# Control Experiments in Planar Manipulation and Grasping*

**Richard M. Murray and S. Shankar Sastry**
Department of Electrical Engineering and Computer Science
Electronics Research Laboratory
University of California, Berkeley, CA 94720

## Abstract

Many algorithms have been proposed in the literature for control of multi-fingered robot hands. This paper compares the performance of several of these algorithms, as well as some extensions of more conventional manipulator control laws, in the case of planar grasping. A brief introduction to the subject of robot hands and the notation used in this paper is included.

## 1 Introduction

Multifingered robot hands can be used to increase the fine motion capabilities of a robot manipulator. Like a human hand, a robot hand can accurately perform small manipulations of a wide variety of objects. A great deal of research has focused on the kinematic issues of hands and the generation of stable grasps (see for example [20], [10] and [16]). More recently the problem of generating feedback control laws for coordinated manipulation of an object being grasped by a robot hand has been considered ([1], [12], [4]). A related area of interest is the control of multiple robots performing a single coordinated task([26], [6], [23], [15]).

There are several articulated hands that have been developed to study problems in grasping and manipulation ([20], [8], [18]). Many of these hands, such as the MIT/Utah hand [8] and the Stanford/JPL hand ([20], [24]) are quite complex and require sophisticated computer architectures to control them ([21], [3]). Unfortunately, this combination makes the implementation of proposed control algorithms correspondingly complex. Others, such as the NYU hand [5], use stepper motors as joint actuators and make control schemes which command joint torques more difficult to implement. As a result of these obstacles, few experimental results of the algorithms proposed by researchers are available.

To provide a facility for experimental verification of control algorithms we have built a very simple two-fingered planar hand. Due to the simplicity of its design, the implementation of a control algorithm is simplified and the performance of the hand can be studied more quickly and easily. Additionally, an intuitive understanding of the structure of a control law can often be reached.

This paper compares the performance of several control schemes implemented on this hand. In order to allow the control laws to be used in more complicated environments, we present the algorithms for the general case of a many-fingered hand operating in three dimensions and simplify to the planar case only when necessary.

## 2 Grasp dynamics

This section provides a brief introduction to grasping and the notations used in this paper. For a more complete discussion of the kinematics of grasping see [9]. The dynamics outlined here were developed in [12] and [4] and can also be found in [14].

### 2.1 Hand kinematics

A *contact* between a finger and an object is described by a mapping between forces exerted by the finger at the point of contact and the resultant force and torque at some reference point on the object (say the center of mass). If we have $k$ fingers contacting the object, then the net force on the object is the sum of the forces due to each finger. The *grasp map*, $G$, is the map between finger forces and the resultant object force. We represent this map as $6 \times n$ matrix, where where $n$ is the number of contact forces generated by the fingers. If $F_o$ represents the forces and torques exerted on the object in the palm reference frame and $f_{c_i}$ is the force exerted by the $i^{th}$ finger in that same frame, then

$$F_o = \begin{bmatrix} G_1 & \cdots & G_k \end{bmatrix} \begin{bmatrix} f_{c_1} \\ \vdots \\ f_{c_k} \end{bmatrix} = G f_c, \qquad \begin{matrix} F_o \in R^6 \\ f_c \in R^{n_1+\cdots+n_k} \end{matrix} \tag{1}$$

The grasp map is a function of the position and orientation of the object as well as that of the fingertips.

The null space of the grasp map corresponds to finger forces which cause no net force to be exerted on the object. We call the forces on the object resulting from finger forces which lie in the null space of G, denoted $\mathcal{N}(G)$, *internal* or *null forces*. It is these internal forces which allow us to grip an object.

The velocity of the contact points can be related to the velocity of the object using the principle of virtual work. If $x_c$ and $X_o$ represent the positions of the fingertips and object, respectively, then it is straightforward to show

$$\dot{x}_c = G^T \dot{X}_o. \tag{2}$$

We are interested in using kinematic mechanisms as the fingers of our hand. For each finger $i$ we associate a forward kinematic map $K_i : R^{m_i} \to R^6$ which takes a joint position to an end effector position. Taking the derivative of this map about a point, $\theta_i$, and stacking the resulting Jacobians for each finger, we get the Jacobian of the forward kinematic map for the hand, $J_h$, and

$$\dot{x}_c = J_h(\theta)\dot{\theta} \tag{3}$$

Again we can apply the principle of virtual work to relate the forces at the contact points, $f_c$, to the individual joint torques, $\tau \in R^{m_1} \times R^{m_2} \times \cdots \times R^{m_k}$,

$$\tau = J_h^T(\theta) f_c \tag{4}$$

### 2.2 Hand dynamics

The dynamics of a robot manipulator, and in particular a single finger of a hand, can be represented as a differential equation with respect to the joint angles, $\theta_i$,

$$M_i(\theta_i)\ddot{\theta}_i + C_i(\theta_i, \dot{\theta}_i)\dot{\theta}_i + N_i(\theta_i, \dot{\theta}_i) = \tau_i - J_i^T f_{c_i} \tag{5}$$

where $M_i(\theta_i) \in R^{n_i \times n_i}$ is the symmetric moment of inertia matrix for the $i^{th}$ finger, $C_i(\theta_i, \dot{\theta}_i)\dot{\theta}_i \in R^{n_i}$ is a vector of coriolis and centrifugal terms, $N_i(\theta_i, \dot{\theta}_i) \in R^{n_i}$ is a vector of gravity and friction forces and $\tau \in R_n$, is the vector of applied joint torques. The final term in equation 5 is the torque due to the force applied at the fingertip. It is the addition of this term that causes coupling between the fingers (due to the object being grasped). We also note that with proper definition of $C_i(\theta_i, \dot{\theta}_i)$ the matrix $\dot{M}_i - 2C_i$ is skew-symmetric (see [7] or [17]).

Stacking the equations for all the fingers in the hand we can write the hand dynamics as

$$M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + N(\theta, \dot{\theta}) = \tau - J_h^T f_c \tag{6}$$

The dynamics of the object are governed by the Newton-Euler equations. Expressed in the base (inertial) frame, these equations can be

*Research supported in part by NSF under grant DMC-84-51129 and NASA under grant NAG 2-243



CH2750-8/89/0000/0624$01.00 © 1989 IEEE

written in terms of the object position, $x_o$ and angular velocity, $\omega_o$, with respect to the center of mass

$$\begin{bmatrix} m_o I & 0 \\ 0 & I_b \end{bmatrix} \begin{bmatrix} \ddot{x}_o \\ \dot{\omega}_o \end{bmatrix} + \begin{bmatrix} 0 \\ \omega_o \times I_b \omega_o \end{bmatrix} = \begin{bmatrix} f_o \\ \tau_o \end{bmatrix} \tag{7}$$

where $m_o I \in R^{3\times 3}$ is the object mass matrix, and $I_b \in R^{3\times 3}$ is the object inertia matrix. Note that the inertia matrix is a function of object orientation and be written as $I_b = R_o I_o R_o^T$ where $R_o \in SO(3)$ is the rotation matrix between the base coordinate frame and a coordinate frame affixed to the object.

Letting $M_o$ represent the combined mass and inertia matrix, $X_o$ represent the position and orientation of the object and $F_o$ represent the forces and torques applied to the object at its center of mass, we can write equation 7 more simply as

$$M_o \ddot{X}_o + C_o(X_o, \dot{X}_o)\dot{X}_o = F_o = G f_c + f_e \tag{8}$$

where the second equality follows from equation 1 and $f_e$ is due to external forces applied to the object (such as gravity). Once again if we define $C_o$ carefully it can be shown that $\dot{M}_o - 2C_o$ is skew-symmetric.

Equations 6 and 8 represent the dynamics of a hand grasping an object. The only additional equation needed is the specification of the contact force, $f_c$. In general this force depends on the characteristics of the fingertip and the compliances of the hand and object. For our purposes we will assume that all bodies are rigid and the contacts are never broken. In this case the fingertips are constrained to move at the same speed as the contact points on the object. This constraint can be written as

$$J_h \dot{\theta} = G^T \dot{X}_o \tag{9}$$

where $J_h\dot{\theta}$ is the vector of fingertip velocities and $G^T\dot{X}_o$ is the velocity of the contact points on the object.

As shown in [14], equations 6, 8 and 9 can be combined to yield

$$M_h(X_o)\ddot{X}_o + C_h(X_o, \dot{X}_o)\dot{X}_o + N_h(X_o, \dot{X}_o) = G J_h^{-T} \tau - f_e \tag{10}$$

where

$$\begin{aligned} M_h &= M_o + G J_h^{-T} M(\theta) J_h^{-1} G^T \\ C_h &= C_o + G J_h^{-T} \left( C(\theta, \dot{\theta}) J_h^{-1} G^T + M(\theta) \frac{d}{dt}\left(J_h^{-1} G^T\right) \right) \\ N_h &= G J_h^{-T} N(\theta, \dot{\theta}) \end{aligned}$$

and $\dot{M}_h - 2C_h$ is skew-symmetric. This is the description of the hand dynamics in object coordinates. We have assumed here that the finger coordinates, $\theta$, are derivable from the object position, $X_o$ —this holds for point contact models, but can fail for rolling contacts in three dimensions. See [4] for a more thorough discussion.

In the case of planar grasping the object dynamics are somewhat simplified since the object is only allowed to rotate about the axis perpendicular to the plane of motion. If we represent the position and orientation of the object as $(x, y, \phi)$ and the inertia of the object as $I_o \in R$ we have

$$\begin{bmatrix} m_o & 0 & 0 \\ 0 & m_o & 0 \\ 0 & 0 & I_o \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \\ \tau_\phi \end{bmatrix} \tag{11}$$

and hence $C_o = 0$. Furthermore, in three dimensions a parameterization of $SO(3)$ (the space of rigid body rotations) must be selected and this can add some complexity to the dynamics. Since there is only one axis of rotation in the planar case, the representation of the orientation of the object is particularly simple.

# 3 Control Algorithms for Grasping

## 3.1 The grasping control problem

Position control of a multi-fingered hand can be broken into two parts

1. tracking – the center of mass of the object should follow a specified trajectory.
2. holding – the finger forces should lie within the friction cone for each finger at all times.

Condition 2 is important not only because we do not wish to lose our grip on the object, but also because we assumed in our derivation of the grasp dynamics that contact was maintained. Without this constraint we would have to specify the dynamics of contact.

If we choose a grasp properly it can be shown that given an arbitrary set of finger forces, $f_c$, we can find an internal force, $f_N \in \mathcal{N}(G)$, such that the combined force $f_c + f_N$ is inside the friction cone [4]. Thus, given a force generated to solve the tracking problem, we can always add a force to this such that condition 2 is satisfied. Since internal forces cause no net motion of the hand or object, this additional force does not affect the net force exerted by the fingers on the object. We shall assume in the sequel that such an internal force is available at all times. Section 3.6 discusses the choice of this force in more detail.

To satisfy the tracking problem, we will examine several different algorithms. Each of the algorithms makes different assumptions about the grasp dynamics and all of them assume that arbitrary internal and external forces can be generated by the fingers on the object.

## 3.2 Individual joint control

The first algorithm we will study is based on the idea that we can specify the trajectory of the object by transforming that trajectory into the space of the joint variables and then controlling the fingers individually. In using this approach we intentionally neglect the dynamics of the object and concern ourselves only with the finger dynamics. Thus, we model the system as

$$M(\theta) + N(\theta, \dot{\theta}) = \tau \tag{12}$$

We are given the desired joint trajectory, $\theta_d$, and its acceleration, $\ddot{\theta}_d = G^T J_h^{-1} \ddot{X}_d + \frac{d}{dt}(G^T J_h^{-1}) \dot{X}_d$, which is calculated using the inverse kinematic map between the object location and the joint positions. In general this map is not unique, but often it is not hard to choose between solutions. For example, of the four possible solutions for a two-fingered hand in the plane, only one is usually feasible since the other solutions intersect the object.

To control each individual joint we use a computed torque control law ([2], [13]).

$$\tau = M(\theta)\left(\ddot{\theta}_d + K_v \dot{e}_\theta + K_p e_\theta\right) + N(\theta, \dot{\theta}) \tag{13}$$

where $K_v$ and $K_p$ are positive definite matrices and $e_\theta = \theta_d - \theta$. Our error dynamics become

$$M(\theta)\left(\ddot{e}_\theta + K_v \dot{e}_\theta + K_p e_\theta\right) = 0 \tag{14}$$

We can now choose $K_v$ and $K_p$ so that the error dynamics are exponentially stable. i.e. $x \to x_d$.

We must also add a null force term to grip the object. Since the null force term does not cause any motion in the object, it does not affect the dynamics of the links and our tracking stability remains unchanged. If we are given a null force term $f_N$, then we can apply this force by adding a joint torque of $J_h^T f_N$ (the null force applied at the finger tips, reflected back to the joints). The final control law is then

$$\tau = M(\theta)\left(\ddot{\theta}_d + K_v \dot{e}_\theta + K_p e_\theta\right) + N(\theta, \dot{\theta}) + J_h^T f_N \tag{15}$$

## 3.3 Force transformation

If we had only the object dynamics to consider (equation 8), we could use a computed torque control law in object coordinates having the form

$$F_o = M_o\left(\ddot{X}_d + K_v \dot{e}_x + K_p e_x\right) \tag{16}$$

This gives us the desired forces (and torque) to be applied to the center of mass of the object. We can find the forces that would have to be applied at the fingertips to get such an object force by premultiplying by $G^+ = G^T(GG^T)^{-1}$, a pseudo-inverse of $G$. This transforms object forces to finger forces having minimum norm (i.e., zero internal force component). Similarly, we can transform the finger forces to joint torques by premultiplying by $J_h^T$. Thus to generate an object force as in equation 16 we apply torque

$$\tau = J_h^T G^+ M_o\left(\ddot{X}_d + K_v \dot{e}_x + K_p e_x\right) \tag{17}$$

This algorithm takes more calculation than the joint control algorithm since typically we must calculate the position of the object given

the joint angles before we can apply the control law. This calculation requires sine and cosine calculations (for the fingertip locations) and an arctangent operation (for the orientation of the object). Once the control law is calculated we must multiply by $M(\theta)J_h^{-1}G^T$ which requires at most $n^2$ multiplications ($M(\theta)J_h^{-1}G^T$ is *not* block diagonal).

To speed up the overall control sample rate we can break the calculation into two pieces. The update loop calculates $M(\theta)J_h^{-1}G^T$ and $J_h f_N$ while the control loop carries out the matrix multiplications and additions. If the trajectory that the object is following is changing slowly, then we find empirically that we can run the update loop more slowly and speed up the control loop.

### 3.4 Generalized Computed Torque

Since the hand dynamics in equation 10 have the same basic form as the dynamics of a simple manipulator, it is straightforward to extend manipulator control laws into hand control laws. If we use the computed torque paradigm with equation 10 we get

$$\tau = J_h^T G^+ \left[ M_h \left( \ddot{X}_d + K_v \dot{e}_x + K_p e_x \right) + C_h \dot{X} + N_h \right] + J_h^T f_N \qquad (18)$$

This gives an error equation in object coordinates of

$$M_h \left( \ddot{e}_x + K_v \dot{e}_x + K_p e_x \right) = 0 \qquad (19)$$

and away from singularities of $J_h$, our dynamics are governed by

$$\ddot{e}_x + K_v \dot{e}_x + K_p e_x = 0 \qquad (20)$$

The computational resources required to implement this algorithm are considerable. Not only must we calculate the inertia matrix, but we have to cancel all the nonlinear terms. These nonlinear terms contain many trigonometric calculations but they can be minimized by the use of carefully constructed lookup tables. This algorithm was originally proposed for multi-fingered hands in [12] and has been extended to the case of rolling contacts in [4].

### 3.5 Natural and Stiffness Controllers

Natural control was proposed by Koditschek in 1984 [11] as an alternative to computed torque which did not rely on exact cancellation of nonlinear dynamics. It relies on the skew-symmetric property of the robot dynamics, $\alpha^T \left( \dot{M} - 2C \right) \alpha = 0$ for all $\alpha \in R^n$. It is an extension of the simpler PD control law and it can be shown to be asymptotically stable for the tracking control problem. The natural control law has the form

$$\tau = J_h^T G^+ \left[ M_h \ddot{X}_d + C_h \dot{X}_d + N_h + K_v \dot{e}_x + K_p \dot{e}_x \right] + J_h^T f_N \qquad (21)$$

where $K_p$ and $K_v$ are again positive definite gain matrices.

Stiffness control is a slight extension of natural control that has been shown to have exponentially stable error dynamics. Forms of this control law with proofs of stability can be found in [19] and [22]. The control law presented here is a slight generalization of the laws proposed by others, but similar forms can be found in the literature [25]. The generalization leads to a more complicated proof of stability but allows more control over the resulting stiffness. We use the control law

$$\tau = J_h^T G^+ \left[ M_h \left( \ddot{X}_o + \lambda \dot{e} \right) + C_h \left( \dot{X}_o + \lambda e \right) + N_h + K_p e + K_v \dot{e} \right] + J_h^T f_N \qquad (22)$$

where $\lambda > 0$ and $K_p$, $K_v$ are positive definite.

The complexity of these algorithms is roughly the same as the computed torque control law.

### 3.6 Choosing the grasp force, $f_N$

All of the algorithms have relied on the choice of a grasping force, $f_N \in \mathcal{N}(G)$ which maintains contact between the fingertips and the object by insuring that the finger forces lie in the friction cone. There are several possible methods for calculating this term. Since ideally $f_N$ does not affect the force applied to the center of mass of the object we should be free to choose $f_N$ without worrying about its effect on the tracking control problem. The simplest $f_N$ is a constant $f_N$. It must be large enough so that finger forces never leave the friction cone over the entire trajectory of the object. Generally this requires a knowledge of the bounds on the external forces that can be exerted on the object. The advantage of this approach is that $J_h^T f_N$ can be calculated at the same rate as $J_h$—saving computation time.

One difficulty is that in a real-world hand the maximum motor torques that can be generated are finite. Thus, we are not guaranteed that we can apply an $f_N$ which causes $f_c + f_N$ to lie in the friction cone without saturating the motors. Another issue is the effect of the null force term in the presence of errors. If a large internal force term is used and, due to sensor or actuator errors, it does not actually lie in the null space of the grasp matrix, the resulting force can cause positioning errors and in the extreme case, instability.

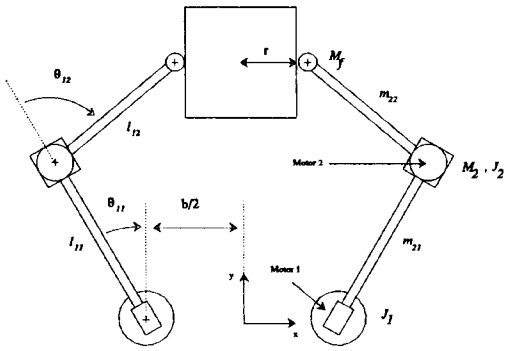


Figure 1: Top view of Styx

| | | |
|---|---|---|
| Link lengths | $l_{11}$, $l_{21}$ | 15.24 cm |
| | $l_{12}$, $l_{22}$ | 12.16, 12.55 cm |
| Fingertip radius | $r_f$ | 1.91 cm |
| Base separation | $b$ | 20.32 cm |
| Link mass | $m_{11}$, $m_{21}$ | 53 g |
| | $m_{12}$, $m_{22}$ | 17, 20 g |
| Motor mass | $M_2$ | 328 g |
| Motor inertia | $J_1$ | 18 g cm$^2$ |
| | $J_2$ | 1.74 g cm$^2$ |
| Fingertip mass | $M_f$ | 3 g |

Table 1: Parameter values for Styx

## 4 Experimental comparison of algorithms

The algorithms presented in the previous chapter have been implemented on a multi-fingered hand known as Styx. Styx is a planar two-fingered hand built at the University of California, Berkeley to test different multi-fingered hand control algorithms. A labeled diagram of Styx is shown in figure 1; the parameter values can be found in table 1.

The motors used in Styx are direct drive DC motors mounted at the base of each joint and are driven with a pulse width modulated 20 kHz square wave. Since the second motor is mounted at the end of the first joint, the dynamics of the first joint are relatively independent of the configuration of the robot (i.e., $M(\theta)$ can be considered to constant). Each motor contains a quadrature encoder which is used to sense position. The resolution of this encoder is 500 lines/revolution, which generates 2000 edges (or counts) per 360 degree rotation.

Styx is connected to an IBM PC/AT running at 8.0 MHz with an 8087 floating point coprocessor. The motors and encoders are interfaced to the AT using a set of four HP HCTL-1000 motion control chips interfaced to the AT bus. The HCTL chips generate a pulse width modulated signal which is fed to an amplifier. The quadrature signal from the position encoders is connected directly to the chip inputs. Although the HCTL-1000 is capable of on-chip position and velocity control of a motor, this feature was not used in the experiments presented here. All control algorithms were implemented on the IBM PC/AT using the HCTL-1000 solely as an interface to the motors.

The software to drive Styx is composed of an assembly language scheduler which controls the sample rate of the inner (control) and outer (update) loops. The algorithms themselves are written in the C programming language and compiled using the Microsoft 5.1 Optimizing

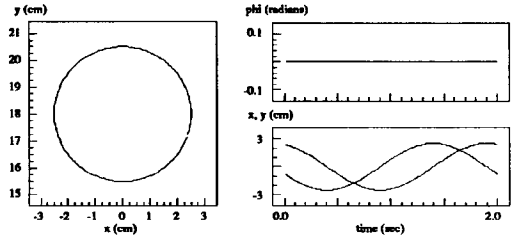

Figure 2: Desired trajectory

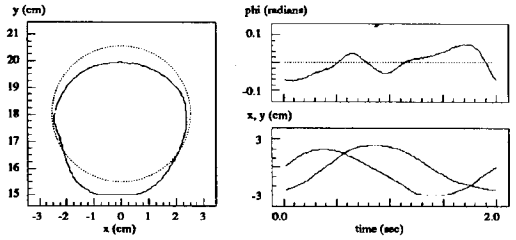

Figure 3: Joint control algorithm

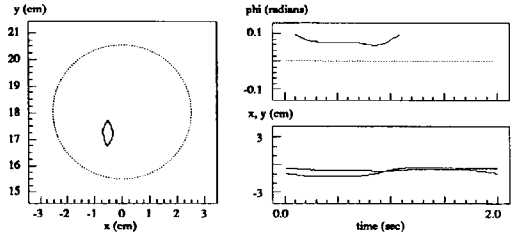

Figure 4: Force transformation algorithm

C compiler. Control rates of 120 Hz have been achieved for the more complicated controllers by careful use of table lookups and coding. For the results presented here, all controllers were run with a control loop frequency of 100 Hz and an update loop frequency which was varied between 25 Hz and 100 Hz, depending on the algorithm and the type of comparison performed.

All of the algorithms implemented on Styx made some basic simplifying assumptions:

1. Motor dynamics can be ignored – for small velocities the torque generated by a motor is roughly proportional to the input pulse width. Additionally, friction terms were left out of the dynamic model (i.e., $N(\theta, \dot{\theta}) = 0$). This included a relatively large deadband between the applied motor current and the measured joint torque due to static friction. This deadband was approximately 10% of the maximum motor current that could be generated by the amplifiers.
2. Jacobian and mass matrices change slowly – since the trajectories commanded were slow relative to the control rate, the Jacobian and mass matrices did not need to be recalculated in the control loop. The update loop, which runs at a slower rate, was used instead. For Styx, the Jacobian requires a much more CPU intensive calculation than the basic control law (which is similar for all controllers) so that the update rate is the limiting factor in the speed of an algorithm.
3. Fingers can be modeled as point contacts – the actual finger tips used on Styx were small rubber circles. To avoid the added computational complexity required to model the rolling contacts, the fingertips were modeled as simpler point contacts. For Styx this meant that the center of mass of the object shifted slightly as the object moved through its trajectory. This shift is typically less than 2.5 mm.

Several different trajectories were traversed with each control law. A single circular trajectory is presented here to conserve space and to allow easy visual interpretation of the results. The trajectory shown in figure 2 is a circle with a diameter of 5 cm and a period of 0.5 Hz. This circle is too fast for most of the controllers to track accurately but was chosen to emphasize sources of error in the controllers. The object being manipulated is a cardboard box with radius 15.18 cm and mass 33 g. The box was allowed to slide on a wooden table; the friction between the box and the table was ignored. Heavier boxes were difficult to manipulate due to limits on the maximum joint torques that could be generated with the current hardware.

Since we are comparing different control algorithms we must decide on what criterion to judge an algorithm. Although the final test of an algorithm should be how well it satisfies a set of design criteria, other comparisons can be useful to get a feel for the relative strengths of an algorithm. Three types of comparisons are presented here.

## 4.1 Fixed Gain comparisons

The *fixed gain* comparison uses a fixed $K_p$ and $K_v$ for each of the algorithms. $K_p$ and $K_v$ were chosen by selecting the cutoff frequency, $\omega_n$, and the damping factor, $\zeta$, for the closed loop system using any of the computed torque based schemes (which give a linear error equation). Given these two values, we set $K_p = \omega_n^2 I$ and $K_v = 2\zeta\omega_n I$. For all of the experiments presented in this section, $\omega_n$ was chosen as 2.5 Hz, or one tenth of the (Jacobian) update frequency. This value of $\omega_n$ was used so that noise introduced into the system by the update loop would be attenuated by 40 dB. $\zeta$ was chosen as 0.5 to provide fast transient response (using a critical damping factor, $\zeta = 1$, gave sluggish response). The fixed gain comparison is really a measure of how closely the system model matches the actual system—we would expect that the most complicated model would yield the best controller.

### 4.1.1 Joint control

The joint control algorithm proved to be very sensitive to the radius of the object being grasped. Since the inverse kinematic solution requires knowledge of the object radius, errors in this radius cause the desired joint position to be wrong. If the modeled radius is too small we get a constant joint error and this joint error results in additional internal force and object position errors (due to nonlinearity of the hand kinematic map). Likewise, if the modeled object radius is too large then the internal force term causes a constant error in the joint positions. In fact, if the controller gains are large enough the PD control law can override the constant internal grasping force and cause contact to be broken. The constant displacement seen in figure 3 is due to unintentionally setting the object radius slightly too large (this same radius was used by all of the algorithms).

Figure 3 also shows that the orientation error for the joint control algorithm is very sensitive. Because a small change in orientation produces a very small change in the joint position (as compared to an error in object position) the joint control algorithm is not very effective at controlling the orientation. Increasing the gain in the joints (which requires an increase in controller rate) can help overcome this problem.

### 4.1.2 Force transformation

For Styx, the mass of the fingers is much heavier than the mass of the object and so we do not expect an algorithm which ignores the finger dynamics to perform well. Figure 4 confirms our intuition. Commanding torques which are sufficient to move only the object produces large errors due to the mass of the fingers.

### 4.1.3 Computed torque

The performance of the computed torque algorithm is the best of any of the algorithms presented in this section (see figure 5). The position error is comparable to that of the joint control algorithm but the orientation error is much lower. Also the computed torque algorithm is insensitive to errors in the object size—the controller is effectively controlling the position of the line connecting the fingertips and simultaneously pushing in along that line. For very small objects, the orientation becomes very sensitive to the fingertip positions and the performance degrades somewhat.

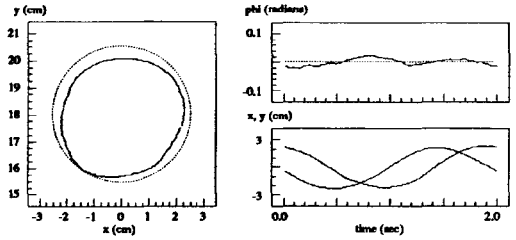

Figure 5: Computed torque algorithm

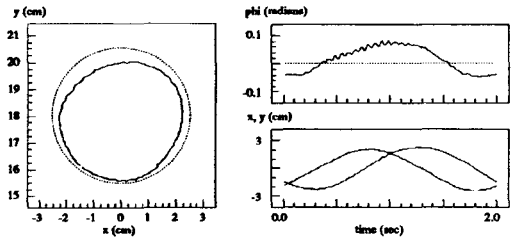

Figure 6: Natural control law - equivalent gains

#### 4.1.4 Natural and stiffness control

The natural and stiffness controllers do not give a linear error equation and hence our design criterion for choosing $K_p$ and $K_v$ does not apply. In fact this is one of the problems with these algorithms--there is no simple method for choosing the gains. The proof of stability gives general conditions for convergence ($K_p > 0$, $K_v > 0$) but does not provide a good indication of the performance to be expected. If we execute the algorithms with the gains used in this section we find that the grasped object moves very slightly (much less than with the force transformation controller shown in figure 4).

### 4.2 Equivalent gain comparisons

In order to compensate for the low gains of the natural and stiffness controllers one might consider using $M_h K_p$ and $M_h K_v$ as gains. This would then give a control law very similar to computed torque and we might expect equivalent performance. Unfortunately, the stability analysis for the natural and stiffness controllers requires that the gain matrices be constant and $M_h$ is a function of finger and object position. Furthermore, $M_h > 0$ and $K_p > 0$ does not imply $M_h K_p > 0$. An approximate solution is available. Since $K_p$ and $K_v$ are diagonal (by choice) we can examine the diagonal entries of $M_h$ at some nominal position and use these entries to scale $K_p$ and $K_v$. In the case of a constant diagonal inertia matrix this would then give us a control law similar to computed torque. We term this type of comparison an *equivalent gain* comparison since it attempts to compensate for differences in the magnitudes of the overall gain matrices.

#### 4.2.1 Natural control

The results of scaling the gain matrices for the natural controller are shown in figure 6. We see that the position error is greatly reduced (before scaling the hand barely moved), but there appear to be oscillations in the orientation (upper right graph). In fact the controller was very marginally stable and any slight disturbance would cause the hand to oscillate. Part of the reason that the controller is so nearly unstable may be the form of the inertia matrix for the hand. For the experiments performed here, the diagonal entries of this matrix were quite large but the inertial coupling between the x position and the orientation was of comparable magnitude.

#### 4.2.2 Stiffness control

The stiffness control law appeared to be slightly more stable (see figure 7). The oscillations in orientation are no longer present although there was little improvement in trajectory error. Results of other tests

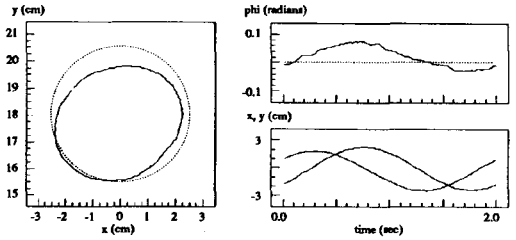

Figure 7: Stiffness control law - equivalent gains

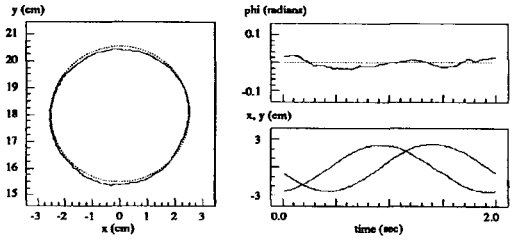

Figure 8: Joint control algorithm (100 Hz, $\omega_n = 5$ Hz)

indicate that the natural controller was in fact more stable but still quite underdamped.

### 4.3 Fixed hardware comparisons

The final comparison is the *fixed hardware* comparison. Here we allow the gains and control rates to be maximized individually. This is the most realistic comparison since it ranks the controllers in overall effectiveness. It is reasonable to imagine that a simple controller might be able to perform better because it can run at a much higher servo rate--thus compensating for uncertainties more quickly. For each controller the following steps where performed:

1. Maximize controller speed – the update and control rates were chosen as large as possible such that both loops could still finish their calculations in the allotted time.
2. Maximize controller gain – the same method as previously outlined was used: $\omega_n$ was chosen as one tenth of the update frequency and $\zeta$ was set to 0.5.

Since the control law was linked with the frequency of the update loop and the control loop was running much faster than the dynamics of the system (effectively emulating a continuous time controller), only the update frequency was varied in the experiments presented here. Due to the structure of the control software, only integer multiples of the control period were used for the update period.

#### 4.3.1 Joint control

The simplicity of the joint control algorithm allows a controller with sufficiently high gain and bandwidth to overcome unmodeled disturbances (see figure 8). Due to other sources of noise in the system the cutoff frequency for this algorithm was placed at 5 Hz instead of 10 Hz. Note the slight shift in the y position due to the aforementioned error in object radius.

#### 4.3.2 Force transformation

We might expect the force transformation algorithm to experience similar improvements. Due to its moderate complexity, an update rate of 50 Hz was the maximum achievable rate (up from 25 Hz). Figure 9 shows that the overall performance was still very poor. This is to be expected since the formulation completely ignored the finger dynamics, which were much larger than the object dynamics.

#### 4.3.3 Computed torque

Computing the full set of nonlinear compensating terms allows the computed torque algorithm to be run with an update rate of only 25 Hz

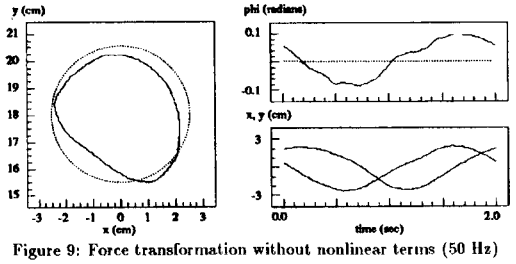

Figure 9: Force transformation without nonlinear terms (50 Hz)

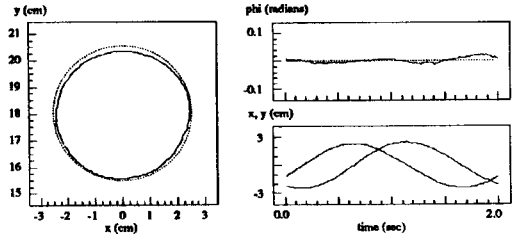

Figure 10: Computed torque without nonlinear terms (33 Hz)

(shown in figure 5). This limits the frequency response of the controller as well as the DC gain (which is determined by $K_p = \omega_n^2$). A considerable savings in computational complexity can be gained by ignoring the nonlinear terms. Calculations indicate that these terms are small relative to the link/motor inertias. By ignoring these nonlinear terms we can increase $\omega_n$ to 3.3 Hz and we see that the controller is able to follow the desired trajectory much more closely (figure 10). Notice how small the orientation error is compared to other algorithms.

#### 4.3.4 Natural and stiffness control

The natural and stiffness controllers were of sufficient complexity that it was not possible to increase the update rate past 25 Hz, even by ignoring the nonlinear terms. Therefore the equivalent gain comparisons shown in figures 6 and 7 where the best results obtained.

### 4.4 Conclusions

Based on the experiments performed on Styx, the most effective control laws are the simple joint control law and the generalized computed torque control law. Although the joint control law ignores the interaction between the joints (caused by grasping the object), it can be run at sufficiently high rates (and hence gains) to overcome errors. It has the additional advantage that it is stable even when contact is broken, since the joint controllers are individually stable. The computed torque control law, being a coordinated control law, relies on the fact that contact is not broken. While intermittent breaks don't usually present a problem, the controller does exhibit strange behavior when the object is removed from the fingers' grasp—the fingers move in to the center of the line connecting the fingertips, where the orientation gain becomes very large. Undesirable oscillations result.

The disadvantage of the joint control law is that the error dynamics are not linear and therefore it is difficult to predict the results. If the object being grasped has a large inertia relative to the fingers, the joint controller will experience problems like those of the force transformation controller, resulting in large errors. The performance of the generalized computed torque algorithm will be basically unchanged, since all dynamics were considered in deriving the algorithm and the resulting error system is linear. This makes the computed torque control law an attractive alternative for position control of multi-fingered hands.

#### Acknowledgements

The authors would like to thank the members of the robotics lab at U.C. Berkeley, and in particular Ping Hsu, John Hauser, Zexiang Li and Arlene Cole, for many useful suggestions and comments on this research.

## References

[1] S. Arimoto. Cooperative motion control of multi-robot arms or fingers. In *IEEE RA Conference*, pages 1407–1412, 1987.

[2] A. K. Bejczy. *Robot Arm Dynamics and Control.* Technical Report 33-699, Jet Propulsion Laboratory, 1974.

[3] B. Chen, R. Fearing, B. Armstrong, and J. Burdick. Nymph: a multiprocessor for manipulation applications. In *IEEE RA Conference*, pages 1731–1736, 1986.

[4] A. Cole, J. Hauser, and S. Sastry. Kinematics and control of multifingered hands with rolling contact. In *IEEE RA Conference*, pages 228–233, 1988.

[5] J. Demmel, G. Lafferrier, J. Schwartz, and M. Sharir. Theortical and experimental studies using a multifinger planar manipulator. In *IEEE RA Conference*, pages 390–395, 1988.

[6] S. Hayati. Hybrid postion/force control of multi-arm cooperating robots. In *IEEE RA Conference*, pages 82–89, 1986.

[7] P. Hsu. *Control of Mechanical Manipulators.* PhD thesis, Department of Electrical Engineering, University of California, Berkeley, California, 1988.

[8] S. Jacobsen, J. Wood, K. Bigger, and E. Iverson. The Utah/MIT hand: work in progress. *IJRR*, 4(3):221–250, 1986.

[9] J. Kerr. *An Analysis of Multi-fingered Hands.* PhD thesis, Stanford University, Department of Mechanical Engineering, 1984.

[10] J. Kerr and B. Roth. Analysis of multifingered hands. *IJRR*, 4(4):3–17, Winter 1986.

[11] D. Koditschek. Natural motion for robot arms. In *IEEE CDC*, pages 733–735, 1984.

[12] Z. Li, P. Hsu, and S. Sastry. On kinematics and control of multifingered hands. In *IEEE RA Conference*, pages 384–389, 1988.

[13] J. Y. S. Luh, M. W. Walker, and R. P. Paul. Resolved acceleration control of mechanical manipulators. *IEEE AC Transactions*, AC-25, 1980.

[14] R. Murray. *Experimental Results in Planar Grasping.* Master's thesis, University of California, Berkeley, 1988.

[15] Y. Nakamura, K. Nagai, and T. Yoshikawa. Mechanics of coordinative manipulation of multiple robot mechanisms. In *IEEE RA Conference*, pages 991–998, 1987.

[16] V. Nguyen. *The Synthesis of Stable Force-Closure Grasp.* Master's thesis, Massachusetts Institute of Technology, 1986.

[17] R. Ortega and M. W. Spong. *Adaptive Motion Control of Rigid Robots: A Tutoral.* Technical Report, Coordinated Sciences Lab., University of Illinois, Urbana, Illinois, 1988.

[18] K. Rao, G. Medioni, H. Liu, and G. A. Bekey. Robot hand-eye coordination: shape description and grasping. In *IEEE RA Conference*, pages 407–411, 1988.

[19] N. Sadegh. *Adaptive Control of Mechanical Manipulators: Stability and Robustness Analysis.* PhD thesis, Department of Mechanical Engineering, University of California, Berkeley, California, 1987.

[20] J. K. Salisbury. *Kinematic and Force Analysis of Articulated Hands.* PhD thesis, Stanford University, Department of Mechanical Engineering, 1982.

[21] D. M. Siegel, S. Narasimhan, J. M. Hollerbach, D. Kriegman, and G. Gerpheide. Computational architecture for the Utah/MIT hand. In *IEEE RA Conference*, pages 1884–1889, 1986.

[22] J. E. Slotine and W. Li. On the adaptive control of robot manipulators. *IJRR*, 6:49–59, 1987.

[23] T. Tarn, A. Bejczy, and X. Yuan. Control of two coordinated robots. In *IEEE RA Conference*, pages 1193–1202, 1986.

[24] S. T. Venkataraman and T. E. Djaferis. Multivariable feedback control of the JPL/Stanford hand. In *IEEE RA Conference*, pages 77–82, 1987.

[25] J. T. Wen and D. S. Bayard. New class of control laws for robot manipulators. part 1: non-adaptive case. *International Journal of Control*, 47(5):1361–1385, 1988.

[26] Y.F. Zheng and J.Y.S. Luh. Control of two coordiantes robots in motion. In *IEEE CDC*, pages 1761–1766, 1985.