Applied Joint-Space Torque and Stiffness Control of Tendon-Driven Fingers

Muhammad E. Abdallah, Robert Platt Jr., Charles W. Wampler, Brian Hargrave

***Abstract*— Existing tendon-driven fingers have applied force control through independent tension controllers on each tendon, i.e. in the tendon-space. The coupled kinematics of the tendons, however, cause such controllers to exhibit a transient coupling in their response. This problem can be resolved by alternatively framing the controllers in the joint-space of the manipulator. This work presents a joint-space torque control law that demonstrates both a decoupled and significantly faster response than an equivalent tendon-space formulation. The law also demonstrates greater speed and robustness than comparable PI controllers. In addition, a tension distribution algorithm is presented here to allocate forces from the joints to the tendons. It allocates the tensions so that they satisfy both an upper and lower bound, and it does so without requiring linear programming or open-ended iterations. The control law and tension distribution algorithm are implemented on the robotic hand of Robonaut-2.**

I. Introduction

Tendon transmission systems are often used in the actuation of fingers for high degree-of-freedom (DOF) hands. The remote actuation allows for significant reductions to the size and weight of the fingers, features that are important for dexterous manipulation. Since the tendons can only transmit forces in tension, the number of actuators must exceed the DOF's to achieve fully determined control of the finger. It turns out that only *one* tendon more than the number of DOF's is needed [1]. If arranged correctly, the $n+1$ tendons can independently control the n DOF's while always maintaining positive tensions. Such an "$n+1$" arrangement is very attractive due to its minimum number of actuators. Each extra actuator and transmission system greatly increase the demands on space, power, and maintenance for the system.

On the other hand, an "$n+1$" arrangement also introduces a layer of complexity to the control. This complexity arises from the coupled relationship between the tendon and joint displacements. Traditionally, such fingers have been controlled using what can be referred to as *tendon-space controllers*. Under such schemes, desired joint torques are translated into desired tendon tensions, then each tension is controlled by an independent controller. Salisbury and Craig were the first to implement such a scheme using the Stanford/JPL hand [2]. Similar schemes where implemented on the CT-ARM manipulator and POSTECH hand, although they both used $2n$ tendons [3], [4].

Platt, *et al.* have shown, however, that tendon-space controllers introduce a transient coupling to the dynamics of the finger [5]. This disturbance arises intrinsically from the kinematics of the tendons, rather than from bandwidth limitations in the controllers. This problem is solved by implementing *joint-space controllers* that regulate the reference torques in the joint-space and thus decouple the tendon effects.

The ability to control torques is important in assembly applications, where fingers physically interact with unstructured environments. The torque control can then be implemented in either a stiffness or impedance framework, to define either the static or dynamic interaction of the finger. This work presents a stiffness controller for tendon-driven manipulators. The controller is implemented in both a tendon-space and joint-space formulation.

The existing systems all implemented their torque control based on proportional-integral (PI) controllers, be they in the tendon-space [2], [4] or in the joint-space [5]. The joint-space controller of [5] struggled when conduits where added to the tendon drive-train. The conduits add complex dynamics and hysteresis due to their distributed stiction. The controller presented here proved to be both faster and more robust to the unmodeled conduit dynamics than the comparable PI controller.

In addition, this work presents a tension distribution algorithm for allocating the desired forces from the joints to

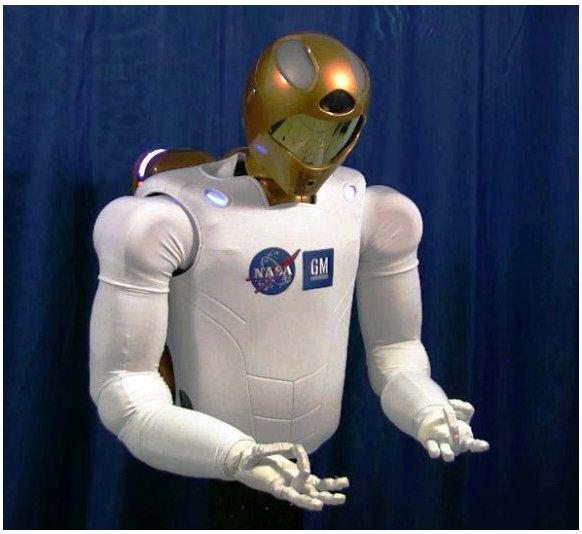

Fig. 1. The torque controlled fingers of Robonaut-2.

M. Abdallah and C. Wampler are with the Manufacturing Systems Research Lab, General Motors R&D, Warren, MI 48090, USA {muhammad.abdallah, charles.w.wampler}@gm.com

R. Platt was with the Johnson Space Center, NASA, Houston, TX 77058, USA robert.platt-1@nasa.gov

B. Hargrave is with Oceaneering Space Systems, Houston, TX 77058, USA bhargrave@oceaneering.com

Patents are pending on this work.

the tendons. This step is necessary in all torque controllers, where it must satisfy the fundamental requirement of positive tensions. The algorithm presented here distributes the tensions so that they satisfy both a lower and upper bound. It satisfies the lower bound while minimizing the internal tension applied. At the same time, it satisfies the upper bound while eliminating the coupled effects of saturation. The algorithm is computationally efficient, requiring no linear programming or open iterative solutions.

A few alternatives exist in the literature. Salisbury and Craig present a method for setting the lowest tension equal to zero, and we build off their method [2]. Jacobsen, *et. al* present a computationally efficient analytical solution for antagonistic pairs [6], which is then extended to multiple DOF systems by Lee and Tsai [7]. This solution, however, only provides for positive tensions and is heavily system dependent, requiring considerable analytical derivation for each system. Hirose and Ma present an alogrithm that changes the joint torques and does not necessarily minimize the internal tension [8].

Both the joint-space controller and the tension distribution algorithm are currently implemented on the humanoid hand of the GM-NASA Robonaut-2 robot. These topics are presented over three sections. The first section presents the tension distribution algorithm. The second section presents the control law in both formulations. Finally, the third section presents the experimental results from Robonaut-2.

II. Tension Distribution Algorithm

A. Problem

In the torque control of tendon-driven fingers, the desired joint torques must first be translated into tendon tensions. This problem is referred to as *tension distribution*, and it must ensure that each tension value is non-negative. Alternatively, the step can be framed as a problem of solving for the necessary internal tension on the finger. The distribution can essentially be solved using a linear programming technique—which is undesirable due to the complexity and open iterative nature of the solution.

We present a solution that ensures that each tension falls within the bounded range $[f_{min}, f_{max}]$, where $f_{min} \geq 0$. It sets the lowest tension equal to f_{min} and thus minimizes the internal tension. Whenever the highest tension exceeds f_{max}, it solves for the linear scaling of the torques needed to satisfy the bounds while minimizing the internal tension.

Fundamental to the problem is the relationship between the n joint torques, $\boldsymbol{\tau}$, and the $n+1$ tendon tensions, $\boldsymbol{f}$. The transformation from tensions to torques is:

$$\begin{pmatrix} \boldsymbol{\tau} \\ t \end{pmatrix} = P\boldsymbol{f}, \qquad P = \begin{bmatrix} R \\ W \end{bmatrix}, \tag{1}$$

where t is defined as the internal tension. $R \in \mathbb{R}^{n \times n+1}$ is known as the tendon map; it contains the joint radii data mapping tendon tensions to joint torques. W is an $n+1$ row matrix that does not lie in the range space of R^T. For the system to be tendon controllable, R must be full row rank and have an all-positive null-space [9]. P is thus a nonsingular matrix. Throughout this work, bold symbols represent column matrices.

The solution for the tensions can be found from the inverse of (1). That inverse can be partitioned as follows:

$$\boldsymbol{f} = P^{-1}\begin{pmatrix} \boldsymbol{\tau} \\ t \end{pmatrix}, \qquad P^{-1} = [A \ \boldsymbol{a}], \tag{2}$$

where $A \in \mathbb{R}^{n+1 \times n}$ and $\boldsymbol{a} \in \mathbb{R}^{n+1 \times 1}$. We will select W to be orthogonal to R; that is, $RW^T = 0$. W thus spans the null-space of R and, by assumption, is all-positive. Under this orthogonality condition:

$$A = R^+, \quad \boldsymbol{a} = W^+, \tag{3}$$

where the superscript ($^+$) indicates the pseudoinverse. Note that $\boldsymbol{a}$ is all-positive, since the pseudoinverse of a positive vector is also positive. Assuming constant joint radii, A and $\boldsymbol{a}$ are constant matrices that can be precomputed.

B. Solution

The first step in the algorithm is to distribute the tensions so that the minimum value equals f_{min}. Let A_i represent the rows of A and a_i the elements of $\boldsymbol{a}$. We require that,

$$f_i = A_i\boldsymbol{\tau} + a_i t \geq f_{min}. \tag{4}$$

This entails the following solution for the internal tension, presented in [2]. Recall that $a_i > 0$.

$$t_0 = \max_i \frac{f_{min} - A_i\boldsymbol{\tau}}{a_i} \tag{5}$$

By substituting this internal tension value (t_0) into (2), we can obtain the tension distribution. We refer to this distribution as the *initial solution.*

$$\boldsymbol{f} = [A \ \boldsymbol{a}]\begin{pmatrix} \boldsymbol{\tau} \\ t_0 \end{pmatrix}. \tag{6}$$

Now, we want to take this algorithm a step further and set an upper bound for the tension values. The first step is to check if any of the tensions exceed the upper bound, f_{max}. Let index l represent the element with the lowest tension and h represent the element with the highest tension. If $f_h > f_{max}$, we will linearly scale the torques such that:

$$\boldsymbol{f} = [A \ \boldsymbol{a}]\begin{pmatrix} \alpha\boldsymbol{\tau} \\ t \end{pmatrix}, \tag{7}$$

where α is a positive scalar. Now, we will find the explicit solution where $f_l = f_{min}$ and $f_h = f_{max}$. The result follows, which we refer to as the *scaled solution.*

$$\begin{aligned} d &= (a_h A_l - a_l A_h)\boldsymbol{\tau} \\ \alpha &= \frac{a_h f_{min} - a_l f_{max}}{d} \\ t &= \frac{f_{max} A_l - f_{min} A_h}{d}\boldsymbol{\tau} \end{aligned} \tag{8}$$

As we will show in the next subsection, this solution guarantees that $\boldsymbol{f} \in [f_{min}, f_{max}]$ under two conditions: when

$f_{min} = 0$ or the finger design has a *balanced configuration*. A finger with a balanced configuration exhibits no net torques when the tensions are all equal. Otherwise, the solution does not guarantee that all elements lie within the desired limits. In the case that an element does exceed the limits, the scaled solution (8) needs to be iterated after reassigning the index l or h, respectively, to the new extreme element.

Having an open-ended iterative solution is undesirable in a high bandwidth, real-time application. We show here, however, that the need to iterate is rare due to the nature of the tendon transformation. For a typical design, it can occur for less than 2% of the commanded torque values. Not only that, but the first iteration is sufficient for solving the problem. Hence, instead of an open-ended iterative problem, the algorithm can be capped at one iteration. Of course, no iterations at all are needed if either $f_{min} = 0$ or the configuration is balanced. The next subsection discusses these claims.

The advantage of this algorithm lies in two key points. First, it distributes the tendons with a computationally efficient algorithm that does not need linear programming. Second, it caps the maximum tension with a linear scaling of the desired joint torques. This feature protects the tendons from being overloaded either by the controller or by the environment. It also allows the system to avoid mechanically saturating the tensions, which would introduce a coupled disturbance to the joint torques.

Here is a summary of the *tension distribution algorithm*:

1) Find the initial solution using (5) and (6).
2) Assign index h to the element with the highest tension and l to the element with the lowest tension.
3) If $f_h < f_{max}$, exit. Else, find the first scaled solution using (8) and (7).
4) Assign index h to the element with the highest tension and l to the element with the lowest tension.
5) If $f_h < f_{max}$ and $f_l > f_{min}$, exit. Else, find the second and final scaled solution using (8) and (7).

C. Analysis

The previous subsection claims that the scaled solution rarely pushes another element beyond the bounds. It also claims that no iterations are needed when $f_{min} = 0$ or the finger has a balanced configuration. This subsection presents an analysis of these claims.

First, consider the condition for the existence of a solution in (8). Derived from the limiting case of $\boldsymbol{f} = \boldsymbol{a}t_o$, a solution will exist if and only if:

$$f_{max} \geq \max_i \frac{a_i}{a_l} f_{min}. \tag{9}$$

This condition should be readily satisfied in typical finger implementations. Given this condition, it can be shown that α lies in the range $(0, 1)$.

Second, consider the relationship between the scaled and initial solutions. Whenever the scaled solution maintains the relative order of the elements, no iteration will be needed. Let ${}^0\boldsymbol{f}$ refer to the initial solution while ${}^1\boldsymbol{f}$ refer to the first scaled solution. It can be shown that the two relate as follows.

$${}^1\boldsymbol{f} = \alpha\ {}^0\boldsymbol{f} + \tfrac{(1-\alpha)f_{min}}{a_l}\boldsymbol{a} \tag{10}$$

The first term on the right-hand side represents the linearly scaled portion of the result; it thus maintains the order of the elements. The second term, however, represents a change in the linear distribution. Hence, when $f_{min} = 0$, the term drops out and the scaled solution fully maintains the relative magnitudes of the elements. This guarantees that the bounds are satisfied after the first scaled solution.

Alternatively, the linear distribution is also maintained if the elements of $\boldsymbol{a}$ are all equal. We refer to this condition as a *balanced configuration*. The condition occurs when the columns of R, $\boldsymbol{r}_i$, sum to zero.

$$\sum_{i=1}^{n+1} \boldsymbol{r}_i = \boldsymbol{0} \tag{11}$$

Hence, a vector $\boldsymbol{f}$ of equal tensions will lie in the null-space of R. Intuitively, it implies that the joint radii are so balanced as to produce no net torques when the tensions are all equal.

When neither of these two conditions is true, the relative order of the elements can change and a different element can jump the limit. In a typical finger design, however, the relative difference between the elements of $\boldsymbol{a}$ will be small and the jump will rarely occur. Fingers designed to manipulate in both directions will not diverge far from the balanced condition; otherwise, the force control will be heavily biased in one direction. In a numerical study of a representative finger, it was observed that a third element rarely exceeds the bounds after the first scaled solution, and that the algorithm can be capped at one iteration. That study is presented in the next subsection.

D. Computational Results

The algorithm was tested in a Matlab simulation modeling the Robonaut index finger. The finger has three joints and four tendons ($R \in \mathbb{R}^{4\times3}$). The actuators were designed for a maximum of 50 lb tension, producing a maximum joint torque of 26 in-lbs on the finger. The tendon map for the finger follows.

$$P = \begin{bmatrix} 0.15 & 0.15 & -0.15 & -0.15 \\ 0.265 & -0.195 & 0.265 & -0.195 \\ 0 & 0 & 0.195 & -0.195 \\ 0.195 & 0.367 & 0.281 & 0.281 \end{bmatrix} \tag{12}$$

The algorithm limits were set to $f_{min} = 2$ lbs and $f_{max} = 40$ lbs. The value of f_{min} was selected conservatively; lower, more typical values reduce the chance of needing iterations. According to condition (9), a solution exists given $f_{max} \geq 4$.

The simulation tested all torque combinations spanning a range of $\tau_i \in [-50, 100]$ in-lbs, with a 1 lb resolution. After the first scaled solution, an element of f exceeded the limits on less than 2% of the torque values. Even in those cases, the high and low values remained close to the limits. Out of the 3 million test points tried, the absolute maximum tension value was 41.1 lbs, and the absolute minimum was 1.2 lbs.

Running the same test with one iteration, the limits were exceeded on (statistically) 0% of the values. The absolute extreme values returned were 40.2 and 2.0 lbs respectively. Hence, we can confidently cap the number of iterations in the algorithm at one.

Given this tension distribution algorithm, we can now turn to the actual finger controllers. The next section presents the control law in both a tendon-space and joint-space formulation.

III. Finger Stiffness Controller

The ability to control torques is important for manipulators that interact physically with their environments. The torque controller presented here is formulated as a stiffness controller for the finger joints. It can also be applied to higher-order terms for full impedance control.

The stiffness control commands a torque proportional to the joint error. Given a desired vector of joint values, $\boldsymbol{q}_d$, the vector of desired joint torques follows.

$$\boldsymbol{\tau}_d = K(\boldsymbol{q}_d - \boldsymbol{q}) \tag{13}$$

K is the diagonal stiffness matrix and $\boldsymbol{q}$ is the sensed joint positions.

The next two subsections present the tendon-space and joint-space formulations of the torque controller. In both cases, we need to solve for the desired internal tension, t_d, such that the tension values are all positive. The tension distribution algorithm of section II provides a solution that applies both a lower and upper bound to the tensions.

A. Tendon-Space Controller

We will first develop the tendon-space formulation for the torque controller. Since actuators often already employ a well-tuned position controller, a standard approach to force control uses an inner position loop. The torque control loop thus needs to pass commanded actuator positions, $\boldsymbol{x}_d$, to the lower loop, where it is assumed that a high-gain PD controller exists around the actuator position. The desired tendon tensions, $\boldsymbol{f}_d$, are found from the desired torques through the tendon map matrix as follows.

$$\boldsymbol{f}_d = P^{-1} \begin{pmatrix} \boldsymbol{\tau}_d \\ t_d \end{pmatrix} \tag{14}$$

The tendon-space controller models the tendon as a linear spring, where

$$\boldsymbol{f} = k_s(\boldsymbol{x} - \boldsymbol{x}_o). \tag{15}$$

k_s is the spring constant, $\boldsymbol{x}$ is the current position of the actuator, and $\boldsymbol{x}_o$ is the unstretched position. This relation assumes that the spring constant is effectively equal for all the tendons. This is a valid assumption, since the tendon lengths are relatively equal. For a desired tension, the relation becomes,

$$\boldsymbol{f}_d = k_s(\boldsymbol{x}_d - \boldsymbol{x}_o). \tag{16}$$

Subtracting (15) and (16) results in the following relation for the desired actuator position.

$$\boldsymbol{x}_d = \boldsymbol{x} + \frac{1}{k_s}(\boldsymbol{f}_d - \boldsymbol{f}) \tag{17}$$

This relation inspires the following tendon-space control law. It feeds forward the current actuator position and adds damping to increase the stability of the controller.

$$\boldsymbol{x}_d = \boldsymbol{x} - k_d\dot{\boldsymbol{x}} + k_p(\boldsymbol{f}_d - \boldsymbol{f}) \tag{18}$$

k_p and k_d are the constant, scalar gains. Since $\boldsymbol{x}$ is both proportional to $\boldsymbol{f}$ and has a less noisy signal, $\dot{\boldsymbol{x}}$ is employed for the damping term instead of $\dot{\boldsymbol{f}}$.

A key advantage of this control law is that it does not employ an integrator as in [2], [4]. The feed forward term makes the controller faster, avoiding the lag and wind-up problems associated with the integrator; however, it also results in non-zero steady-state error.

B. Joint-Space Controller

The tendon-space control law (18) does a good job of independently tracking the desired tensions. Its transient behavior, however, displays a coupled response amongst the joints, one that introduces unnecessary motion given either step inputs or disturbance responses. This coupling is a direct product of the control law, rather than the passive dynamics [5]. The following controller seeks to eliminate this coupled transience by operating in the joint-space.

Accordingly, an analogous control law to (18) is formulated in the joint-space. This allows for the independent regulation of the joint torques.

$$\bar{\boldsymbol{q}}_d = \bar{\boldsymbol{q}} - k_d\dot{\bar{\boldsymbol{q}}} + K_p(\bar{\boldsymbol{\tau}}_d - \bar{\boldsymbol{\tau}}) \tag{19}$$

The null-space components are included in this relation, where $\bar{\boldsymbol{q}} = \begin{pmatrix} \boldsymbol{q} \\ \theta \end{pmatrix}$ and $\bar{\boldsymbol{\tau}} = \begin{pmatrix} \boldsymbol{\tau} \\ t \end{pmatrix}$. $\bar{\boldsymbol{\tau}}$ is computed from the sensed tensions through (1). K_p is the proportional gain matrix; it is diagonal but no longer scalar since the internal tension needs a different gain than the joint torques. k_d, on the other hand, is intentionally left scalar.

To convert the joint velocities to tendon velocities, a standard virtual work analysis provides the following dual transformation [10]. This relation assumes negligible friction in the joint pulleys as well as constant external torque.

$$\dot{\boldsymbol{x}} = P^T \begin{pmatrix} \dot{\boldsymbol{q}} \\ \dot{\theta} \end{pmatrix} \tag{20}$$

$\dot{\theta}$ is defined as the internal velocity. Assuming equal tendon stiffnesses, $\dot{\theta}$ parameterizes the space of tendon velocities that apply no change to the joint torques. Defining the initial positions as zeros leads to the following relation in the position domain.

$$\boldsymbol{x} = P^T \begin{pmatrix} \boldsymbol{q} \\ \theta \end{pmatrix} \tag{21}$$

Given this transformation, we can now derive our final joint-space controller.

$$\begin{aligned} \boldsymbol{x}_d &= P^T\bar{\boldsymbol{q}}_d \\ &= P^T\left[\bar{\boldsymbol{q}} - k_d\dot{\bar{\boldsymbol{q}}} + K_p(\bar{\boldsymbol{\tau}}_d - \bar{\boldsymbol{\tau}})\right] \\ &= \boldsymbol{x} - k_d\dot{\boldsymbol{x}} + P^T K_p\left(\bar{\boldsymbol{\tau}}_d - \bar{\boldsymbol{\tau}}\right) \end{aligned} \tag{22}$$

Keeping k_d scalar allows us to translate the damping term to the tendon space. The collocation achieved by using the

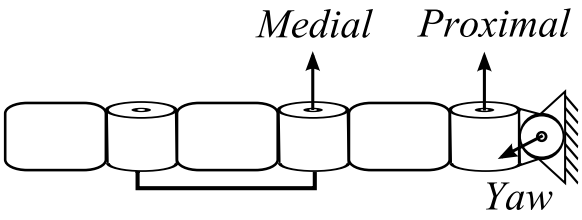

Fig. 2. A model of the Robonaut-2 index finger. Motion of the distal joint is mechanically linked to the medial joint.

actuator sensing, $\boldsymbol{x}$ and $\dot{\boldsymbol{x}}$, instead of the joint sensing, $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$, increases the stability of the system. Otherwise, the feed-forward and damping terms would lag the actuation by the unmodeled dynamics of the tendon-conduit transmission.

C. Discussion

The tendon-space control law (18) can be rearranged as follows.

$$\boldsymbol{x}_d = \boldsymbol{x} - k_d\dot{\boldsymbol{x}} + P^{-1}k_p\left(\bar{\boldsymbol{\tau}}_d - \bar{\boldsymbol{\tau}}\right) \tag{23}$$

Note the similarity between (22) and (23). The essential difference between the two control laws is in the use of P^T versus P^{-1}. Replacing the inverse with the transpose provides the decoupling of the joint-space motion. This is analogous to the duality associated with the Jacobian in the Cartesian control of serial manipulators. Consider the two control laws: $\Delta\boldsymbol{q} = kJ^{-1}\Delta\boldsymbol{y}$, and $\Delta\boldsymbol{q} = kJ^T\Delta\boldsymbol{y}$. In this expression: J is the end-effector Jacobian, $\boldsymbol{y}$ is the end-effector position, $\boldsymbol{q}$ is the joint angles, and k is the gain. The first law produces straight line motion in Cartesian space, while the second produces coupled Cartesian motion.

In [5], PI torque regulators were implemented on a finger having no conduits about the tendons. When conduits were introduced, that approach struggled due to the distributed stiction between the tendons and conduits, which add complex dynamics or hysteresis to the system. We ended up not needing to model the hysteresis thanks to the combination of the feed-forward term and the sensor-actuator collocation. This combination allowed our controller to produce both a more stable and faster response compared to the customary PI controller.

IV. Experimental Validation

A. Mechanical System

The two control laws were tested on an index finger of the Robonaut-2 robot, a model of which is shown in Fig. 2. The finger has four tendons and three independent DOF's: a yaw, a proximal pitch, and a medial pitch. The yaw joint is perpendicular to both pitch joints, and the tendon mapping matrix is shown in (12).

The system is actuated by brushless DC motors with a planetary reduction gearhead. A ball-screw provides the linear conversion for the motor power, which is then transmitted to the finger through a tendon-conduit arrangement. Feedback on the tendon tensions are provided through strain-gauges lying in the path of the transmission.

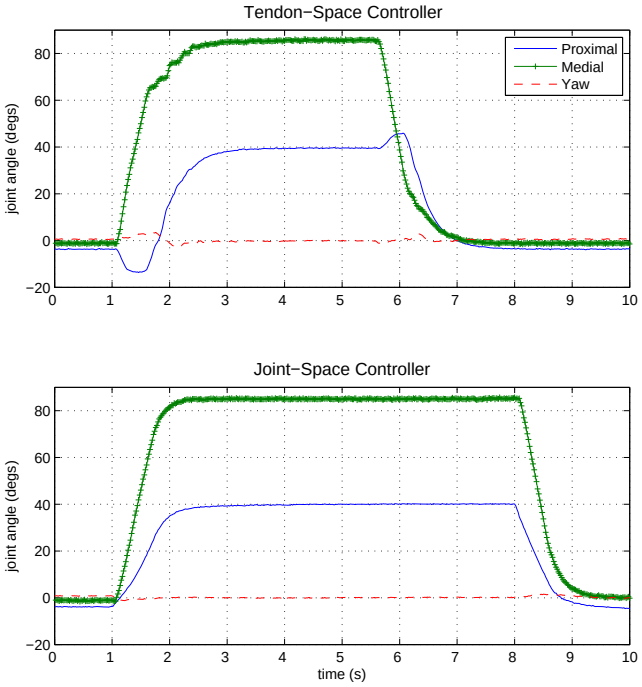

Fig. 3. Comparison of the two finger controllers. A positive then negative step input of {0, 45, 90} degs was commanded. Note both the decoupled and faster response of the joint-space controller.

B. Step Response

The step response of each controller was tested using a step input of $\boldsymbol{q}_d = (0, 45, 90)^T$ degs. The results are shown in Fig. 3. The tendon-space controller exhibited the aforementioned transient coupling, coupling that disturbed the tension tracking, delayed the response time, and produced unsightly motion in the finger. This behavior was consistently demonstrated throughout our tests; it surfaced also in the response to external disturbances.

The joint-space controller, on the other hand, eliminated the coupling and significantly increased the speed of the response. For both moving joints, the joint-space controller reduced the settling time by almost 25%. The medial joint dropped from a settling time of 1.3 s to 0.97 s. The proximal joint dropped from a settling time of 2.2 s to 1.7 s.

The controller parameters are shown in Table I. In each case, the gains were tuned to maximize performance. The lower-loop position controller implemented the same gains for both runs, which were tuned to produce a critically-damped response. As seen in the table, a higher stiffness value was applied to the yaw joint in both cases. The yaw joint is relatively ill conditioned due to significantly smaller radii; hence, the higher stiffness better controlled its motion. The steady-state errors seen in the results can be addressed with a limited-range integrator.

C. Tension & Torque Regulation

A second experiment demonstrated the performance of the torque regulation and tension distribution. In this experiment,

	tendon-space	joint-space
k_d	0.01	0.01
k_p, K_p	0.01	$\begin{bmatrix} 0.05 & 0 & 0 & 0 \\ 0 & 0.05 & 0 & 0 \\ 0 & 0 & 0.05 & 0 \\ 0 & 0 & 0 & 0.003 \end{bmatrix}$
K	$\begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 0.3 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$

TABLE I

CONTROLLER PARAMETERS USED IN THE EXPERIMENTAL RUN.

the finger was operated in torque control mode. Instead of the stiffness input in (13), a reference torque is directly commanded. With the finger sitting against a hard surface, the reference torque for the proximal joint ramped up from 0 to 3.1 in-lb in regular increments. The minimum and maximum tension were set at 1 and 8 lb, respectively. At about 2.3 in-lb, the tensions saturated and the scaled solution kicked in for the distribution algorithm.

Results of the experiment are shown in Fig. 4. The first figure shows the tension distribution satisfying the upper and lower limits. Ideally, the lowest tension should always equal 1 lb, while the maximum tension should equal 8 lbs during saturation. Deviations from this ideal are due to errors in the torque regulation and sensor calibration. The second figure shows the torque regulation, where the displayed torques are computed from the tension feedback using (1). While the medial and yaw torques should remain at zero, the proximal should follow the reference torque until saturation. At that point, it should follow the scaled solution. The experiment demonstrates how saturation is achieved without joint coupling.

V. CONCLUSIONS

For tendon-actuated manipulators, joint-space control offers clear advantages over tendon-space control. Not only does it eliminate the coupled transience exhibited by tendon-space formulations, it also significantly increases the speed of the response. The final form of the control law (22) reveals how simple the transformation from tendon-space to joint-space is. Simply switching from P^{-1} to P^T achieves the desired decoupling. Surprisingly, the same features that helped reduce the joint-space control law to such a simple form also enhanced the performance. These features resulted in a final controller that demonstrated both greater speed and stability than the typical PI controllers. This fact was critical for the implementation of a conduit sheathed drive-train.

The problem of tension distribution, faced by all torque controllers, is essentially a linear programming problem. Using knowledge of the system, we are able to reduce the problem to a tractable algorithm—one conducive to real-time implementation. The upper bound in the algorithm is needed to protect the tendons and to maintain predictable, decoupled torques. The non-zero lower bound allows the controller to either accommodate calibration errors in the tension sensors, or to reduce the internal tension on the finger.

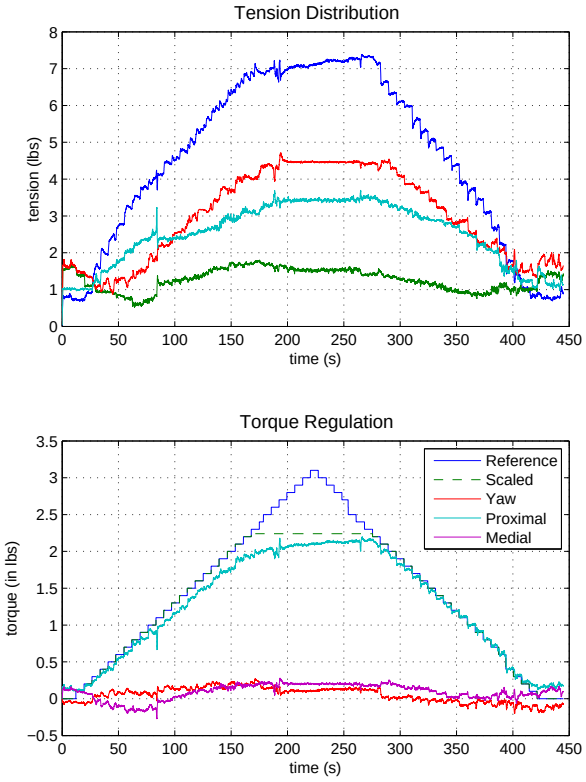

Fig. 4. With the finger in torque control mode, the reference proximal torque is given a triangle input from 0 to 3 in-lb. The controller kept the tensions bounded by the range [1 lb, 8 lb], allowing the tensions to saturate without coupling the joint torques.

REFERENCES

[1] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC Press, 1994.

[2] J. Salisbury and J. Craig, "Articulated hands: Force control and kinematic issues," *International Journal of Robotics Research*, vol. 1, no. 1, pp. 4–17, 1982.

[3] S. Ma, S. Hirose, and H. Yoshinada, "Design and experiments for a coupled tendon-driven manipulator," *IEEE Control Systems Magazine*, vol. 13, no. 1, pp. 30–36, 1993.

[4] Y. Lee, H. Choi, W. Chung, and Y. Youm, "Stiffness control of a coupled tendon-driven hand," *IEEE Control Systems Magazine*, vol. 14, no. 5, pp. 10–19, 1994.

[5] R. Platt, M. E. Abdallah, C. W. Wampler, and B. Hargrave, "Joint-space torque and stiffness control of tendon-driven manipulators," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS) (in submission to)*, Taipei, Taiwan, October 2010.

[6] S. Jacobsen, J. Wood, D. Knutti, and K. Biggers, "The Utah/MIT hand: Work in progress," *Intl. Journal of Robotic Research*, vol. 3, no. 4, pp. 21–50, 1984.

[7] J. Lee and L. Tsai, "Torque resolver design for tendon-driven manipulators," *ASME Journal of Mechanical Design*, vol. 115, pp. 877–883, 1993.

[8] S. Hirose and S. Ma, "Coupled tendon-driven multijoint manipulator," in *IEEE Intl Conf on Robotics and Automation (ICRA)*, Sacramento, April 1991, pp. 1268–1275.

[9] H. Kobayashi, K. Hyodo, and D. Ogane, "On tendon-driven robotic mechanisms with redundant tendons," *International Journal of Robotics Research*, vol. 17, no. 5, pp. 561–571, May 1998.

[10] M. T. Mason and J. K. Salisbury, *Robot Hands and the Mechanics of Manipulation*, 2nd ed. Cambridge, MA: MIT Press, 1982.