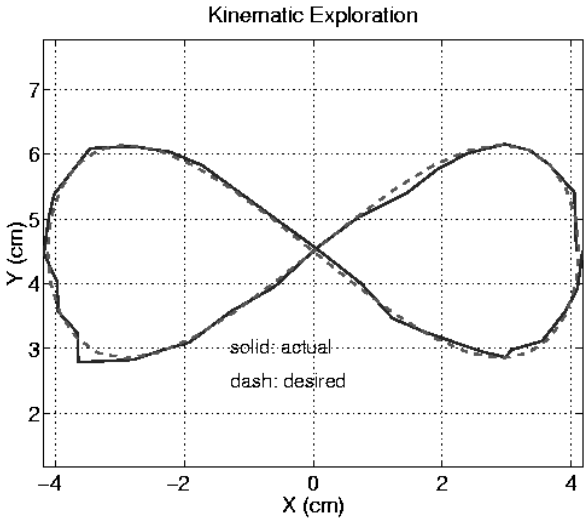


Figure 11: Kinematic exploration of a "figure-8" path.

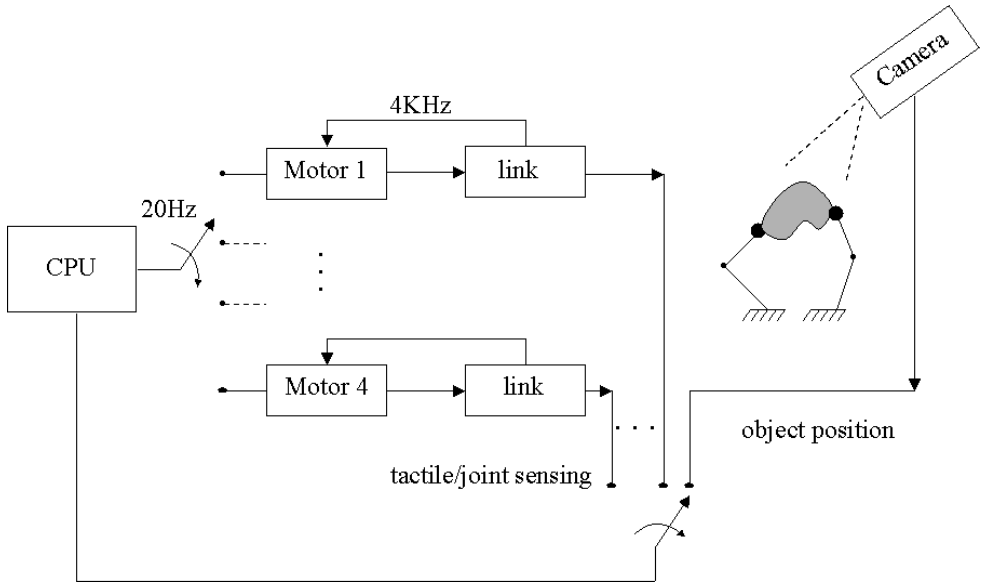


Figure 12: Low-level control architecture.

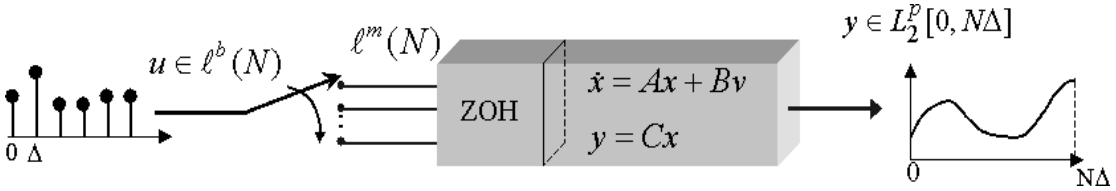


Figure 13: A computer-controlled system.

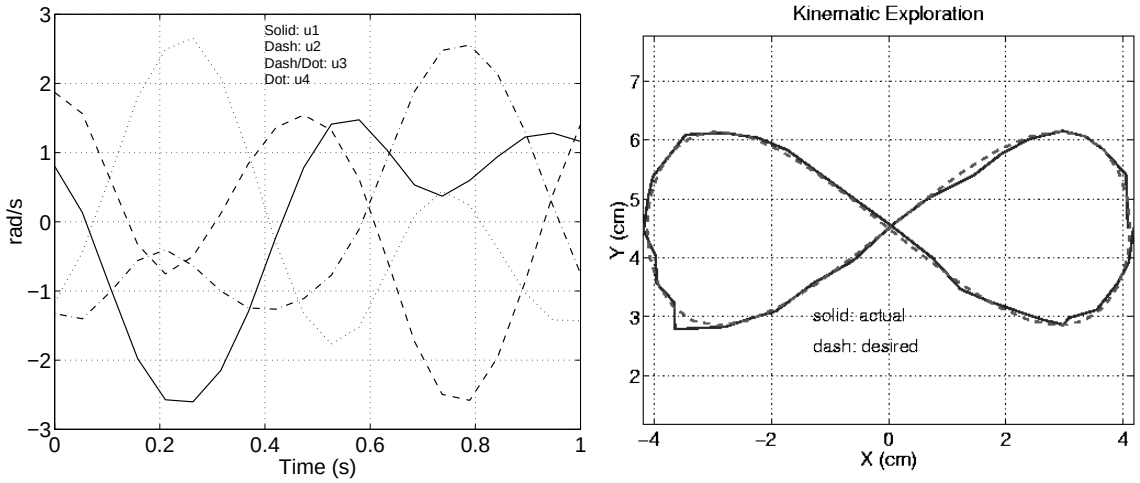


Figure 14: Nominal joint and object trajectories - figure-8.

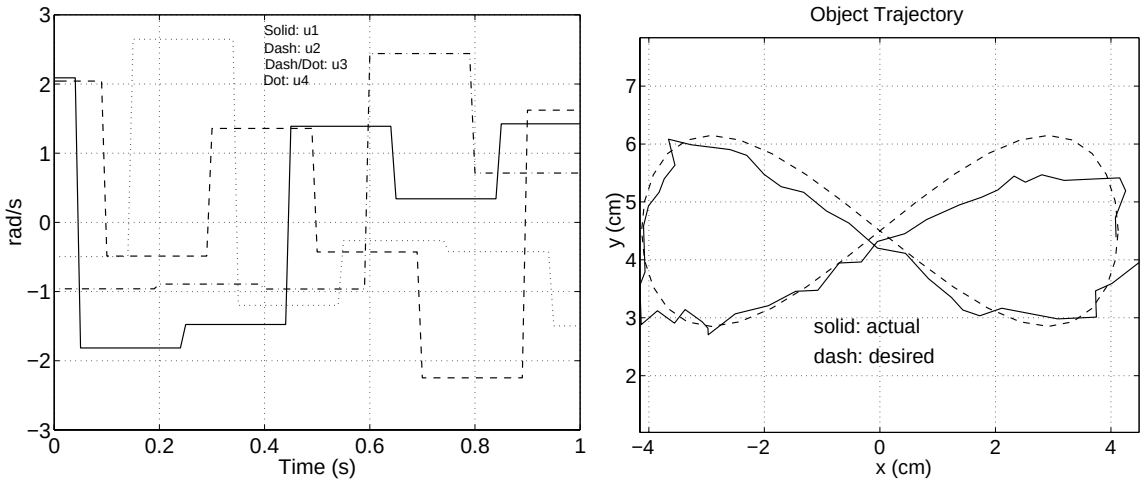


Figure 15: Optimal inputs (uniform attention) and resulting object trajectory.

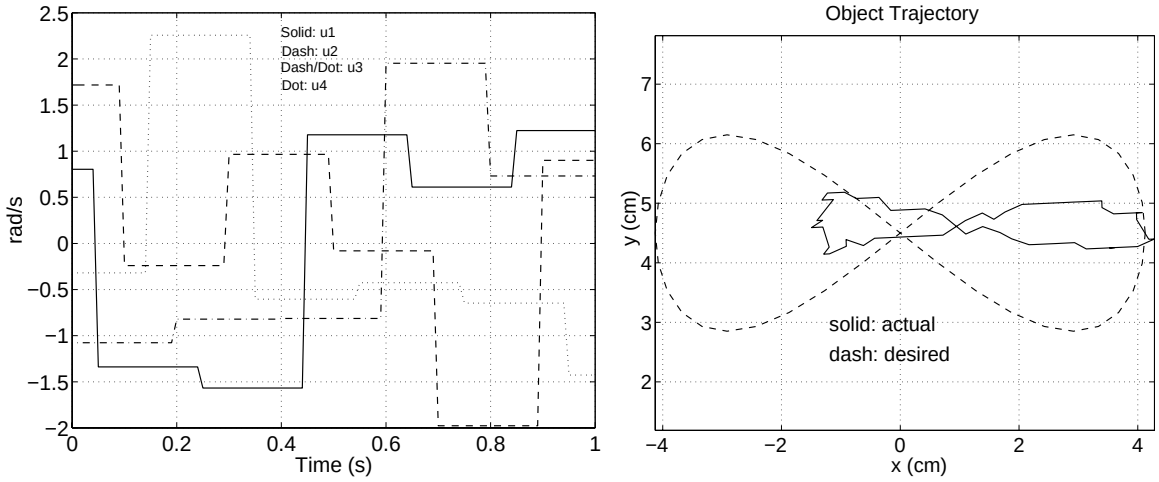


Figure 16: "Average" inputs (uniform attention) and resulting object trajectory.

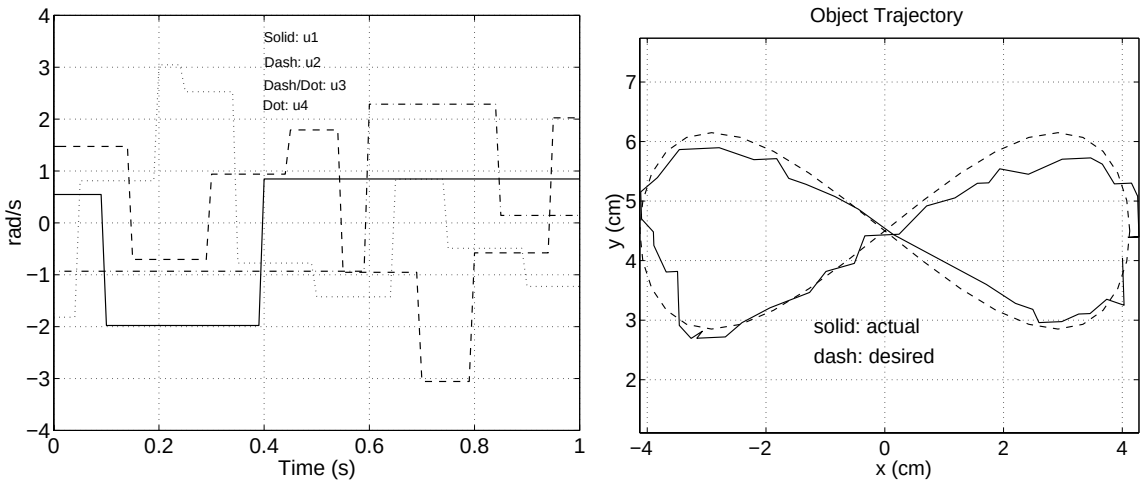


Figure 17: Optimal inputs (non-uniform attention).

Author Information

Roger W. Brockett	Dimitrios Hristu[1]
Maxwell Dworkin Lab	3119 A.V. Williams Bldg
Harvard University	University of Maryland
Cambridge, MA 02138	College Park, MD 20742
617 495-3922	301 405-7485
brockett@hrl.harvard.edu	hristu@isr.umd.edu

1. This work was done while the author was with the Division of Engineering and Applied Sciences at Harvard University, Cambridge, MA.

This work was funded by the following grants: Army DAAG 55 97 0114, NSF EEC 94 02384 and Army DAAL 03-92- G 0115.