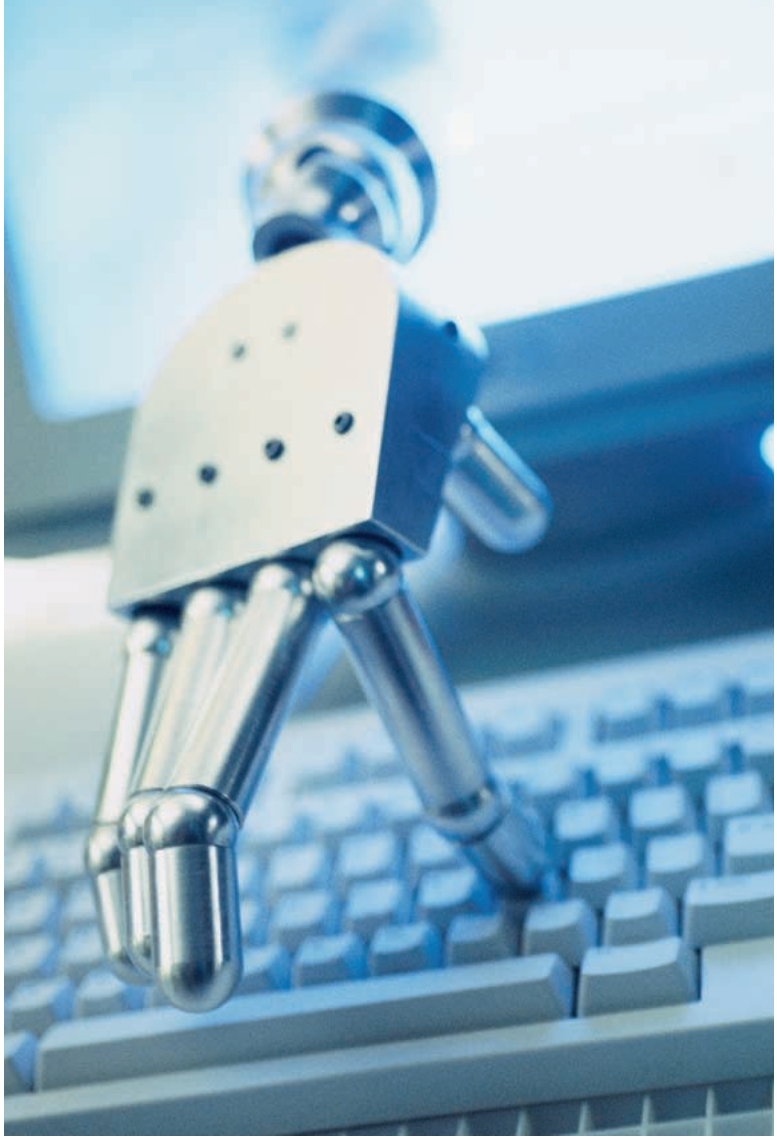
FEATURE



©EYEWIRE.COM

Experimenting with Hybrid Control

By Dimitrios Hristu-Varsakelis and Roger W. Brockett

Recently developed experimental and numerical environments have helped breathe life into the various control theories found in textbooks and have thereby greatly changed the educational experience of 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 communica-

Hristu-Varsakelis (hristu@eng.umd.edu) is with the University of Maryland, 176 Glenn L. Martin Hall, College Park, MD 20742, U.S.A. Brockett is with Maxwell Dworkin Lab, Harvard University, Cambridge, MA 02138, U.S.A.

0272-1708/02/$17.00©2002IEEE

tion requirements, signal processing, data logging, and so on, 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 several challenges beyond those discussed above. The systems under consideration must be very flexible. They should also reflect the complexity of purpose and the possibility of multimodal operation that one expects to find in complex systems. Of course, the hardware must continue to be reliable and relatively easy to understand at an intuitive level.

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 in this article is to describe the structure of the system and to present a sample of the experiments that were facilitated by it.

The manipulator can interact with objects of different complexity and is subject to communication constraints arising in a completely natural and generic way.

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.

Electromechanical Hardware

The manipulator pictured in Fig. 1 is a two-fingered hand that was developed in the Harvard Robotics Lab (HRL). The manipulator consists of a pair of fingers, made as "mirror images" of one another, mounted 5.4 cm 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 12.9 and 5.6 cm, 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 11.8 cm. The finger workspace is approximately 10 cm tall with an elliptical base whose major and minor axes are 15 cm and 12 cm long. That workspace is imaged by a camera 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 25 oz-in (100 oz-in peak) with a motor constant of 10.8 oz-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 onboard microprocessor implements a proportional-integral-derivative (PID) controller that can servo on the shaft position or velocity with programmable feedback and feedforward gains. This internal controller accepts user-supplied commands that are used to determine the motor current at a rate of 4 kHz. 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 set points, 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 9600 b/s. Measured in terms of instructions, the communication rate will of course depend on the number of characters sent to

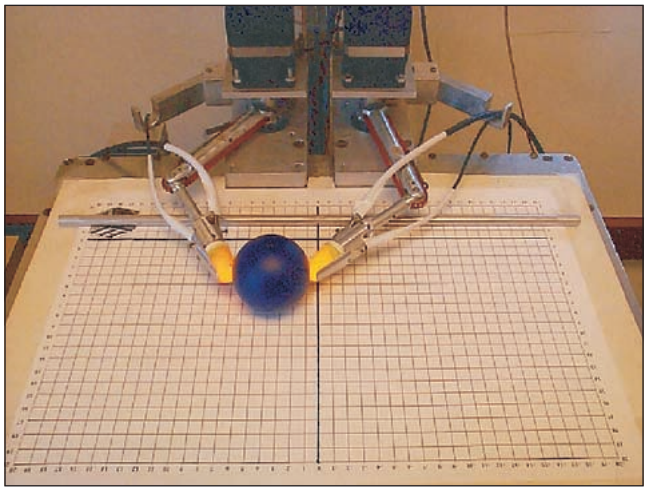


Figure 1. *The HRL planar manipulator.*

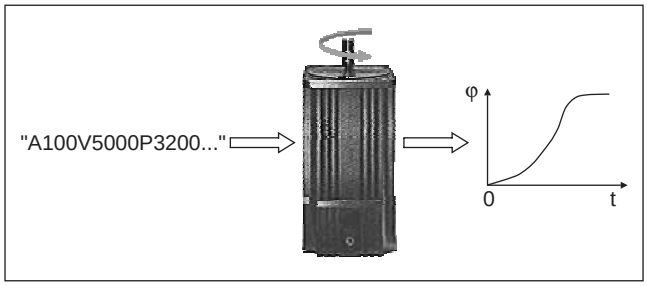


Figure 2. *Language-driven motors.*

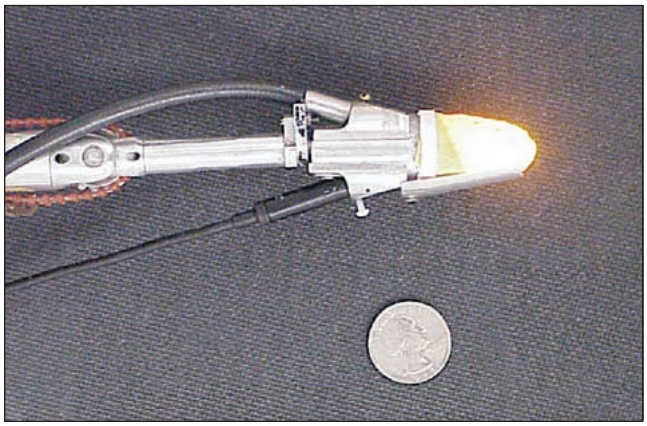


Figure 3. *The tactile sensor.*

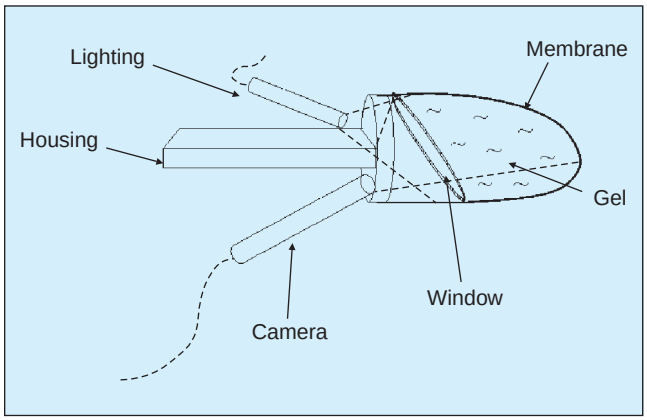


Figure 4. *Tactile sensor schematic.*

each motor. If a position set point and measurement request are transmitted to each of the four motors (a total of approximately 60 characters), then the effective communication rate is approximately 20 Hz. Each motor is followed by 10/1 gear reduction. The distal link of each finger is connected to the motor's gearbox with a chain drive, whereas the proximal link is connected directly to the gearbox. 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 using 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]-[5], some dating back more than a decade. One disadvantage 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]-[12], are not adaptable to deformable fingertips. The work in [4] and [13] explored different ways of constructing nonrigid fingertips; foam, rubber, powder, and gel were investigated. The gel-filled membrane showed the 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].

Fig. 3 shows the deformable tactile sensor that has been developed in the Harvard Robotics Lab as a result of a 15-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 that provides an area of contact. A clear, fluidlike gel fills the membrane, sealed off 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 6.2 cm long and has a diameter of 2 cm at its base. A schematic is shown in Fig. 4. The sensor's metal housing holds a camera with a diameter of 7.5 mm 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 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 "wrap around" the object locally 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 the 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] and [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) provides us with the two-dimensional projection of the deformations of the dots. Projective geometry tells us that there exists an infinity of solutions for the new three-dimensional coordinates of the dots. Under deformation, the portion of the membrane that 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 4-cm^2 area on the fingerpad. The coordinates of the grid are measured with respect to a Euclidean frame whose origin is at the center of the charge-coupled device (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.

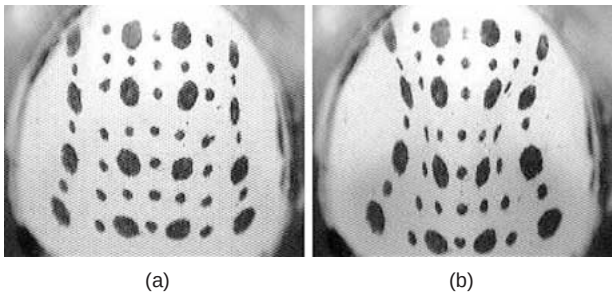


Figure 5. *Camera view of membrane: (a) undeformed; (b) in contact with an object.*

Figure 6. *Grasping with a deformable fingertip.*

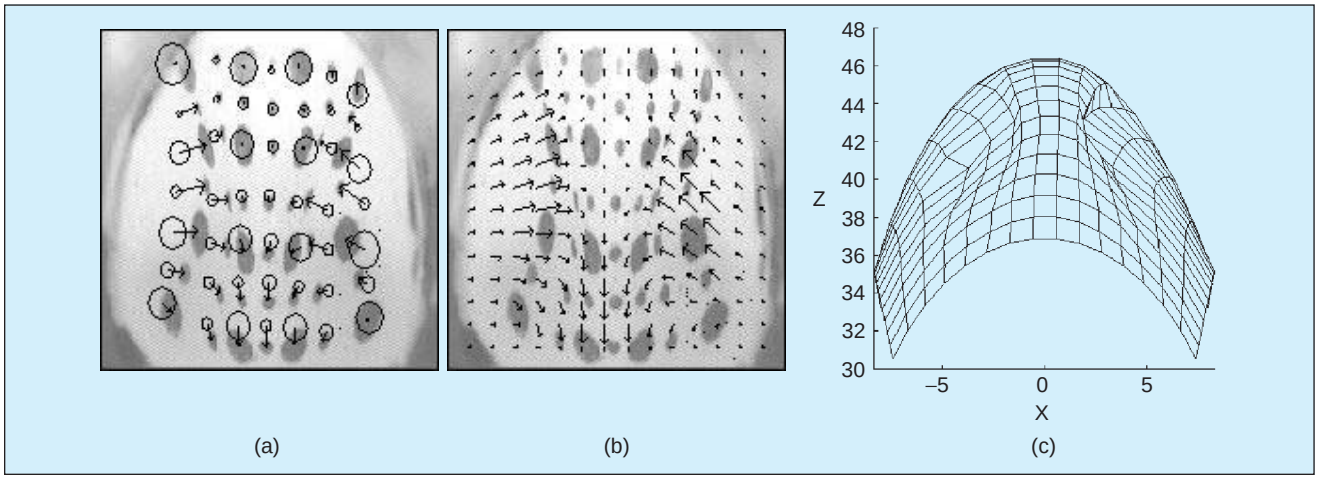


Figure 7. *Fingertip operation: (a) image data of the displacement of the pattern of dots is used to interpolate a flow field, (b); the image flow field, along with other constraints, enables reconstruction of the 3-D shape of the deformed membrane (c).*

Tactile Sensor Performance

By computing the membrane displacement along the inward-pointing normal for each point on the grid, we can identify the points that are part of a contact. Fig. 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 1 mm into the membrane. The area of contact included 14 grid points with their centroid at (–4.5 mm, –3.7 mm, 22.2 mm) 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)).

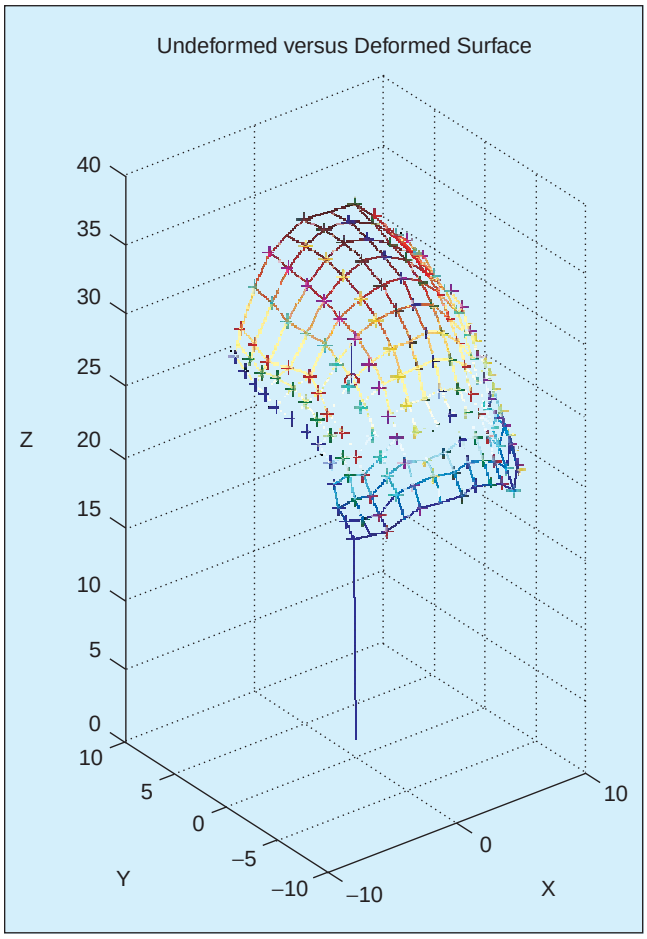


Figure 8. *Tactile sensing example.*

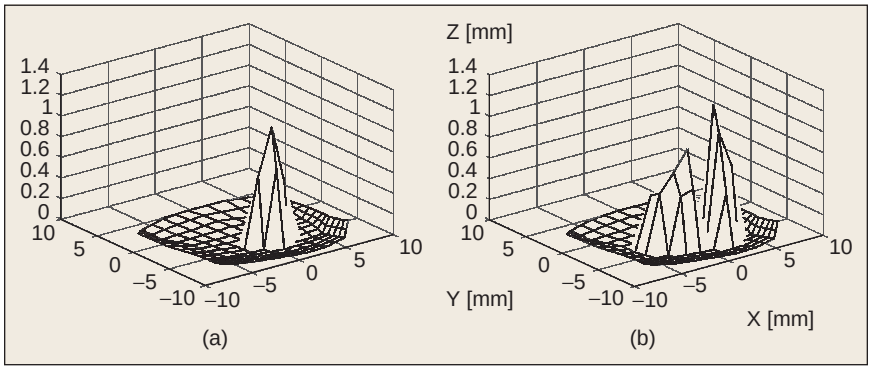


Figure 9. *Detection of (a) single and (b) multiple contacts.*

Typically, the reconstructed grid is used to estimate the maximum deformation and contact coordinates. The minimum inward displacement that can be detected is 0.5 mm. For small deformations of the membrane, the sensor can localize contact with a maximum error of 1.9 mm, 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 400-MHz Pentium PC, the maximum rate of performing this reconstruction is 15 Hz 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 its 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. These images are processed 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, using simple image thresholding, the tracking algorithm can locate any dark-colored object against the light background of the desktop. Images obtained from the overhead camera are similar to that of Fig. 1. In this case, only position information is computed. The overhead camera provides 192×120 images that correspond to an area of 36 cm $\times$ 28 cm on the desktop. Consequently, each pixel images a 1.9-mm $\times$ 2.3-mm 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

$$\hat{p}_{k+1} = p_k + v_k \Delta t + \frac{1}{2} a_k \Delta t^2$$
$$v_{k+1} = \frac{p_{k+1} - p_k}{\Delta t}$$
$$a_{k+1} = \frac{v_{k+1} - v_k}{\Delta t} \quad (1)$$

and the tracking algorithm searches a small region around $\hat{p}_{k+1}$ to find the object and determine p_{k+1}. The maximum tracking rate is 60 Hz on a dual 400-MHz PC, currently limited by the maximum rate at which the camera can capture video frames. Fig. 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 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 and in control and coordination. Multifingered 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) [1], [2], [17], [18]. The contact evolution is itself dependent on the kinematic configuration of the hand, the object trajectory, and most important, 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, and so on, remain "hidden." Toward that end, consider the following prototype problem.

- *Problem Statement 1:* Given an object and a feasible trajectory that is parametrized 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 Problem 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."

The actuators used to drive the manipulator 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.

Kinematic Exploration of an Object Trajectory

The first step toward 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 degrees 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

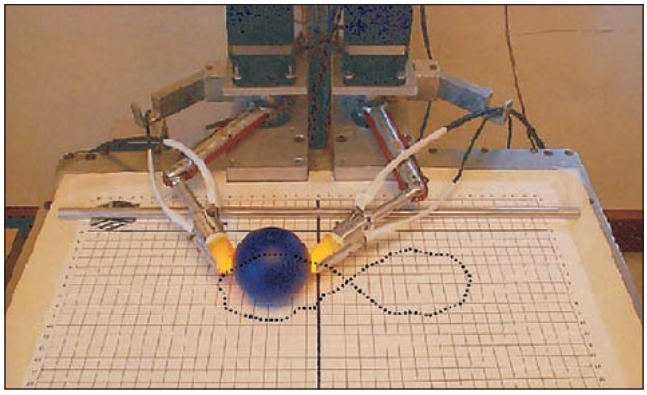


Figure 10. *Tracking an object on the desktop.*

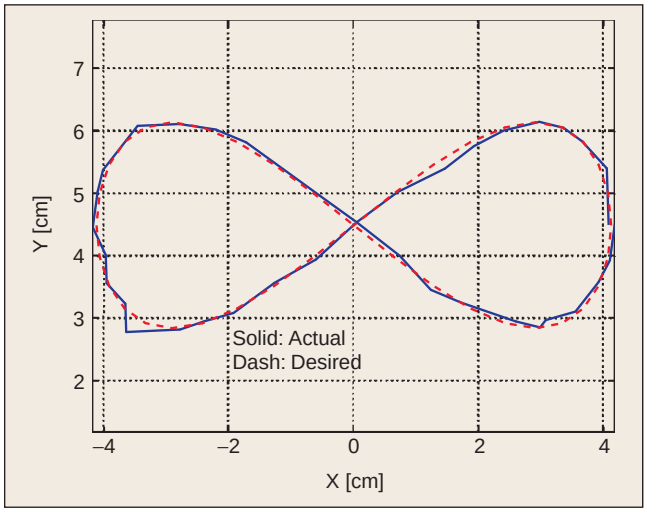


Figure 11. *Kinematic exploration of a "figure-8" path.*

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 of equations [1]:

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

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 (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 equality in (2) can be used to select the internal forces 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 set points.
1) Determine a desired incremental motion for the object toward the next object trajectory set point.
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 (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 set point has been reached, store the desired joint positions and velocities (computed from (2) for a given desired object velocity) for this set point. Otherwise repeat from step 2.
7) Repeat from step 1 until the object has attained all trajectory set points.

Fig. 11 shows a typical trajectory obtained using the above algorithm. For this task, a 2-in-diameter spherical object was used. The apparent "noise" in the actual trajectory is due to the resolution limit of the overhead camera that 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 describe how the software was organized and to discuss its possible reuse.

Over the last decade, there has emerged an engineering approach to the linguistic description of motion control tasks [20]-[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 have implemented a set of control primitives (akin to the "atoms" and "behaviors" of MDL [24]). These are building blocks in terms of which grasping and manipulation tasks can be described. Our goal was to implement the beginnings of a motion language for use with multifingered 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 the subject (e.g., [25]). When it comes to the control of electromechanical systems, however, 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. These sets are treated as assembly code,

and we designed low-level routines that 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 the motor uses the command as a set point 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.
- *Feedforward Control:* The controller transmits a predefined sequence of control inputs to the motors (in real time).

Hardware-Independent Primitives

We would like motion description languages to have as much portability as 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. This constitutes an important but tedious process that is beyond the scope of this article. In the interest of space, we omit many of the details; however, we do want to avoid the gross oversimplification sometimes found in AI texts and wish to call attention to the fact that behind each of the following primitives 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 (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 toward 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) 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 include 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 (1) that 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 prespecified distance on either side. When the object's velocity decreases below a specified threshold, the fingers are moved quickly toward 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, to compute the incremental joint motions required to bring about a desired incremental object motion by means of (2) (see also [17]-[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 (2) in a feedback control loop. Given a desired object trajectory (for an object in the manipulator's workspace), the control-

Lack of time to communicate is an important constraint compared to lack of computational power. The HRL manipulator was used to explore aspects of this "coupling" between control and communication.

ler uses low-level and point-to-point manipulation primitives to move the object to each of a sequence of configurations, all lying on the desired trajectory.

A Note on Software Architecture

We do not expect the reader to be interested in coding details (which are available from the authors); however, we do want to give a general idea of the architecture of our software, as well as the manner in which primitives interact with each other. Following the paradigm of MDL [23], [24], [26], each hardware-independent primitive was implemented as a feedback loop that can be interrupted based on sensor data (for example, a vision sensor indicates that an object has been located and the manipulator should now move to grasp it). Once a primitive is interrupted, another begins to run. This procedure takes place on a computer (called the "control processor") that transmits control inputs to the actuators and decides which primitive to run at a given time.

Due to the complexity of our sensor suite, a significant amount of preprocessing must take place before the raw data (e.g., pixel values) are converted to information that can be used by a primitive (e.g., coordinates of an object-fingertip contact). For this reason, all code that implements sensing operations is executed on a dedicated second processor (called the "sensing processor"). The sensing processor maintains a polling table with the addresses of some or all of the available sensing routines. During each feedback cycle, these routines are executed and their output(s) are returned to the control processor. Of course, each primitive might have different requirements with respect to which sensors must be polled (e.g., tactile sensors only versus tactile sensors and overhead camera) or how the raw data should be processed (e.g., measuring local curvature at a contact versus simply detecting contact). For this reason, our implementation allows for a primitive to dynamically interrupt the sensing processor and reset its polling schedule to include a new subset of sensors. Such an action causes the sensing processor to begin running a new set of routines and posting the resulting data back to the control processor for use by the primitive that made the request. This arrangement makes the software efficient (because a sensor does not consume CPU time if there is no demand for its output) and enforces the separation between the hardware-independent algorithm and the data gathering necessary to drive that algorithm.

As we have indicated, feedback loops running on the control processor are closed by data arriving from the sensing processor. Of course, the highest frequency at which sensor data can be generated (and thus the highest rate at which a feedback loop can run) depends on the sensing modalities and amount of preprocessing that have been requested by the corresponding control primitive. In addition, because the sensing processor does not run a real-time operating system, there is a slight variability in the time intervals between data arrivals at the control processor. The resulting "jitter" in the feedback loop does not seem to cause a problem in the operating domains we have explored.

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 to reexamine 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 that 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 to the necessity for 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 manip-

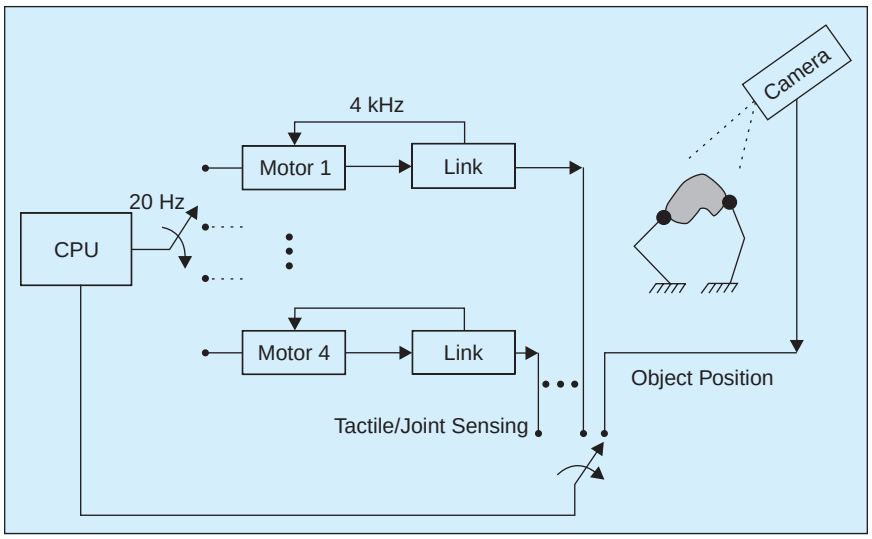


Figure 12. *Low-level control architecture.*

ulator 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 choose which motors and sensors to exchange information with at a particular time. 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 and that captures characteristics of a rather general class of hybrid control systems.

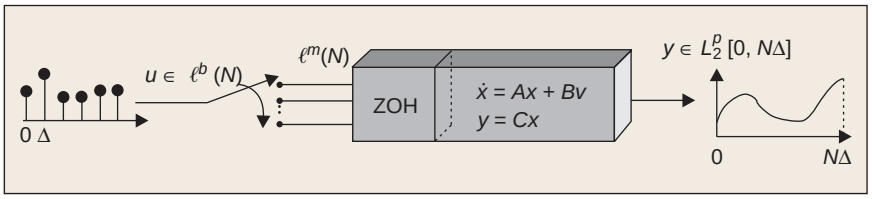


Figure 13. *A computer-controlled system.*

A Model for Limited Communication Control

Consider a continuous-time, linear time-invariant (LTI) system that 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" [27], 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

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

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 $\Sigma_j \sigma_j(i) \le b$ for $i = 0,\ldots,N-1$. It is worth noting that the definition of a communication sequence allows us to quantify the amount of "attention" 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 [28] and stabilization [29] 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: $\Lambda_{(G,\Delta,N)}(t): \ell^m(N) \to L_2^p[0,N\Delta]$. 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

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

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 $M(\sigma)$

$$M: \ell^b(N) \to \ell^m(N).$$

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 $\sigma' \in \text{Range}(M)$ 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 [28]). 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:

$$x_d(t) = 4.2\cos(t)$$
$$y_d(t) = 4.5 + 1.7\sin(2t), \quad t \in [0,1]\ \text{s}. \tag{3}$$

For 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 an LTI system, then the question is one of "inverting" that system to find the input that will produce a desired output [30]. We must make a distinction here between the Moore-Penrose inverse, which is what we look for in this section, and 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 $L_2^p[0,T]$:

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

then our trajectory-following problem 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

$$\|y_d - \Lambda M u\|_{L_2}^2. \tag{4}$$

It can be shown [28] that if the original LTI system has normal rank (i.e., $\limsup_{|s|=1}\text{rank}[C(sI-A)^{-1}B] = m$), and if the controller communicates with every input at least once over the course of the task, then the operator ΛM is one-to-one and thus has a generalized (Moore-Penrose) inverse given by

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

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

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

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

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

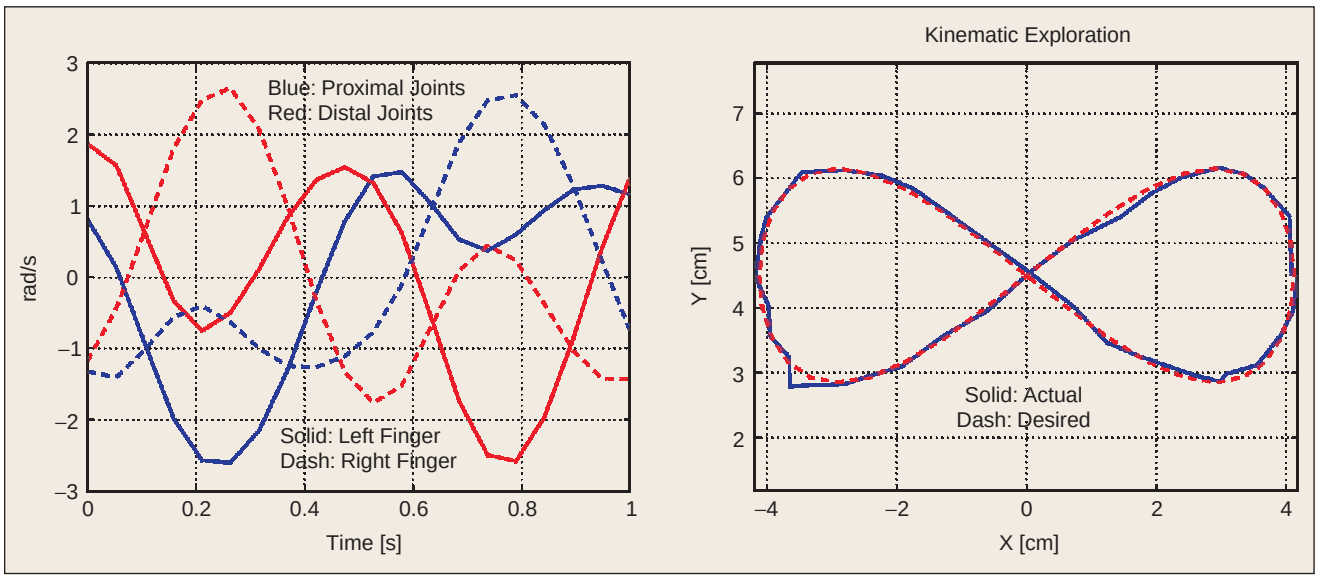


Figure 14. *Nominal joint and object trajectories, figure-8.*

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

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

Equation (5) can be considered an analog 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$(K) = n$:

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

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 (3) (shown in Fig. 14). A communication sequence was selected, and the generalized inverse of (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 joints are labeled as follows: 1—left proximal, 2—left distal, 3—right proximal, and 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 (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 $u_k = \int_{t_k}^{t_{k+1}} u(t)dt / (t_{k+1} - t_k)$. Fig. 16 shows the input/output pair corresponding to that approach. Tracking performance was quite poor, with an L_2 error of 12.1.

Nonuniform 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:

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

again using the standard basis vectors in $\mathbb{R}^4$ to specify which input gets updated at a particular time. Notice that distal joints (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 $\Delta = 0.05$ s, we allocated communication cycles using as a guide the amount of rotation required by each joint over that interval. Tracking performance was slightly improved compared to what was achieved with uniform attention. The L_2 error was 3.2 (Fig. 17).

Conclusions and New Vistas

We have described an experimental facility designed to study aspects of multimodal 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

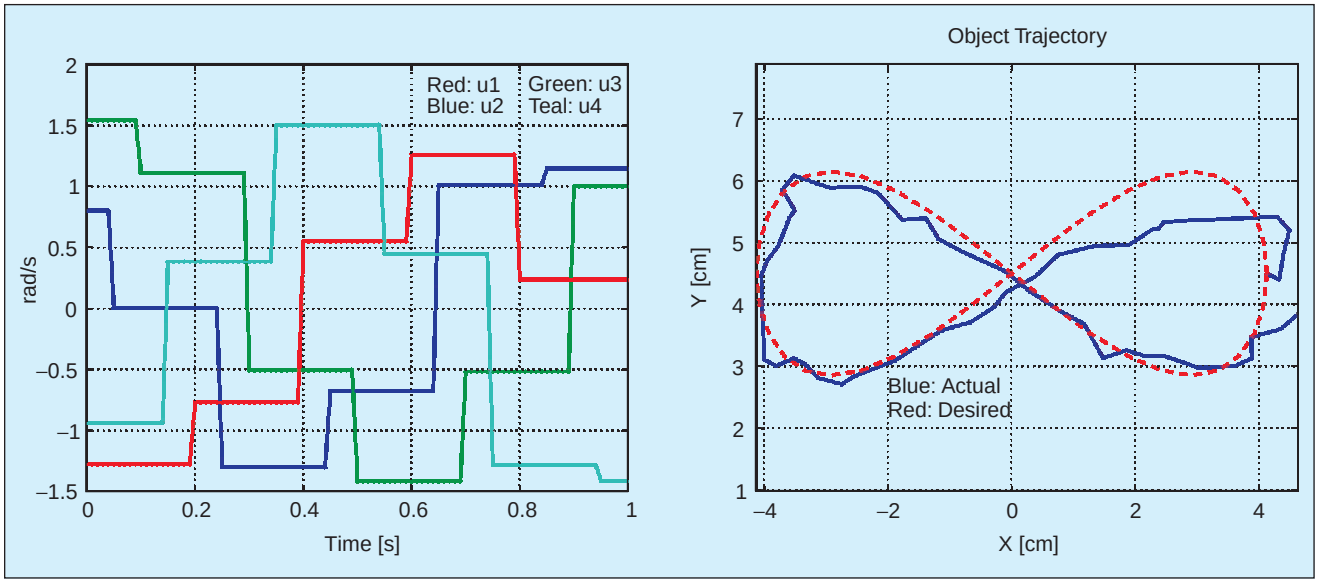


Figure 15. *Optimal inputs (uniform attention) and resulting object trajectory.*

gains maturity, new issues are emerging related to our understanding of larger, multicomponent 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" that 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 who have used the system over the past 15 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 upgradability. 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 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

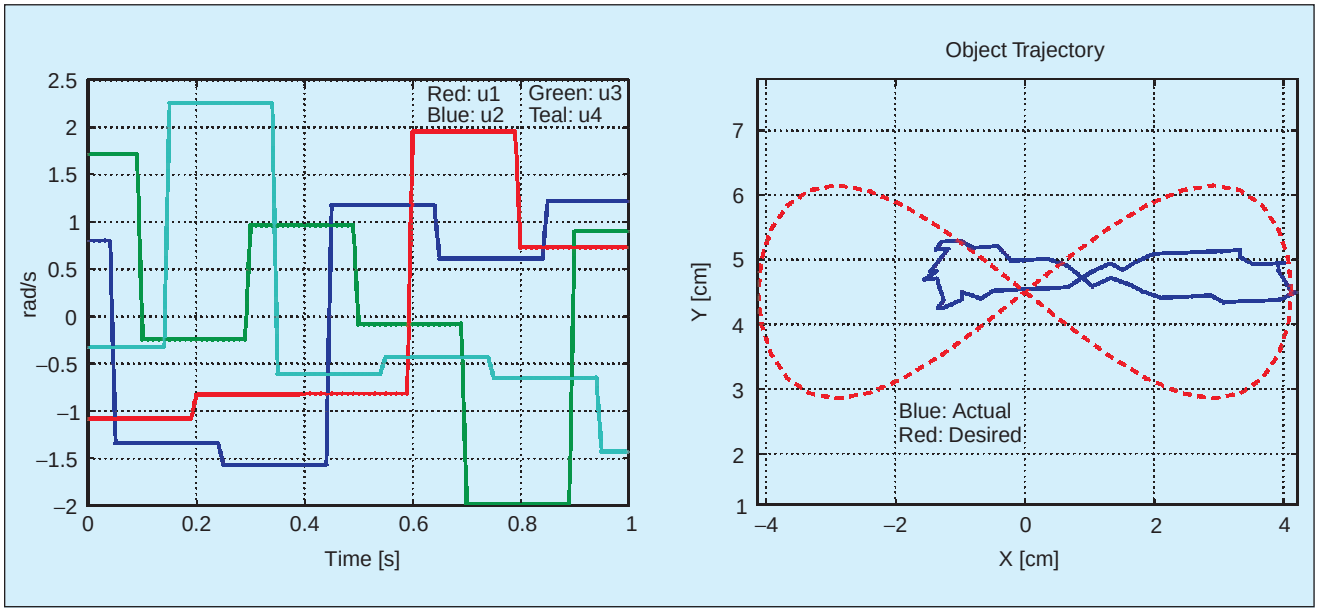


Figure 16. *"Average" inputs (uniform attention) and resulting object trajectory.*

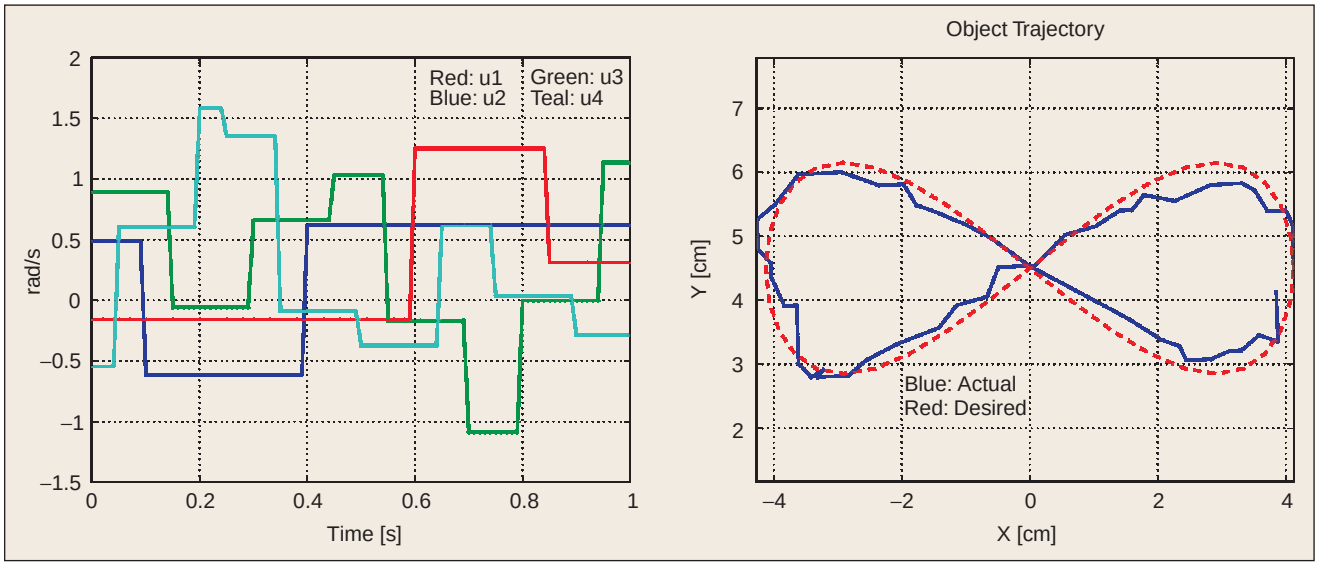


Figure 17. *Optimal inputs (nonuniform attention).*

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 network or of a 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 15 undergraduate research projects at Harvard, under the direction of the second author. It has influenced our teaching 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 a useful paradigm for intelligent control.

Acknowledgment

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

References

[1] J. Kerr and B. Roth, "Analysis of multifingered hands," *Int. J. Robot. Res.*, vol. 4, no. 4, pp. 3-17, Winter 1986.

[2] D.J. Montana, "The kinematics of contact and grasp," *Int. J. Robot. Res.*, vol. 7, no. 3, pp. 17-32, 1988.

[3] R.W. Brockett, "Robotic hand with rheological surfaces," in *Proc. 1985 IEEE Int. Conf. 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. J. Robot. Res.*, 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. 2000 IEEE Int. Conf. Robotics and Automation*, Apr. 2000, pp. 508-513.

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

[7] R.A. Russell, "Compliant-skin tactile sensor," in *Proc. 1987 IEEE Int. Conf. Robotics and Automation*, June 1987, pp. 2221-233.

[8] W.D. Hillis, "Active touch sensing," *Int. J. Robot. Res.*, vol. 1, no. 2, pp. 33-44, 1982.

[9] R.S. Fearing, "Tactile sensing mechanisms," *Int. J. Robot. Res.*, 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. Berlin: IFS Publications, Springer-Verlag, 1986, pp. 145-155.

[11] S. Begej, "Planar and finger-shaped optical tactile sensors for robotic manipulation," *IEEE Trans. Robot. Automat.*, 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.1992 IEEE Int. Conf. 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. J. Robot. Res.*, vol. 15, no. 4, pp. 335-350, 1996.

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

[15] N.J. Ferrier and R.W. Brockett, "Reconstructing the shape of a deformable membrane from image data," *Int. J. Robot. Res.*, vol. 19, pp. 1-22, 2000.

[16] D. Hristu, "Optimal control with limited communication," Ph.D. dissertation, Harvard Univ., Division of Engineering and Applied Sciences, 1999.

[17] J. Kerr, "An analysis of multifingered hands," Ph.D. dissertation, Stanford Univ., Dept. Mechanical Engineering, 1984.

[18] D.J. Montana, "Tactile sensing and the kinematics of contact," Ph.D. dissertation, Harvard Univ., 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 Int. Conf. Robotics and Automation*, June 1995, pp. 743-750.

[20] R.W. Brockett, "On the computer control of movement," in *Proc. 1988 IEEE Conf. Robotics and Automation*, April 1988, pp. 534-540.

[21] R.W. Brockett, "Formal languages for motion description and map making," in *Robotics*, J. Baillieul, D.P. Martin, R.W. Brockett, and B.R. Donald, Eds. American Mathematical Society, 1990, pp. 181-93.

[22] R.M. Murray, D.C. Deno, K.S.J. Pister, and S.S. Sastry, "Control primitives for robot systems," *IEEE Trans. Syst., Man, Cybernet.*, vol. 22, no. 1, pp. 183-193, 1992.

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

[24] R.W. Brockett, "Language driven hybrid systems," in *Proc. 33rd IEEE Conf. 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*. Reading, MA: Addison-Wesley, 1995.

[26] D. Hristu, "The MDLe engine: A software tool for hybrid motion control," Institute for Systems Res., University of Maryland, Tech. Rep., 2001.

[27] R.W. Brockett, "Stabilization of motor networks," in *Proc. 34th IEEE Conf. Decision and Control*, 1995, pp. 1484-1488.

[28] D. Hristu, "Generalized inverses for finite-horizon tracking," in *Proc. 38th IEEE Conf. Decision and Control*, 1999, vol. 2, pp. 1397-402.

[29] D. Hristu and K. Morgansen, "Limited communication control," *Syst. Contr. Lett.*, vol. 37, no. 4, pp. 193-205, 1999.

[30] R.W. Brockett, "The invertibility of dynamic systems with application to control," Ph.D. dissertation, Case Western Reserve Univ., 1964.

Dimitrios Hristu-Varsakelis is an Assistant Professor in the Department of Mechanical Engineering and an Affiliate member of the Institute for Systems Research at the University of Maryland, College Park. He received the B.S. degree in electrical engineering and computer science from the University of California, Berkeley, in 1992 and the M.S. degree in electrical engineering from Rensselaer Polytechnic Institute in 1994. From 1994-1999, he was a graduate student in the Division of Engineering and Applied Sciences at Harvard University and received the M.S. degree in applied mathematics and the Ph.D. in engineering sciences. His research interests include control with limited communication, hybrid systems, and intelligent machines. He is a member of the IEEE and a recipient of the 1999 Eliahu Jury award from the Division of Engineering and Applied Sciences, Harvard University.

Roger W. Brockett is An Wang Professor of Electrical Engineering and Computer Science in the Division of Applied Sciences at Harvard University. He has worked on various aspects of control theory and applications, including recent work in hybrid systems involving elements of computing as well as control. He received the American Automatic Control Council's Richard Bellman Control Heritage Award and the IEEE's Control Systems Science and Engineering Award. He is a Fellow of the IEEE and a member of the National Academy of Engineering.