8-2007

Control Techniques for Robot Manipulator Systems with Modeling Uncertainties

David Braganza
Clemson University, dbragan@clemson.edu

CONTROL TECHNIQUES FOR ROBOT MANIPULATOR SYSTEMS WITH MODELING UNCERTAINTIES

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Electrical and Computer Engineering

---

by
David Braganza
August 2007

Accepted by:
Dr. Darren M. Dawson, Committee Chair
Dr. Ian D. Walker
Dr. John R. Wagner
Dr. Timothy C. Burg

ABSTRACT

This dissertation describes the design and implementation of various nonlinear control strategies for robot manipulators whose dynamic or kinematic models are uncertain. Chapter 2 describes the development of an adaptive task-space tracking controller for robot manipulators with uncertainty in the kinematic and dynamic models. The controller is developed based on the unit quaternion representation so that singularities associated with the otherwise commonly used three parameter representations are avoided. Experimental results for a planar application of the Barrett whole arm manipulator (WAM) are provided to illustrate the performance of the developed adaptive controller.

The controller developed in Chapter 2 requires the assumption that the manipulator models are linearly parameterizable. However there might be scenarios where the structure of the manipulator dynamic model itself is unknown due to difficulty in modeling. One such example is the continuum or hyper-redundant robot manipulator. These manipulators do not have rigid joints, hence, they are difficult to model and this leads to significant challenges in developing high-performance control algorithms. In Chapter 3, a joint level controller for continuum robots is described which utilizes a neural network feedforward component to compensate for dynamic uncertainties. Experimental results are provided to illustrate that the addition of the neural network feedforward component to the controller provides improved tracking performance.

While Chapter's 2 and 3 described two different joint controllers for robot manipulators, in Chapter 4 a controller is developed for the specific task of whole arm grasping using a kinematically redundant robot manipulator. The whole arm grasping control problem is broken down into two steps; first, a kinematic level path planner is designed which facilitates the encoding of both the end-effector position as well as the manipulators self-motion positioning information as a desired trajectory for the manipulator joints. Then, the controller described in Chapter 3, which provides

asymptotic tracking of the encoded desired joint trajectory in the presence of dynamic uncertainties is utilized. Experimental results using the Barrett Whole Arm Manipulator are presented to demonstrate the validity of the approach.

DEDICATION

This dissertation is dedicated to my parents and sisters for the love and support which has made this work possible.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Darren Dawson for his constant guidance and encouragement during my dissertation work. I appreciate the opportunity he gave me to work on a number of different projects which has made my research experience at Clemson all the more worthwhile. His professionalism and drive to succeed have truly motivated me to strive for excellence in all my work.

I would also like to thank Dr. Ian Walker for the opportunity to work on the OCTOR project which enabled me to complete part of the research in this dissertation. I have worked with him quite often in the past couple of years and his passion for robotics and his elegant explanations of complex topics have made learning from him very enjoyable.

This dissertation would not have been possible without the nurturing and support of my parents and sisters, they have kept me from losing sight of my dreams.

My research at CRB has been a great experience and has been helped along by many colleagues. I'd like to thank Michael McIntyre for his patient explanations and help in understanding the paper writing process during the early years of my PhD. Vilas Chitrakaran for always having a solution to robotics or software issues; his professional attitude toward work was also an inspiration. Enver Tatlicioglu for helping me out with all those math problems and for many stimulating discussions.

Finally, my stay at Clemson for almost five years would not have been possible if I didn't have some wonderful friends to support me, I'd like to thank them all for bearing with me through the tough times.

TABLE OF CONTENTS

Table of Contents (Continued)

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

List of Figures (Continued)

Figure Page

CHAPTER 1
INTRODUCTION

Organization

This dissertation is organized into four chapters. Chapter 2 presents the development of an adaptive task-space tracking controller for rigid link robot manipulators with uncertainty in their kinematic and dynamic models. The controller is developed based on the unit quaternion representation to avoid singularities associated with other three parameter representations. Also, the controller does not require the measurement of task-space velocities. There might be situations where accurate dynamic modeling of the manipulator is not possible, in these cases it is difficult to implement an adaptive controller as in Chapter 2. With this in mind, in Chapter 3, a joint level neural network tracking controller is presented which can deal with high levels of uncertainty in the robot dynamic model. The neural network based controller is applicable to rigid and flexible link manipulators as well as continuum robot manipulators since it does not depend on any specific model of the manipulator. Finally, in Chapter 4, the task of whole arm grasping using a kinematically redundant robot manipulator whose dynamic model is unknown and where the contact forces between the robot and the object are unmeasurable is considered. The whole arm grasping controller is developed by first designing a high level path planner and then using the controller developed in Chapter 3 as the joint level tracking controller. The remainder of this chapter provides an introduction and motivation to study each of the control problems being considered in this dissertation.

Tracking Control for Robots with Kinematic and Dynamic Uncertainty

The control objective in many robot manipulator applications is to command the end-effector motion to achieve a desired response. The control inputs are applied to the manipulator joints, and the desired position and orientation is typically encoded

in terms of a Cartesian coordinate frame attached to the robot end-effector with respect to the base frame (i.e., the so-called task-space variables). Hence, a mapping (i.e., the solution of the inverse kinematics) is required to convert the desired task-space trajectory into a form that can be utilized by the joint space controller. If there are uncertainties or singularities in the mapping, then this can result in degraded performance or unpredictable responses by the manipulator. Several parametrizations exist to describe orientation angles in the task-space to joint-space mapping, including three-parameter representations (e.g., Euler angles, Rodrigues parameters) and the four-parameter representation given by the unit quaternion. Three-parameter representations always exhibit singular orientations (i.e., the orientation Jacobian matrix in the kinematic equation is singular for some orientations), while the unit quaternion represents the end-effector orientation without singularities. By utilizing the singularity free unit quaternion, the emphasis of chapter 2 is to develop a tracking controller that compensates for uncertainty throughout the kinematic and dynamic models. Some previous task-space control formulations based on the unit quaternion can be found in [1], [2], [3], [4], [5], and the references therein. A quaternion-based resolved acceleration controller was presented in [2], and quaternion-based resolved rate and resolved acceleration task-space controllers were proposed in [5]. Output feedback task-space controllers using quaternion feedback were presented in [3] for the regulation problem and in [1] for the tracking problem. Model-based and adaptive asymptotic full-state feedback controllers and an output feedback controller based on a model-based observer were developed in [4] using the quaternion parametrization.

A common assumption in most of the previous robot controllers (including all of the aforementioned quaternion-based task-space control formulations) is that the robot kinematics and manipulator Jacobian are assumed to be perfectly known. From a review of literature, few controllers have been developed that target uncertainty in the manipulator forward kinematics and Jacobian. For example, in [6], [7], [8], [9], [10], [11], several approximate Jacobian feedback controllers that exploit a static,

best-guess estimate of the manipulator Jacobian to achieve task-space regulation objectives despite parametric uncertainty in the manipulator Jacobian. In [12], a task-space adaptive controller for set point control of robots with uncertainties in the gravity regressor matrix and kinematics was developed. In [13], an adaptive regulation controller for robot manipulators with uncertainty in the kinematic and dynamic models was developed. The result in [13] also accounted for actuator saturation since the maximum commanded torque could be a priori determined due to the use of saturated feedback terms in the controller. Recently in [14], an adaptive regulation controller for rigid-link electrically driven robot manipulators with uncertainty in kinematics, manipulator dynamics and actuator dynamics was developed.

All of the aforementioned controllers that account for kinematic uncertainty are based on the three-parameter Euler angle representation. Moreover, all of the previous results only target the set-point regulation problem. The only results which target the more general tracking control problem for manipulators with uncertain kinematics are given in [15], [16], [17]. However, these results are also based on the Euler angle representation and with the exception of [16] they all require the measurement of the task-space velocity. In [16], a filtered derivative of the task-space position is used to generate an approximation of the task-space velocity signal. Hence motivated by previous work, an adaptive tracking controller is developed in chapter 2 for robot manipulators with uncertainty in the kinematic and dynamic models. The controller is developed based on the unit quaternion representation so that singularities associated with three parameter representations are avoided. In addition, the developed controller does not require the measurement of the task-space velocity. The stability of the controller is proven through a Lyapunov based stability analysis. Experimental results for a planar application of the Barrett whole arm manipulator (WAM) are provided to illustrate the performance of the developed controller.

Neural Network Controller for Continuum Robots

Continuum or hyper-redundant manipulators [18, 19], belong to a special class of robotic manipulators which are designed to exhibit behavior similar to biological trunks [20–23], tentacles [24], or snakes [25]. Unlike traditional rigid link robot manipulators, continuum robot manipulators do not have rigid joints and they have a large number of degrees of freedom, this enables continuum manipulators to have some very useful properties. The continuum manipulators can be compliant, extremely dexterous, flexible, and capable of dynamic adaptive manipulation in highly unstructured environments. These properties of soft continuum robot manipulators make them uniquely suited for a large number of applications, including search and rescue, underwater and space exploration.

The development of high performance model based control algorithms for continuum manipulators is a challenging problem for several reasons; since the manipulators must be modeled as continuous curves, the kinematic and dynamic models are difficult to derive, also, the manipulators body is soft and flexible which makes accurate control difficult to achieve. There have been several different approaches which researchers have studied for the control of continuum robot manipulators. For example, Matsuno *et al.* [26], proposed kinematic control techniques for continuum manipulators and [27–30] described set-point controllers for continuum manipulators. In [27], a fuzzy controller was presented and [28] presented an artificial potential function method for obstacle avoidance for a variable length continuum manipulator. In [29], an exponentially stable controller for inextensible continuum manipulators was presented and [30] described sliding mode and impedance control techniques for hyper-flexible manipulators. There are very few techniques which target the more general tracking control problem for continuum manipulators with one of the exceptions being shape tracking control [31], where the manipulator follows a desired shape prescribed by a time-varying spatial curve. A trajectory tracking shape controller for a wheeled snake robot based on the dynamic model and the nonholonomic constraint

conditions of the robot was presented in [32]. All of the aforementioned control techniques for the tracking control of continuum manipulators require the exact dynamic model to be known.

The concept of using a neural network based control strategy for joint tracking control of a conventional robot manipulator is quite well understood. In [33, 34], and the references therein, neural network controllers were developed for a large number of robot manipulator models including rigid link manipulators and flexible joint manipulators. These two results survey in great detail the research that has been conducted on closed loop neural network control of robotic manipulators. A selection of some recent results on neural network control of robotic manipulators include [35–38]. Kim *et al.* [35], developed an output feedback controller for robot manipulators with on-line weight adaptation which provided UUB (uniformly ultimately bounded) tracking. Sun *et al.* [36] presented a discrete neural network controller for robots with uncertain dynamics which did not require off-line training, however this controller also only achieves UUB tracking. Patino *et al.* [37], developed a neural network based robust adaptive tracking controller based on static weights which must be trained off-line. The controller provided global asymptotic stability of the tracking error by including a signum function in the controller. However, the inclusion of the signum function in the controller leads to high frequency chattering in the control signals, as this is undesirable the authors propose substituting the signum function with a saturation function which leads to a degradation of the tracking performance from a global asymptotic stability tracking result to a UUB result. In [38], a parametric adaptive controller that adapts for robot dynamic parameters was coupled with a neural network to compensate for the unmodeled friction effects. The controller provided asymptotic stability but required the structure of the robot's dynamic model to be known *a priori*.

From this brief survey of results and in general from control theory, we note that the more information that a controller has about the plant, the better the track-

ing result. As has been mentioned previously, the dynamic modeling of extensible continuum manipulators is difficult and remains an active research topic. All of the previously mentioned controllers for continuum robot manipulators are either set-point controllers, or tracking controllers that require an accurate dynamic model of the manipulator. Hence, these controllers are not suitable for tracking control of continuum manipulators as they will exhibit diminished performance. This reduced performance can be a drawback as it does not allow all the capabilities of the continuum manipulator to be utilized. The problem of continuum robot control thus represents a significant barrier to progress in this emerging field.

Since a complete accurate dynamic model of the continuum manipulator does not exist, the main focus of the current work is to develop an efficient tracking controller for extensible continuum manipulators which can deal with a high level of uncertainty in the structure of the manipulators dynamic model. The proposed controller which is based on our preliminary work [39], consists of a neural network feedforward component along with a nonlinear feedback component. Specifically, the design of the neural network component is based on the augmented back propagation algorithm [33], and it is used to compensate for the nonlinear uncertain dynamics of the continuum robot manipulator by leveraging the universal approximation properties of the neural network as a feedforward compensator. The feedback component utilized is a continuous nonlinear controller [40], which does not require any model information. The advantages of the proposed control scheme compared to the previously mentioned works is that the controller is continuous and asymptotic tracking can be proved without any prior knowledge of the robot dynamic model. Furthermore, the back propagation technique enables the neural network weight matrices to be estimated on-line very quickly based on the tracking error signal and without utilizing any prior training period. This technique is particularly useful for real time closed loop control as has been described in [41].

Whole Arm Grasping Using Redundant Robot Manipulators

Kinematically redundant robot manipulators have the unique ability to perform grasps using their entire body to wrap around the object. This concept is known as whole arm manipulation, which refers to the ability of the manipulator to grasp an object with its entire body (or arm), as compared to fingertip grasping performed by traditional robotic grippers and hands, and was first described by Salisbury [42]. Whole arm grasping[1] can be performed by allowing the robot manipulator to make contact with the object in a snake or tentacle like manner, using portions of the manipulator itself to wrap around the object and grasp it. Figure 1.1, shows an example of whole arm grasping using a rigid link robot manipulator and figure 1.2, shows whole arm grasping with a continuum robot manipulator. The equivalent whole hand and whole finger grasping techniques have been studied in [45] and [46], respectively. Whole arm grasping is also known by the equivalent expressions "power grasping" ([47] and [48]) or "enveloping grasping" [49].

The whole arm approach to grasping has a number of useful properties as noted by [42], [47], [50], and others. The authors of [47] point out that distribution of contact points enables increased load capacity. The ability to use the entire body of the manipulator for grasping also allows objects of various dimensions to be grasped [42]. These capabilities could be useful for a very diverse set of applications, including, search and rescue, underwater and space exploration. However, there has been very little experimental work reported on whole arm grasping with kinematically redundant robot manipulators. Specifically, one of the few results in the literature is given in [51] where whole arm grasping with a 30 DOF robotic arm was demonstrated. Recently, Mochiyama *et al.* [52], proposed an impedance control based approach to control the shape of the whole manipulator for whole arm grasping.

Traditional robotic grasping control can be broadly classified into two main categories [53]. The first category, is a geometrical planning based approach which requires

[1]For an overview of robotic grasping and manipulation, the reader is referred to [43], [44], and the references therein.

Figure 1.1 Whole arm grasping with the Barrett WAM manipulator.

Figure 1.2 Whole arm grasping with the OCTARM continuum manipulator.

the object model and the constraint forces to be known *a priori* (e.g. [50] and [54]). Here, the grasping contact points are pre-planned and the desired constraint force for each contact point is assumed to be known. The grasping control system then moves the hand/arm along a pre-determined trajectory and force feedback (force sensors on

the arm or hand) is used to control the interaction forces. The second category for robot grasping control is the sensory approach, where the object model is unknown and the grasping controller relies on tactile force-feedback data. In this sensory based approach, it is often assumed that the arm has a sensory "skin" for force measurements [55]. The arm/hand must either start off close to the object to be grasped, or with all contact points touching the object. Then, the grasp controller positions and re-positions the arm to minimize an error function in an attempt to optimize the grasp configuration [56].

The techniques described above require either that the geometry of the object and the constraint forces be known *a priori* [54], or that the contact forces be measurable using some type of force sensor [50], [55], and [53]. When extending the traditional approaches (i.e., fingertip grasping) to whole-arm grasping, the previously mentioned requirements might not be easily met due to the increased number of contact points and the large number of grasping configurations possible [56]. Motivated by the need to have a whole arm grasping controller which does not require the constraint forces to be known *a priori* while also eliminating the requirement for contact force sensing, a grasping controller for kinematically redundant robot manipulators is designed which requires only the object geometry to be known *a priori*. In addition, the proposed controller does not require the exact dynamic model for the robot manipulator or the measurement of contact forces. This paradigm makes the whole arm grasping technique easily extendable to various manipulator systems.

CHAPTER 2
TRACKING CONTROL FOR ROBOT MANIPULATORS WITH KINEMATIC AND DYNAMIC UNCERTAINTY

The control objective in many robot manipulator applications is to command the end-effector motion to achieve a desired response. To achieve this objective a mapping is required to relate the joint/link control inputs to the desired Cartesian position and orientation. If there are uncertainties or singularities in the mapping, then degraded performance or unpredictable responses by the manipulator are possible. To address these issues, in this chapter, an adaptive task-space tracking controller is developed for robot manipulators with uncertainty in the kinematic and dynamic models. The controller is developed based on the unit quaternion representation so that singularities associated with three parameter representations are avoided. In addition, the developed controller does not require the measurement of the task-space velocity. The stability of the controller is proven through a Lyapunov based stability analysis. Simulation results using a two degree of freedom planar manipulator model are presented to validate the controllers performance, then experimental results for a planar application of the Barrett whole arm manipulator (WAM) are provided to illustrate the performance of the developed controller.

Robot Dynamic and Kinematic Models

A six-link, rigid, revolute robot manipulator can be described by the following dynamic model [57]:

$$M(\theta)\ddot{\theta} + V_m(\theta, \dot{\theta})\dot{\theta} + G(\theta) + F_d\,\dot{\theta} = \tau \tag{2.1}$$

In (2.1), $\theta\,(t) \in \mathbb{R}^6$ is the joint position (It is assumed that the actuated manipulator joint is rigidly connected to the links, so that the link-space and joint-space are equivalent. Hence, the words joint and link can be used interchangeably.), $M\,(\theta) \in \mathbb{R}^{6\times 6}$ represents the inertia matrix, $V_m(\theta, \dot{\theta}) \in \mathbb{R}^{6\times 6}$ is the centripetal-Coriolis matrix,

$G(\theta) \in \mathbb{R}^6$ is the gravity vector, $F_d \in \mathbb{R}^{6\times 6}$ is a constant diagonal matrix which represents the viscous friction coefficients, and $\tau(t) \in \mathbb{R}^6$ represents the input torque vector. The dynamic model given in (2.1) has the following properties [57], which are utilized in the subsequent control design and analysis:

Property 1 *The inertia matrix is symmetric and positive-definite, and satisfies the following inequalities:*

$$m_1 \|x\|^2 \leq x^T M(\theta) x \leq m_2 \|x\|^2 \qquad \forall\, x \in \mathbb{R}^6 \tag{2.2}$$

where m_1, $m_2 \in \mathbb{R}$ are positive constants and $\|\cdot\|$ denotes the standard Euclidean norm.

Property 2 *The inertia and centripetal-Coriolis matrices satisfy the following skew-symmetric relationship:*

$$x^T \left(\frac{1}{2} \dot{M}(\theta) - V_m(\theta, \dot{\theta}) \right) x = 0 \qquad \forall\, x \in \mathbb{R}^6 \tag{2.3}$$

Property 3 *The centripetal-Coriolis matrix satisfies the following skew-symmetric relationship:*

$$V_m(\theta, x) y = V_m(\theta, y) x \qquad \forall\, x, y \in \mathbb{R}^6 \tag{2.4}$$

Property 4 *The norm of the centripetal-Coriolis matrix and the norm of the friction matrix, can be upper bounded as follows:*

$$\|V_m(\theta, x)\|_{i\infty} \leq \zeta_c \|x\| \qquad \forall\, x \in \mathbb{R}^6, \quad \|F_d\| \leq \zeta_f \tag{2.5}$$

where ζ_c, $\zeta_f \in \mathbb{R}$ are positive constants, and $\|\cdot\|_{i\infty}$ denotes the induced-infinity norm of a matrix.

Property 5 *Parametric uncertainty in $M(\theta)$, $V_m(\theta, \dot{\theta})$, $G(\theta)$ and F_d, is linearly parametrizable.*

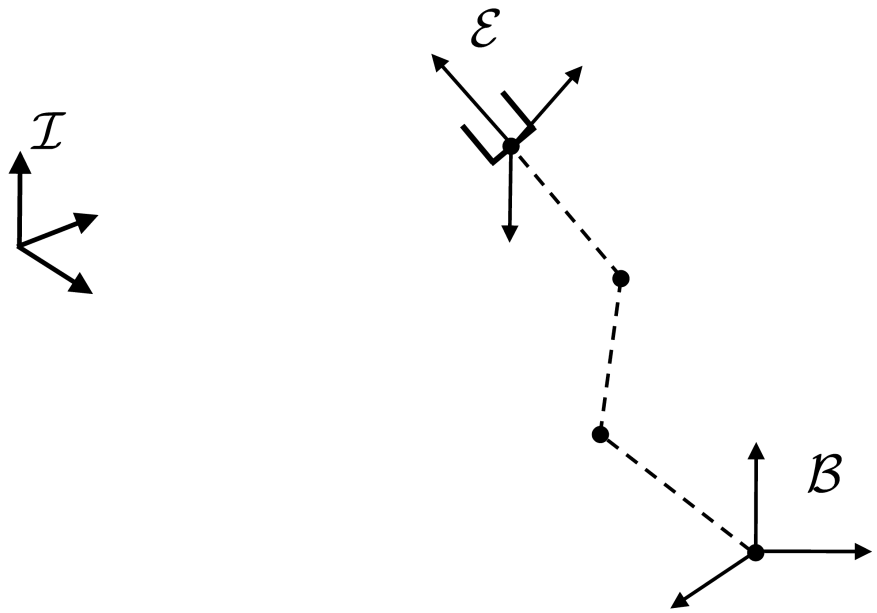

Figure 2.1 Representation of coordinate frames for the system.

Let $\mathcal{E}$ and $\mathcal{B}$ be orthogonal coordinate frames attached to the manipulator's end-effector and fixed base, respectively. Let $\mathcal{I}$ be the coordinate frame used to measure[2] the position and orientation of $\mathcal{E}$ relative to $\mathcal{B}$, for example $\mathcal{I}$ could be the camera coordinate frame. The position and orientation of $\mathcal{E}$ relative to $\mathcal{B}$ can be represented through the following forward kinematic model [3]:

$$\begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} h_p(\theta) \\ h_q(\theta) \end{bmatrix} \tag{2.6}$$

In (2.6), $h_p(\cdot) : \mathbb{R}^6 \to \mathbb{R}^3$ denotes an uncertain function that maps $\theta(t)$ to the measurable task-space position coordinates of the end-effector, denoted by $p(\cdot) \in \mathbb{R}^3$, and $h_q(\cdot) : \mathbb{R}^6 \to \mathbb{R}^4$ denotes an uncertain function that maps $\theta(t)$ to the measurable unit quaternion and is denoted by $q(t) \in \mathbb{R}^4$. The unit quaternion vector, denoted by $q(t) = \left[q_o(t), q_v^T(t)\right]^T$ with $q_o(t) \in \mathbb{R}$ and $q_v(t) \in \mathbb{R}^3$ [58], [59], provides a global

[2]The task-space position and orientation of $\mathcal{E}$ relative to $\mathcal{B}$ is assumed to be measurable, as in [6–17]. For example, a camera system or laser tracking could be utilized.

non-singular parametrization of the end-effector orientation, and is subject to the constraint, $q^T q = 1$. Several algorithms exist to determine the orientation of $\mathcal{E}$ relative to $\mathcal{B}$ from a rotation matrix that is a function of $\theta(t)$. Conversely, a rotation matrix, denoted by $R(q) \in SO(3)$, can be determined from a given $q(t)$ by the formula [3]:

$$R(q) = \left(q_o^2 - q_v^T q_v\right) I_3 + 2 q_v q_v^T + 2 q_o q_v^{\times} \tag{2.7}$$

where I_3 is the 3×3 identity matrix, and the notation $a^{\times}$, $\forall\, a = [a_1,\, a_2,\, a_3]^T$, denotes the following skew-symmetric matrix:

$$a^{\times} \triangleq \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{2.8}$$

The time derivative of (2.6) is given by the following expression[3]:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} J_p \\ J_q \end{bmatrix} \dot{\theta} \tag{2.9}$$

where $J_p(\theta) : \mathbb{R}^6 \to \mathbb{R}^{3\times 6}$ and $J_q(\theta) : \mathbb{R}^6 \to \mathbb{R}^{4\times 6}$ denote the uncertain position and orientation Jacobian matrices, respectively, defined as $J_p(\theta) = \partial h_p / \partial \theta$ and $J_q(\theta) = \partial h_q / \partial \theta$. To facilitate the subsequent development, (2.9) is expressed as follows:

$$\begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = J(\theta)\dot{\theta} \;\text{ where }\; J(\theta) = \begin{bmatrix} J_p \\ B^T J_q \end{bmatrix} \in \mathbb{R}^{6\times 6} \tag{2.10}$$

The expression in (2.10) is obtained by exploiting the fact that $q(t)$ is related to the angular velocity of the end-effector, denoted by $\omega(t) \in \mathbb{R}^3$, via the following differential equation:

$$\omega = B^T \dot{q} \tag{2.11}$$

where the known Jacobian-like matrix $B(q) : \mathbb{R}^4 \to \mathbb{R}^{4\times 3}$ is defined as follows:

$$B = \frac{1}{2} \begin{bmatrix} -q_v^T \\ q_o I_3 - q_v^{\times} \end{bmatrix} \tag{2.12}$$

[3]To simplify the notation, the arguments of some functions in the equations are omitted. However, all functions are explicitly defined in the text.

Remark 1 *The dynamic and kinematic terms for a general revolute robot manipulator, denoted by $M(\theta)$, $V_m(\theta, \dot{\theta})$, $G(\theta)$ and $J(\theta)$, are assumed to depend on $\theta(t)$ only as arguments of trigonometric functions and hence, remain bounded for all possible $\theta(t)$. During the control development, the assumption will be made that if $p(t) \in \mathcal{L}_\infty$, then $\theta(t) \in \mathcal{L}_\infty$ (Note that $q(t)$ is always bounded, since $q^T q = 1$).*

Property 6 *The kinematic system in (2.10) can be linearly parametrized as follows:*

$$J\dot{\theta} = W_j \phi_j \tag{2.13}$$

where $W_j(\theta, \dot{\theta}) \in \mathbb{R}^{6 \times n_1}$ denotes a regression matrix which consists of known and measurable signals, and $\phi_j \in \mathbb{R}^{n_1}$ denotes a vector of n_1 unknown constants.

Property 7 *There exists upper and lower bounds for the parameter ϕ_j such that $J(\theta, \phi_j)$ is always invertible. We will assume that the bounds for each parameter can be calculated as follows:*

$$\underline{\phi}_{ji} \leq \phi_{ji} \leq \overline{\phi}_{ji} \tag{2.14}$$

where, $\phi_{ji} \in \mathbb{R}$ denotes the i^{th} component of $\phi_j \in \mathbb{R}^{n_1}$ and $\underline{\phi}_{ji}$, $\overline{\phi}_{ji} \in \mathbb{R}$ denote the i^{th} components of $\underline{\phi}_j$, $\overline{\phi}_j \in \mathbb{R}^{n_1}$, which are defined as follows:

$$\begin{aligned} \underline{\phi}_j &= \left[\underline{\phi}_{j1}, \underline{\phi}_{j2}, \cdots, \underline{\phi}_{jn_1}\right]^T \\ \overline{\phi}_j &= \left[\overline{\phi}_{j1}, \overline{\phi}_{j2}, \cdots, \overline{\phi}_{jn_1}\right]^T \end{aligned} \tag{2.15}$$

Problem Statement

The objective is to the design the control input $\tau(t)$ to ensure end-effector position and orientation tracking for the robot model given by (2.1) and (2.10) despite parametric uncertainty in the kinematic and dynamic models. We will assume that the only measurable signals are the joint position, joint velocity, and end-effector position. To mathematically quantify this objective, a desired position and orientation of the robot end-effector is defined by a desired orthogonal coordinate frame $\mathcal{E}_d$. The

vector $p_d(t) \in \mathbb{R}^3$ denotes the position of the origin of $\mathcal{E}_d$ relative to the origin of $\mathcal{B}$, while the rotation matrix from $\mathcal{E}_d$ to $\mathcal{B}$ is denoted by $R_d(t) \in SO(3)$.

The end-effector position tracking error $e_p(t) \in \mathbb{R}^3$ is defined as:

$$e_p = p_d - p \tag{2.16}$$

where $p_d(t)$, $\dot{p}_d(t)$, and $\ddot{p}_d(t)$ are assumed to be known bounded functions of time. If the orientation of $\mathcal{E}_d$ relative to $\mathcal{B}$ is specified in terms of a desired unit quaternion $q_d(t) = \left[q_{od}(t),\, q_{vd}^T(t)\right]^T \in \mathbb{R}^4$, with $q_{od}(t) \in \mathbb{R}$ and $q_{vd}(t) \in \mathbb{R}^3$. Then similarly to (2.7), the rotation matrix from $\mathcal{E}_d$ to $\mathcal{B}$ can be calculated from the desired unit quaternion $q_d(t)$ as follows:

$$R_d(q_d) = \left(q_{od}^2 - q_{vd}^T q_{vd}\right) I_3 + 2 q_{vd} q_{vd}^T + 2 q_{od} q_{vd}^{\times} \tag{2.17}$$

where it is assumed that R_d, $\dot{R}_d$, $\ddot{R}_d \in \mathcal{L}_\infty$. As in (2.11), the time derivative of $q_d(t)$ is related to the desired angular velocity of the end-effector (i.e., the angular velocity of $\mathcal{E}_d$ relative to $\mathcal{B}$), denoted by $\omega_d(t) \in \mathbb{R}^3$, through the known kinematic equation:

$$\dot{q}_d = B(q_d)\omega_d \tag{2.18}$$

To quantify the difference between the actual and desired end-effector orientations, we define the rotation matrix $\tilde{R} \in SO(3)$ from $\mathcal{E}$ to $\mathcal{E}_d$ as follows:

$$\tilde{R} \triangleq R_d^T R = \left(e_o^2 - e_v^T e_v\right) I_3 + 2 e_v e_v^T + 2 e_o e_v^{\times} \tag{2.19}$$

where $e_q(t) \triangleq \left[e_o(t),\, e_v^T(t)\right]^T \in \mathbb{R}^4$ represents unit quaternion tracking error that satisfies the constraint:

$$e_q^T e_q = e_o^2 + e_v^T e_v = 1 \tag{2.20}$$

The quaternion tracking error $e_q(t)$ can be explicitly calculated from $q(t)$ and $q_d(t)$ via quaternion algebra by noticing that the quaternion equivalent of $\tilde{R} = R_d^T R$ is the following quaternion product [5], [59]:

$$e_q = q q_d^* \tag{2.21}$$

where $q_d^*(t) \triangleq \left[q_{od}(t),\ -q_{vd}^T(t)\right]^T \in \mathbb{R}^4$ is the unit quaternion representing the rotation matrix $R_d^T(q_d)$. After using quaternion algebra, the quaternion tracking error can be derived as follows (see [5] and Theorem 5.3 of [59]):

$$\begin{bmatrix} e_o \\ e_v \end{bmatrix} = \begin{bmatrix} q_o q_{od} + q_v^T q_{vd} \\ q_{od} q_v - q_o q_{vd} + q_v^{\times} q_{vd} \end{bmatrix} \tag{2.22}$$

Based on (2.11), (2.18), and (2.22), the unit quaternion error system can be formulated as follows [60]:

$$\begin{bmatrix} \dot{e}_o \\ \dot{e}_v \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} e_v^T \tilde{\omega} \\ \frac{1}{2}\left(e_o I_3 - e_v^{\times}\right)\tilde{\omega} \end{bmatrix} \tag{2.23}$$

The angular velocity of $\mathcal{E}$ with respect to $\mathcal{E}_d$ with coordinates in $\mathcal{E}_d$, denoted by $\tilde{\omega}(t) \in \mathbb{R}^3$, can be calculated from (2.19) as follows [61]:

$$\tilde{\omega} = R_d^T\left(\omega - \omega_d\right) \tag{2.24}$$

The end-effector tracking errors are then written using (2.10), (2.16), and (2.24) as:

$$\begin{bmatrix} \dot{e}_p \\ \tilde{\omega} \end{bmatrix} = \Lambda\left(\begin{bmatrix} -\dot{p}_d \\ -\omega_d \end{bmatrix} + J\dot{\theta}\right) \tag{2.25}$$

where $\Lambda \in \mathbb{R}^{6\times 6}$ is defined as:

$$\Lambda = \begin{bmatrix} -I_3 & 0_{3\times 3} \\ 0_{3\times 3} & R_d^T \end{bmatrix} \tag{2.26}$$

where $0_{3\times 3}$ represents a 3×3 matrix of zeros. Based on the above definitions, the tracking objective defined in terms of the end-effector position and unit quaternion error is to design the control input $\tau(t)$ such that:

$$\|e_p(t)\| \to 0 \quad \text{and} \quad \|e_v(t)\| \to 0 \quad \text{as} \quad t \to \infty \tag{2.27}$$

The orientation tracking objective given in (2.27) can also be stated in terms of $e_q(t)$. Specifically, (2.20) implies that:

$$0 \le \|e_v(t)\| \le 1 \qquad \text{and} \qquad 0 \le |e_o(t)| \le 1 \tag{2.28}$$

for all time and if $\|e_v(t)\| \rightarrow 0$ as $t \rightarrow \infty$ then $e_o(t) \rightarrow 1$ as $t \rightarrow \infty$. Thus, if $\|e_v(t)\| \rightarrow 0$ as $t \rightarrow \infty$ then (2.19) along with the previous statement can be used to conclude that $\tilde{R}(t) \rightarrow I_3$ as $t \rightarrow \infty$, and hence, the orientation tracking objective can be achieved.

Tracking Error System Development

To facilitate the development of the open-loop error system, an auxiliary variable $\eta(t) \in \mathbb{R}^6$ is defined as follows:

$$\eta = \left(\Lambda \hat{J}\right)^{-1} \left[\begin{array}{c} \dot{p}_d + k_1 e_p \\ -R_d^T \omega_d + k_2 e_v \end{array} \right] + \dot{\theta} \tag{2.29}$$

where $k_1, k_2 \in \mathbb{R}^{3\times 3}$ are positive, constant, diagonal matrices, and $\hat{J}(\theta, \hat{\phi}_j) \in \mathbb{R}^{6\times 6}$ is an estimated manipulator Jacobian matrix. After adding and subtracting the terms $\Lambda \hat{J}(\theta, \hat{\phi}_j)\dot{\theta}(t)$ and $\Lambda \hat{J}(\theta, \hat{\phi}_j)\eta(t)$ to (2.25) and utilizing (2.26), the following kinematic error system can be developed:

$$\left[\begin{array}{c} \dot{e}_p \\ \tilde{\omega} \end{array} \right] = - \left[\begin{array}{c} k_1 e_p \\ k_2 e_v \end{array} \right] + \Lambda \left(\hat{J}\eta + W_j \tilde{\phi}_j \right) \tag{2.30}$$

where $W_j(\cdot) \in \mathbb{R}^{6\times n_1}$ was introduced in (2.13) and the parameter estimation error term $\tilde{\phi}_j(t) \in \mathbb{R}^{n_1}$ is defined as:

$$\tilde{\phi}_j = \phi_j - \hat{\phi}_j \tag{2.31}$$

The adaptive estimate $\hat{\phi}_j(t) \in \mathbb{R}^{n_1}$ introduced in (2.31) is designed as follows:

$$\dot{\hat{\phi}}_j = proj\left\{y\right\} \tag{2.32}$$

where the auxiliary term $y \in \mathbb{R}^{n_1}$ is defined as:

$$y = \Gamma_1 W_j^T \Lambda^T \left[\begin{array}{c} e_p \\ e_v \end{array} \right] \tag{2.33}$$

where $\Gamma_1 \in \mathbb{R}^{n_1 \times n_1}$ is a constant positive diagonal matrix and the function $proj\{y\}$

is defined as follows:

$$proj\{y_i\} \triangleq \begin{cases} y_i & \text{if } \hat{\phi}_{ji} > \underline{\phi}_{ji} \\ y_i & \text{if } \hat{\phi}_{ji} = \underline{\phi}_{ji} \quad \text{and } y_i > 0 \\ 0 & \text{if } \hat{\phi}_{ji} = \underline{\phi}_{ji} \quad \text{and } y_i < 0 \\ 0 & \text{if } \hat{\phi}_{ji} = \overline{\phi}_{ji} \quad \text{and } y_i > 0 \\ y_i & \text{if } \hat{\phi}_{ji} = \overline{\phi}_{ji} \quad \text{and } y_i \leq 0 \\ y_i & \text{if } \hat{\phi}_{ji} < \overline{\phi}_{ji} \end{cases} \tag{2.34}$$

$$\underline{\phi}_{ji} \leq \hat{\phi}_{ji}(0) \leq \overline{\phi}_{ji} \tag{2.35}$$

where y_i denotes the i^{th} component of y, and $\hat{\phi}_{ji}(t)$ denotes the i^{th} component of $\hat{\phi}_j(t)$ (Note that the above projection algorithm ensures that $\underline{\phi}_j \leq \hat{\phi}_j(t) \leq \overline{\phi}_j$ and hence, using Property 7 we can observe that the estimated manipulator Jacobian matrix $\hat{J}(\theta, \hat{\phi}_j)$ will always be non-singular. For further details of the projection algorithm the reader is referred [62]).

To obtain the closed loop error system for $\eta(t)$, we first take the time derivative of (2.29) to obtain the following expression:

$$\dot{\eta} = \frac{d}{dt}\left\{\left(\Lambda \hat{J}\right)^{-1}\begin{bmatrix} \dot{p}_d + k_1 e_p \\ -R_d^T \omega_d + k_2 e_v \end{bmatrix}\right\} + \ddot{\theta} \tag{2.36}$$

After pre-multiplying (2.36) by $M(\theta)$, substituting (2.1) into the resulting expression for $M(\theta)\ddot{\theta}(t)$, and utilizing (2.29), the following simplified expression can be obtained:

$$M\dot{\eta} = -V_m \eta + \tau + W_y \phi_y \tag{2.37}$$

where $W_y(p_d, \dot{p}_d, \ddot{p}_d, q_d, \omega_d, \dot{\omega}_d, p, q, \theta, \dot{\theta}) \in \mathbb{R}^{6\times n_2}$ is a regression matrix of known and measurable quantities, and $\phi_y \in \mathbb{R}^{n_2}$ is a vector of n_2 unknown constant parameters. The product $W_y(\cdot)\phi_y$ introduced in (2.37) is defined as:

$$\begin{aligned} W_y \phi_y &= M\frac{d}{dt}\left\{\left(\Lambda \hat{J}\right)^{-1}\begin{bmatrix} \dot{p}_d + k_1 e_p \\ -R_d^T \omega_d + k_2 e_v \end{bmatrix}\right\} \\ &\quad + V_m\left(\Lambda \hat{J}\right)^{-1}\begin{bmatrix} \dot{p}_d + k_1 e_p \\ -R_d^T \omega_d + k_2 e_v \end{bmatrix} - G(\theta) - F_d\,\dot{\theta} \end{aligned} \tag{2.38}$$

Based on (2.37) and the subsequent stability analysis, the control input $\tau(t)$ is designed as:

$$\tau = -W_y \hat{\phi}_y - k_r \eta - \left(\Lambda \hat{J}\right)^T \begin{bmatrix} e_p \\ e_v \end{bmatrix} \tag{2.39}$$

where $k_r \in \mathbb{R}^{6\times 6}$ is a constant positive diagonal matrix and $\hat{\phi}_y(t) \in \mathbb{R}^{n_2}$ denotes an adaptive estimate which is generated by the following differential expression:

$$\dot{\hat{\phi}}_y = \Gamma_2 W_y^T \eta \tag{2.40}$$

where $\Gamma_2 \in \mathbb{R}^{n_2\times n_2}$ is a positive constant diagonal matrix. After substituting (2.39) into (2.37), the following closed-loop error system is obtained:

$$M\dot{\eta} = -V_m\eta + W_y\tilde{\phi}_y - k_r\eta - \left(\Lambda\hat{J}\right)^T \begin{bmatrix} e_p \\ e_v \end{bmatrix} \tag{2.41}$$

where the adaptive estimation error is defined as:

$$\tilde{\phi}_y = \phi_y - \hat{\phi}_y \tag{2.42}$$

Remark 2 *Based on the definition of the quaternion error system in (2.23), the kinematic error system in (2.30), and the regression matrix in (2.38), we can conclude that $W_y(\cdot)$ does not require the measurement of the task-space velocity. Further, from the definition of $e_p(t), e_v(t),$ and $\eta(t)$ it is clear that the control input torque $\tau(t)$ does not require measurement of the task space velocity.*

Remark 3 *Although the development presented in this chapter is for a six degree of freedom robot manipulator, the control design can be easily extended to include a kinematically redundant robot manipulator. The modifications in the control design for a kinematically redundant robot manipulator would be similar to the work presented in [60], with the addition that the manipulator Jacobian be uncertain. It is also interesting to note that the null space controller can be designed as in [63] to accomplish several different subtasks, i.e. the self motion of the kinematically redundant robot manipulator can be controlled to achieve a secondary control objective.*

Stability Analysis

Theorem 1 *Given the robotic system described by (2.1), the control input (2.39) along with the adaptive laws defined in (2.32) and (2.40) guarantee asymptotic regulation of the end-effector position error and the unit quaternion error in the sense that* $\|e_p(t)\| \to 0$ *as* $t \to \infty$ *and* $\|e_v(t)\| \to 0$ *as* $t \to \infty$*, thus completing the position and orientation tracking objective.*

Proof. Let $V(t) \in \mathbb{R}$ denote the following non-negative scalar function:

$$\begin{aligned} V &= \frac{1}{2}e_p^T e_p + (1-e_0)^2 + e_v^T e_v + \frac{1}{2}\eta^T M \eta \\ &\quad + \frac{1}{2}\tilde{\phi}_j^T \Gamma_1^{-1} \tilde{\phi}_j + \frac{1}{2}\tilde{\phi}_y^T \Gamma_2^{-1} \tilde{\phi}_y \end{aligned} \tag{2.43}$$

After taking the time derivative of (2.43) and utilizing (2.23), (2.31) and (2.42), the following expression is obtained:

$$\begin{aligned} \dot{V} &= e_p^T \dot{e}_p + (1-e_0)\left(e_v^T \tilde{\omega}\right) + e_v^T \left(e_0 I_3 - e_v^{\times}\right)\tilde{\omega} + \frac{1}{2}\eta^T \dot{M}\eta + \eta^T M \dot{\eta} \\ &\quad - \tilde{\phi}_y^T \Gamma_2^{-1} \dot{\hat{\phi}}_y - \tilde{\phi}_j^T \Gamma_1^{-1} \dot{\hat{\phi}}_j \end{aligned} \tag{2.44}$$

Upon further simplification of equation (2.44) by cancelling common terms, and substituting for $M\dot{\eta}$ from (2.41), the following expression for $\dot{V}(t)$ can be obtained:

$$\begin{aligned} \dot{V} &= \begin{bmatrix} e_p^T & e_v^T \end{bmatrix} \begin{bmatrix} \dot{e}_p \\ \tilde{\omega} \end{bmatrix} - \eta^T k_r \eta - \eta^T V_m \eta + \eta^T W_y \tilde{\phi}_y - \eta^T \left(\Lambda \hat{J}\right)^T \begin{bmatrix} e_p \\ e_v \end{bmatrix} \\ &\quad + \frac{1}{2}\eta^T \dot{M}\eta - \tilde{\phi}_j^T \Gamma_1^{-1} \dot{\hat{\phi}}_j - \tilde{\phi}_y^T \Gamma_2^{-1} \dot{\hat{\phi}}_y \end{aligned} \tag{2.45}$$

After using Property 3, substituting from (2.30), (2.32), and (2.40) and cancelling terms, $\dot{V}(t)$ can be expressed as:

$$\begin{aligned} \dot{V} &= \begin{bmatrix} e_p^T & e_v^T \end{bmatrix} \left(- \begin{bmatrix} k_1 e_p \\ k_2 e_v \end{bmatrix} + \Lambda W_j \tilde{\phi}_j \right) \\ &\quad - \eta^T k_r \eta - \tilde{\phi}_j^T \Gamma_1^{-1} proj\{y\} \end{aligned} \tag{2.46}$$

Substituting for y from (2.33) and using the definition of the projection function, (2.34), the expression for $\dot{V}(t)$ can be upper bounded as follows:

$$\dot{V} \leq -\lambda_{min}\{k_1\} \|e_p\|^2 - \lambda_{min}\{k_2\} \|e_v\|^2 - \lambda_{min}\{k_r\} \|\eta\|^2 \tag{2.47}$$

where λ_{min} is the minimum Eigenvalue of the matrix.

The expressions in (2.43) and (2.47) can be used to prove that $e_p(t)$, $e_v(t)$, $\eta(t)$, $\tilde{\phi}_j(t)$, $\tilde{\phi}_y(t) \in \mathcal{L}_\infty$ and that $e_p(t)$, $e_v(t)$, $\eta(t) \in \mathcal{L}_2$. Using (2.16) and the assumption that $p_d(t) \in \mathcal{L}_\infty$, it is clear that $p(t) \in \mathcal{L}_\infty$. From (2.31) and (2.42) it can be concluded that $\hat{\phi}_j(t)$, $\hat{\phi}_y(t) \in \mathcal{L}_\infty$. Utilizing Property 7, the definition of $\eta(t)$ in (2.29) and the fact that $e_p(t)$, $e_v(t)$, $\eta(t) \in \mathcal{L}_\infty$, we can show that $\dot{\theta}(t) \in \mathcal{L}_\infty$. Moreover, (2.9), (2.16) and the fact that $J(\theta) \in \mathcal{L}_\infty$ can be used to show that $\dot{p}(t)$, $\dot{e}_p(t) \in \mathcal{L}_\infty$. From (2.20), (2.23), (2.25) and (2.28) we can show that $e_0(t)$, $\dot{e}_0(t)$, $\dot{e}_v(t) \in \mathcal{L}_\infty$. From the definition of $W_j(\cdot)$ and $W_y(\cdot)$ in (2.13) and (2.38) respectively and the preceding arguments, it is clear that $W_y(\cdot)$, $W_j(\cdot) \in \mathcal{L}_\infty$. Utilizing (2.32), (2.33), (2.34) and (2.40), we can show that $\dot{\hat{\phi}}_j(t)$, $\dot{\hat{\phi}}_y(t) \in \mathcal{L}_\infty$. The definition of $\tau(t)$ in (2.39) can be used to show that $\tau(t) \in \mathcal{L}_\infty$; hence, $\theta(t)$, $\dot{\theta}(t)$, $\ddot{\theta}(t) \in \mathcal{L}_\infty$ and from (2.36) we can conclude that $\dot{\eta}(t) \in \mathcal{L}_\infty$. Since $\dot{e}_p(t)$, $\dot{e}_v(t)$, $\dot{\eta}(t) \in \mathcal{L}_\infty$ and $e_p(t)$, $e_v(t)$, $\eta(t) \in \mathcal{L}_2$, Barbalat's Lemma [64] can be used to show that, $\|e_p(t)\| \to 0$, $\|e_v(t)\| \to 0$, $\|\eta(t)\| \to 0$ as $t \to \infty$. ■

Simulation Results

To evaluate the performance of the proposed control strategy, the controller was simulated using the dynamics of a two degree of freedom (d.o.f.) planar robot manipulator. In the simulation the position measurements for the end-effector were obtained using the known forward kinematics of the manipulator. The simulation was written in "C++" and hosted on Qmotor [65] in the QNX 6.2.1 real-time operating system. The Qmotor software was selected to simulate the controller since it provides on-line parameter tuning and data logging capabilities. Also, the simulation can be performed in real-time in this software environment which significantly reduces the time required to evaluate the control algorithm. Figures 2.2, 2.3, 2.4, 2.5, and 2.6, show

the results obtained from the simulation for a circular trajectory in the task space. It is seen that the tracking error approaches zero as the kinematic and dynamic parameters converge. Note that the adaptive algorithm presented in this work does not guarantee that the parameter estimates converge to their true values.

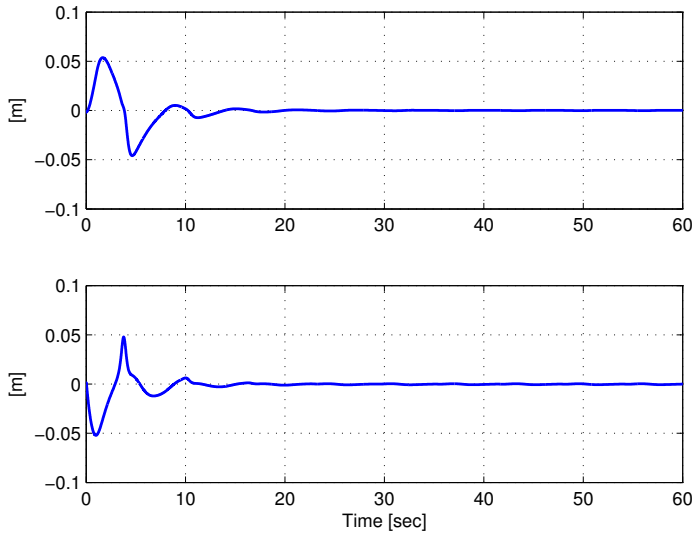

Figure 2.2 End-effector trajectory tracking error for the simulation.

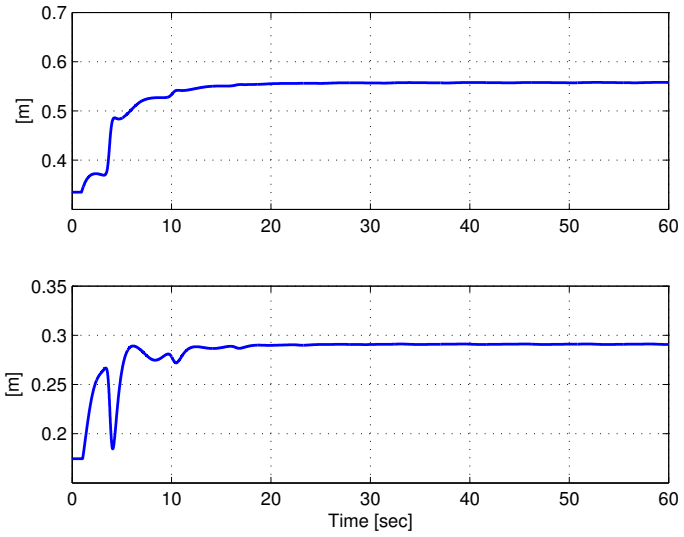

Figure 2.3 Kinematic parameter estimates for the simulation.

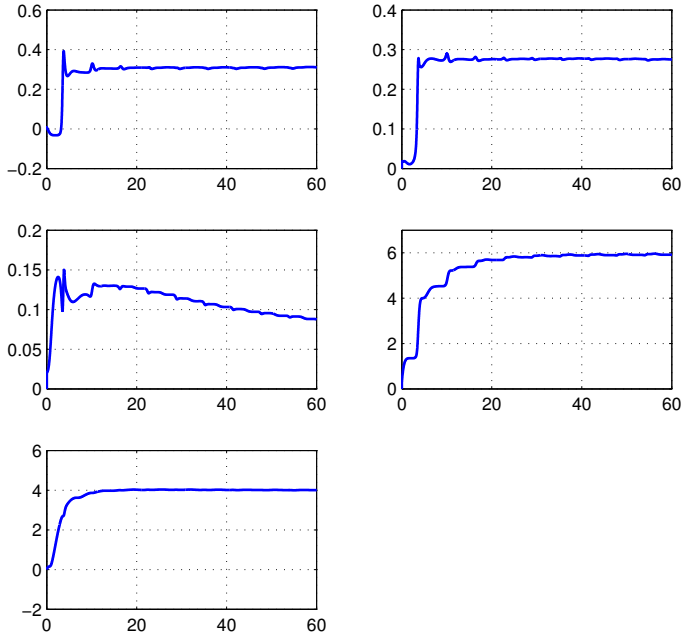

Figure 2.4 Dynamic parameter estimates for the simulation.

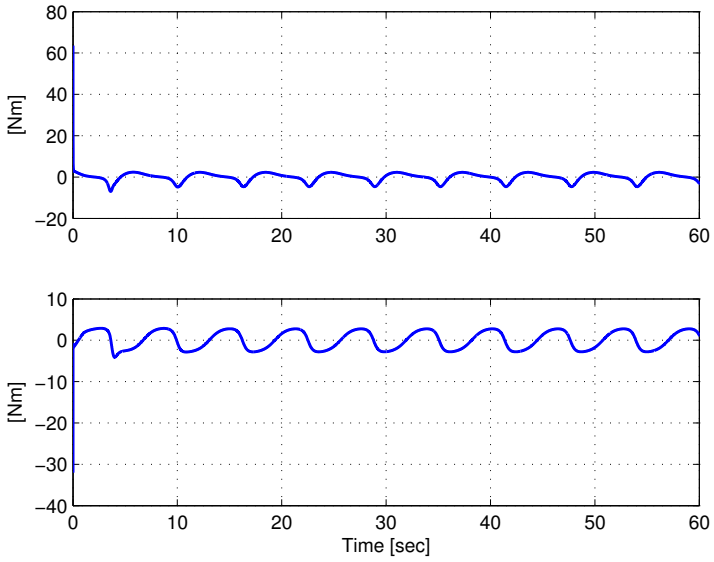

Figure 2.5 Control input torque for the simulation.

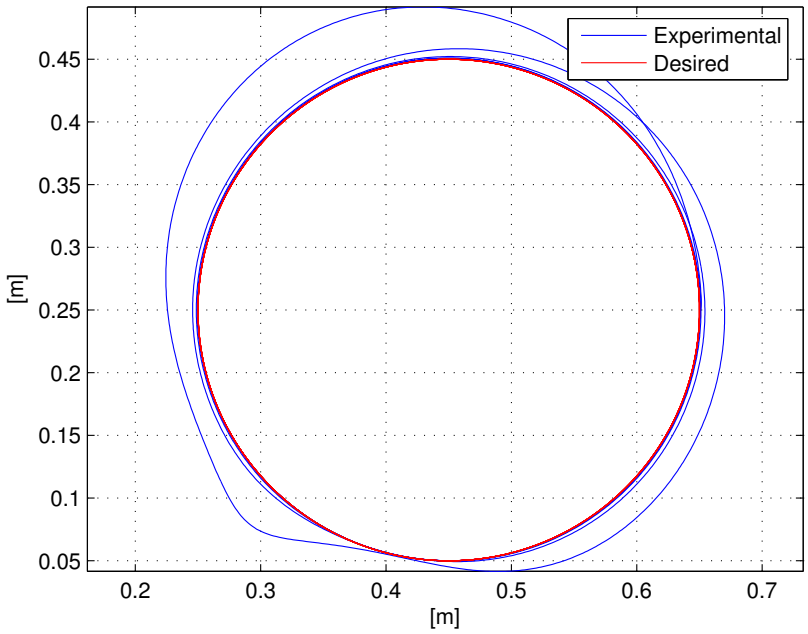

Figure 2.6 Actual and desired end-effector trajectory for the simulation.

Experimental Results

The developed controller was implemented on the Barrett whole arm manipulator (WAM). The WAM is a seven degree of freedom (d.o.f.), highly dexterous and back-drivable robotic manipulator. The objective of the experiment is to verify the performance of the developed adaptive controller. So to simplify the controller implementation, five joints of the robot were locked at fixed angles and the remaining links of the manipulator were used as a two d.o.f. planar robot manipulator (refer to Fig. 2.7). The dynamics of the robot in this planar configuration can be expressed as [61]:

$$\begin{aligned} \tau &= \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} V_{m11} & V_{m12} \\ V_{m21} & V_{m22} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \\ &\quad + \begin{bmatrix} f_{d1} & 0 \\ 0 & f_{d2} \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} \end{aligned} \tag{2.48}$$

Figure 2.7 Barrett Whole Arm Manipulator.

The elements of the inertia and centripetal-Coriolis matrices are defined as follows:

$$\begin{aligned}
M_{11} &= m_1 l_{c1}^2 + m_2 l_{c2}^2 + m_2 l_1^2 + 2 m_2 l_1 l_{c2} \cos(\theta_2) \\
M_{12} &= m_2 l_{c2}^2 + m_2 l_1 l_{c2} \cos(\theta_2) \\
M_{21} &= M_{12} \qquad M_{22} = m_2 l_{c2}^2 \\
V_{m11} &= -m_2 l_1 l_{c2} \sin(\theta_2)\, \dot{\theta}_2 \\
V_{m12} &= -m_2 l_1 l_{c2} \sin(\theta_2) \left(\dot{\theta}_1 + \dot{\theta}_2 \right) \\
V_{m21} &= m_2 l_1 l_{c2} \sin(\theta_2)\, \dot{\theta}_1 \qquad V_{m22} = 0
\end{aligned}$$

where m_1, $m_2 \in \mathbb{R}$ denote the mass of the links, l_1, $l_2 \in \mathbb{R}$ denote the length of the links and l_{c1}, $l_{c2} \in \mathbb{R}$ denote the distance to the centre of mass. The terms f_{d1}, $f_{d2} \in \mathbb{R}$ in (2.48) denote the uncertain friction coefficients of the manipulator. The vector of

uncertain constant dynamic parameters $\phi_y \in \mathbb{R}^{14}$ was found to be:

$$\phi_y = \begin{bmatrix} m_1 l_1 l_{c1}^2, & m_1 l_2 l_{c1}^2, & m_1 l_{c1}^2, & m_2 l_1 l_{c2}^2, & m_2 l_2 l_{c2}^2, & m_2 l_{c2}^2, & m_2 l_1^3, \\ & m_2 l_1^2, & m_2 l_1^2 l_{c2}, & m_2 l_1 l_2 l_{c2}, & m_2 l_1^2 l_2, & m_2 l_1 l_{c2}, & f_{d1}, \quad f_{d2} \end{bmatrix}^T$$

The control algorithm was written in "C++" and hosted on an AMD Athlon 1.2 GHz PC operating under QNX 6.2.1. Data logging and on-line gain tuning were performed using Qmotor 3.0 control software [65]. Data acquisition and control implementation were performed at a frequency of 1.0 kHz using the ServoToGo I/O board. Joint positions were measured using the optical encoders located at the motor shaft of each axis. Joint velocity measurements were obtained using a filtered backwards difference algorithm.

Remark 4 *The kinematics of the robotic system are assumed to be unknown. The task-space variable is assumed to be measured using an external sensor (e.g. a camera system or laser tracking could be used). To simplify the experiment, the task-space measurements were simulated by using the known kinematics of the robot (i.e., we artificially generate the task-space position measurements using the known forward kinematics). This kinematic information is used only to artificially generate the task-space signals and is not used to generate any other signals in the control algorithm.*

The approximated Jacobian matrix which is used in the control implementation is defined as follows:

$$\hat{J}_p = \begin{bmatrix} -\hat{l}_1 \sin(\theta_1) - \hat{l}_2 \sin(\theta_1 + \theta_2) & -\hat{l}_2 \sin(\theta_1 + \theta_2) \\ \hat{l}_1 \cos(\theta_1) + \hat{l}_2 \cos(\theta_1 + \theta_2) & \hat{l}_2 \cos(\theta_1 + \theta_2) \end{bmatrix}$$

where $\hat{J}_p \in \mathbb{R}^{2\times 2}$, $\hat{l}_1$ and $\hat{l}_2$ are estimates for the link lengths. The parameter vector $\hat{\phi}_j \in \mathbb{R}^2$ is defined as:

$$\hat{\phi}_j = \begin{bmatrix} \hat{l}_1 & \hat{l}_2 \end{bmatrix}^T$$

and the estimates were initialized to $\hat{l}_1(0) = 0.42$ [m] and $\hat{l}_2(0) = 0.22$ [m].

The true link lengths are $l_1 = 0.558$ [m] and $l_2 = 0.291$ [m]. We initialized the link length estimates to 75% of the true value. In cases where there is no information available about the link lengths, a best-guess could be used as an initial estimate of the link lengths.

The desired trajectory was defined as:

$$p_d = \begin{bmatrix} 0.55 + 0.2\cos(2t) \\ 0.25 + 0.2\sin(2t) \end{bmatrix}$$

The initial position of the joints were, $\theta_1(0) = 3.3°$, $\theta_2(0) = 45.1°$, which corresponds to $x(0) = 0.75$ [m], $y(0) = 0.25$ [m] in the task-space. The control gains that yielded the best tracking performance were as follows:

$$k_1 = diag\{2.5, 2.0\}, \quad k_r = diag\{80, 40\}$$

$$\Gamma_1 = diag\{8, 1\}, \quad \Gamma_2 = diag\{20, 45, 10, 500, 1, 3, 8, 15, 20, 5, 500, 25, 20, 20\}$$

Remark 5 *In this planar two degree of freedom example, there was no rotational error $e_v(t)$; hence, the gain k_2 is not used.*

Fig. 2.8 shows the actual and desired task space trajectories for the last revolution. Fig. 2.9 shows the position tracking error, it is seen that within 10 seconds the tracking error converges to approximately ± 2 [mm]. Fig. 2.10 and 2.11 show the link length estimates and the dynamic parameter estimates respectively. From Fig. 2.9, 2.10 and 2.11, it can be clear that the tracking error converges as the kinematic and dynamic parameters converge. Fig. 2.12 shows the control input torques to the two links of the Barrett WAM.

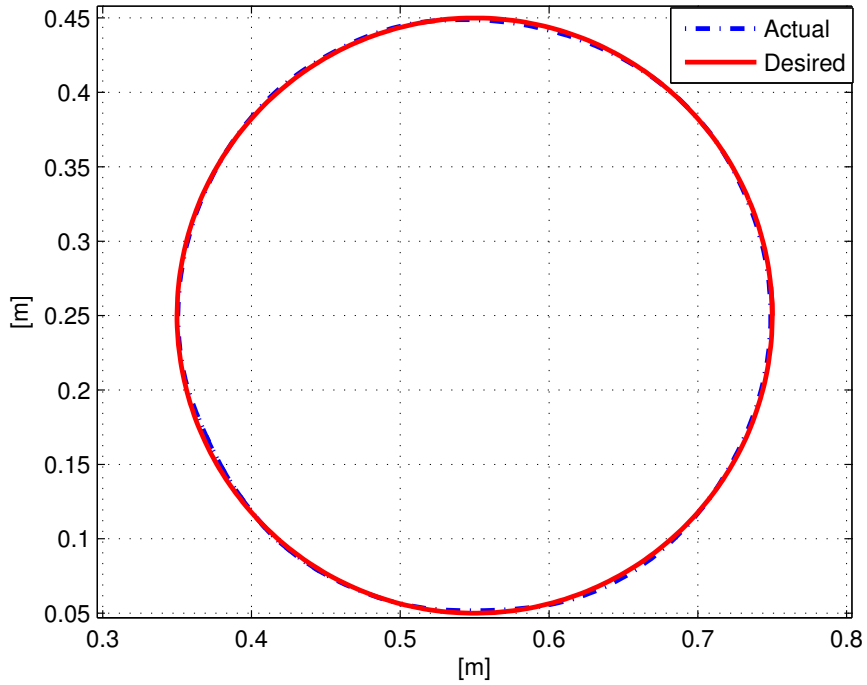

Figure 2.8 Actual and desired end-effector trajectory (only the last revolution is shown).

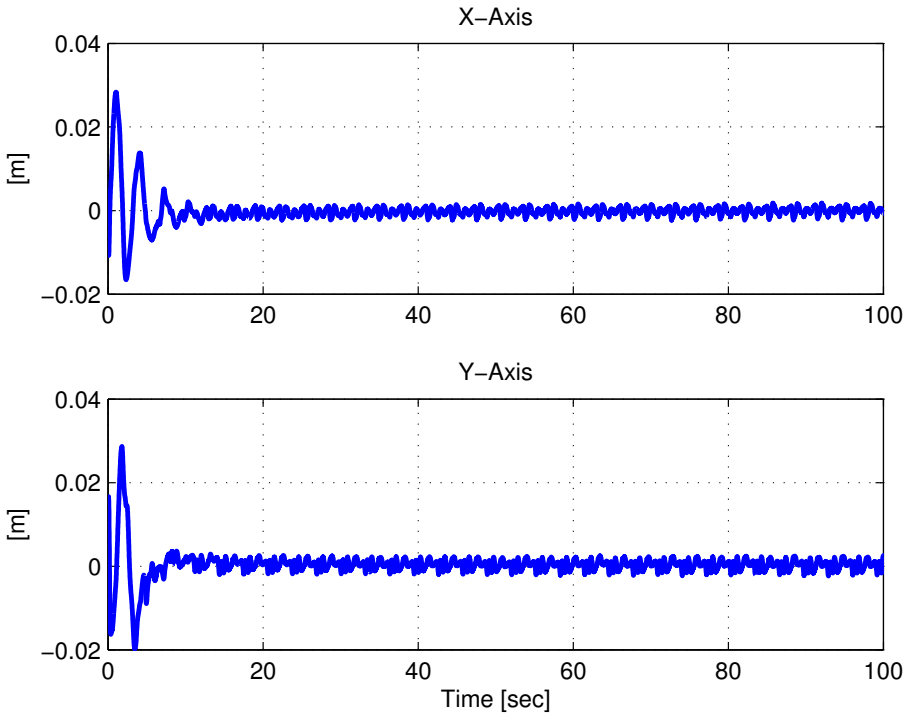

Figure 2.9 End-effector position tracking error.

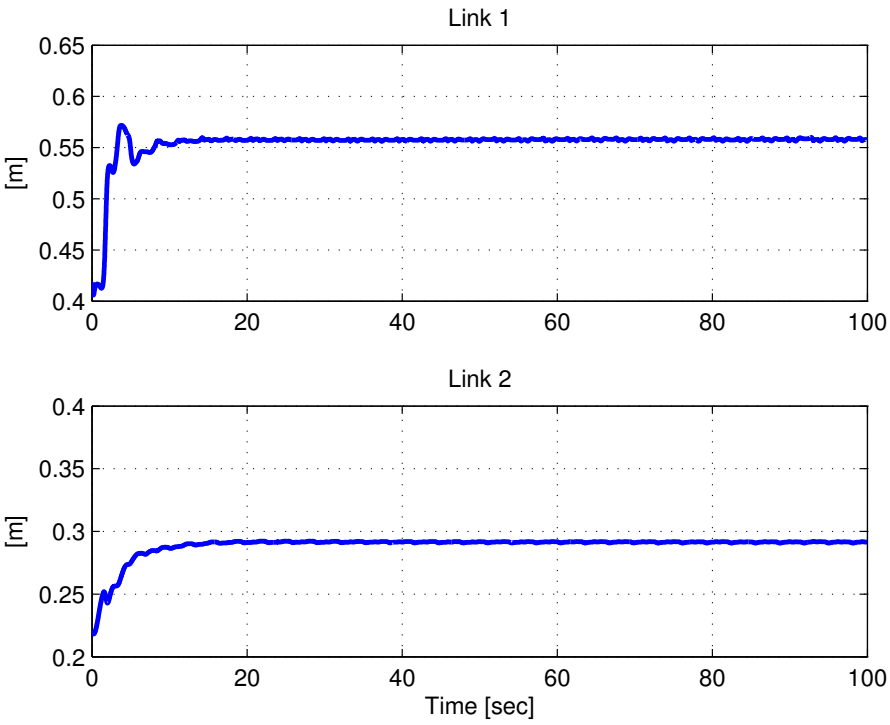

Figure 2.10 Link length estimates.

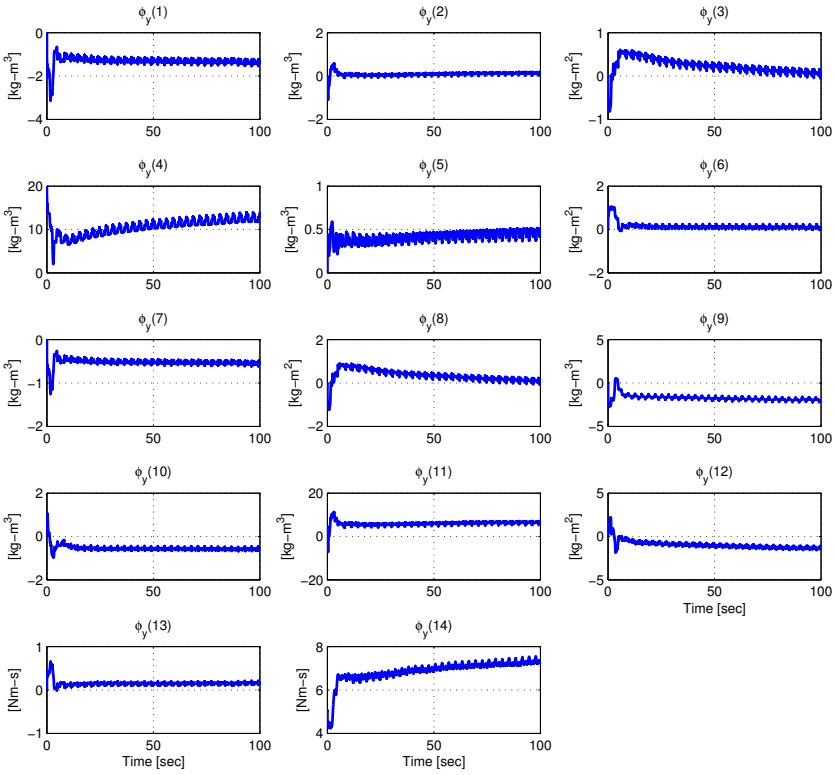

Figure 2.11 Dynamic parameter estimates.

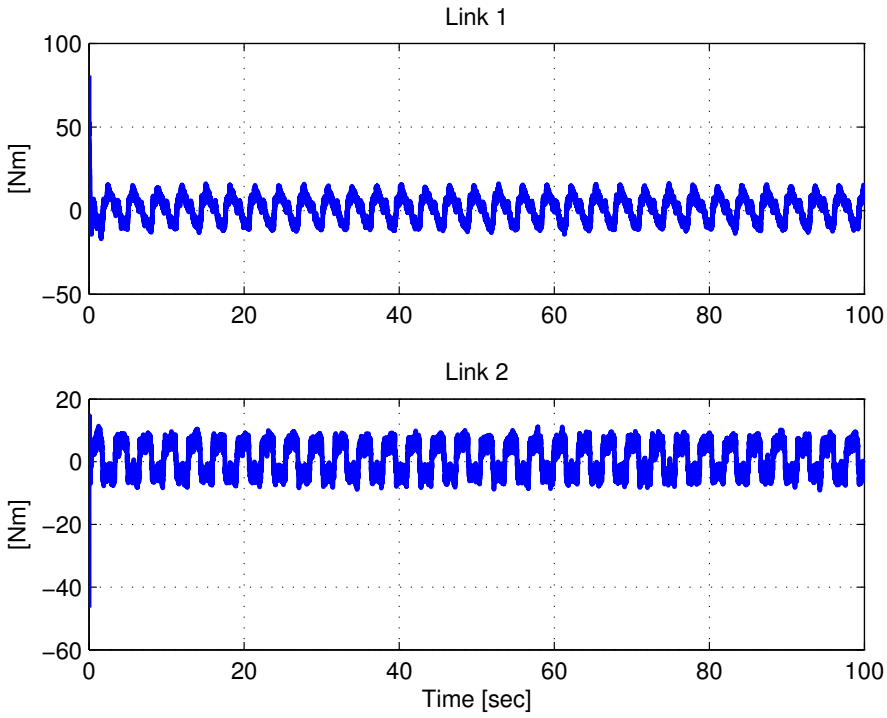

Figure 2.12 Control input torques.

CHAPTER 3

A NEURAL NETWORK CONTROLLER FOR CONTINUUM ROBOTS

In this chapter a neural network based tracking controller is developed for a class of hyper-redundant robot manipulators, also known as "continuum" robot manipulators. The novelty of this work is the development of an efficient tracking controller for extensible continuum manipulators which can deal with a high level of uncertainty in the structure of the manipulators dynamic model. The proposed controller consists of a neural network feedforward component along with a nonlinear feedback component. Specifically, the design of the neural network component is based on the augmented back propagation algorithm [33], and it is used to compensate for the nonlinear uncertain dynamics of the continuum robot manipulator by leveraging the universal approximation properties of the neural network as a feedforward compensator. The feedback component utilized is a continuous nonlinear controller [40], which does not require any model information. The advantages of the proposed control scheme compared to the previously mentioned works is that the controller is continuous and asymptotic tracking can be proved without any prior knowledge of the robot dynamic model. Furthermore, the back propagation technique enables the neural network weight matrices to be estimated on-line very quickly based on the tracking error signal and without utilizing any prior training period. This technique is particularly useful for real time closed loop control.

This chapter is organized as follows, in the first section, the sensing and actuation of the OCTARM VI, which is a soft extensible continuum robot manipulator (see Fig. 3.1), are briefly discussed. Then some of the prior efforts to develop dynamic models for continuum robot manipulators are discussed and a model for the extensible continuum manipulator is presented. In the next section the control objective is explicitly defined and the design of a controller with the neural network feedforward component is presented. To demonstrate the performance of the proposed controller with the neural network feedforward component, the controller was tested on the OC-

TARM. In the experimental results section, the performance of the controller without the neural network feedforward component is compared with that of the controller with the neural network feedforward component to illustrate the effectiveness of the proposed strategy.

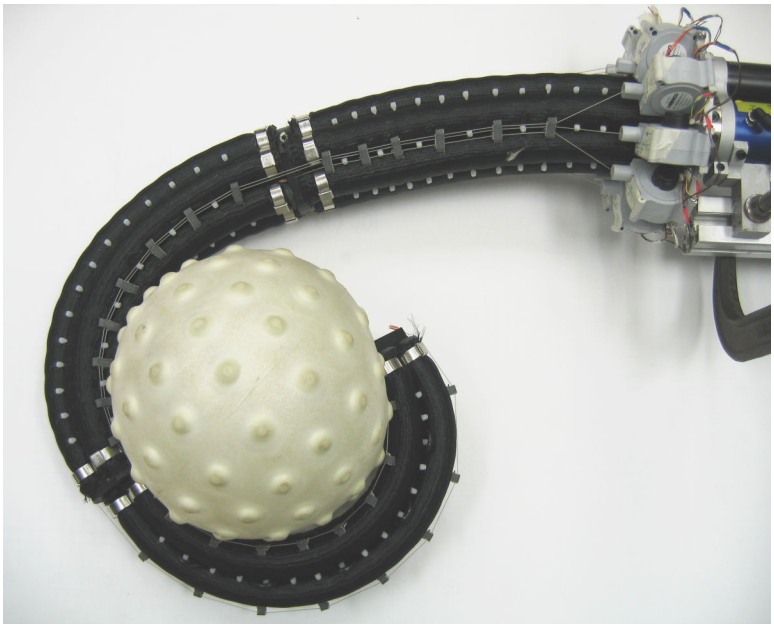

Figure 3.1 OCTARM VI continuum manipulator grasping a ball in Clemson University's robotics laboratory.

Robot Sensing and Actuation

The OCTARM VI manipulator [19, 24, 66], is a biologically inspired soft continuum manipulator resembling a trunk or tentacle. The OCTARM is significantly more versatile and adaptable than conventional robotic manipulators, capable of adaptive and dynamic manipulation in unstructured environments. To provide the desired dexterity the OCTARM VI is constructed with high strain extensor air muscles called McKibben actuators. These actuators are constructed by covering latex tubing with a double helical weave, plastic mesh sheath [67]. These actuators provide a large strength to weight ratio and strain which are required for soft continuum manipulators.

The OCTARM VI (refer to Fig. 3.1) is divided into three sections, with each section consisting of three McKibben actuators. Each section is capable of two axis bending and extension hence allowing nine total degrees of freedom for the manipulator. The manipulator is pneumatically actuated through nine pressure regulators which maintain the pressure in the actuators at a desired value, set using an input voltage. The pressure regulators provide a linear relationship between the control voltage and the air pressure. By varying the air pressure to the actuators on a section the length and shape of the section can be controlled.

The shape of the manipulator can be inferred in terms of curvatures and extensions by measuring the length of each of the nine actuators and using the forward kinematics described in [68]. In this work we are only concerned with developing an efficient low level controller which regulates the length of each of the actuators on the manipulator to follow a desired trajectory[4].

Robot Dynamic Model

From a review of current literature, it is evident that the theory related to the dynamic modeling of continuum robot arms is still in its nascence with few published works available. Some of the previous research includes [69–71], where planar models of the continuum structure were considered, and [72, 73], where the authors develop a three dimensional dynamic model for an inextensible (constant length) continuum manipulator. As such, the complete dynamic modeling of variable length continuum robot arms remains an open research area. In [72], the developed dynamic model was shown to satisfy the familiar property that the continuum manipulators inertia matrix is symmetric and positive definite. With this in mind, in the development being considered we will assume that the dynamic model of a 9 DOF (degree of freedom) continuum robot manipulator can be described by the following expression.

$$M(q)\ddot{q} + N(q, \dot{q}) = u \tag{3.1}$$

[4]Note that the desired trajectory for the lengths of each actuator could be specified directly in terms of the lengths or could be generated using the inverse kinematics described in [68].

where $M(q) \in \mathbb{R}^{9\times 9}$ represents the inertia matrix, $N(q,\dot{q}) \in \mathbb{R}^9$ represents the remaining dynamic terms such as centripetal, Coriolis and frictional forces , $u(t) \in \mathbb{R}^9$ represents the control input vector, and $q(t)$, $\dot{q}(t)$, $\ddot{q}(t) \in \mathbb{R}^9$ represent the actuator length, velocity and acceleration respectively.

The subsequent development is based on the following assumptions

Assumption 1 *The manipulators position $q(t)$ and velocity $\dot{q}(t)$ are measurable.*

Assumption 2 *The dynamic terms denoted by $M(q)$ and $N(q,\dot{q})$ are unknown nonlinear functions of $q(t)$ and $\dot{q}(t)$ which are second order differentiable and satisfy the following properties*

$$M(\cdot), \dot{M}(\cdot), \ddot{M}(\cdot) \in \mathcal{L}_\infty \quad \textit{if } q(t), \dot{q}(t), \ddot{q}(t) \in \mathcal{L}_\infty \tag{3.2}$$

$$N(\cdot), \dot{N}(\cdot), \ddot{N}(\cdot) \in \mathcal{L}_\infty \quad \textit{if } q(t), \dot{q}(t), \ddot{q}(t), \dddot{q}(t) \in \mathcal{L}_\infty. \tag{3.3}$$

Assumption 3 *The inertia matrix $M(q)$ is symmetric and positive-definite, and satisfies the following inequalities*

$$m_1 \left\|\xi\right\|^2 \leq \xi^T M(q) \xi \leq m_2 \left\|\xi\right\|^2 \qquad \forall \xi \in \mathbb{R}^9 \tag{3.4}$$

where m_1, $m_2 \in \mathbb{R}$ are positive constants, and $\|\cdot\|$ denotes the standard Euclidean norm.

Control design

As we have mentioned in the previous section, the dynamic modeling of extensible continuum manipulators remains an open research problem. The main focus of the control design in this section is to develop an efficient tracking controller which can deal with the high level of uncertainty in the structure of continuum robot's dynamic model. The proposed control strategy consists of a neural network feedforward component and a nonlinear feedback component. The feedback component is based on prior work in [40], this controller is chosen because it leaves a lot of flexibility in the

design of the neural network feedforward component. The neural network feedforward component is then designed based on the back propagation algorithm in [33] to meet the boundedness requirements required by the feedback controller.

Feedback Controller

The control objective is to design a continuous controller which provides asymptotic tracking of the actuator lengths and the desired actuator length trajectories in the sense that

$$q(t) \rightarrow q_d(t) \text{ as } t \rightarrow \infty. \tag{3.5}$$

To quantify the control objective, an error signal, denoted by $e_1(t) \in \mathbb{R}^9$, is defined as follows

$$e_1 \triangleq q_d - q. \tag{3.6}$$

Furthermore, an auxiliary tracking error signal $e_2(t) \in \mathbb{R}^9$ is defined as follows

$$e_2 \triangleq \dot{e}_1 + \lambda_1 e_1 \tag{3.7}$$

where $\lambda_1 \in \mathbb{R}^+$ is a control gain. For the closed loop error system development, we define a filtered tracking error signal $r(t) \in \mathbb{R}^9$ as follows

$$r \triangleq \dot{e}_2 + \lambda_2 e_2 \tag{3.8}$$

where $\lambda_2 \in \mathbb{R}^+$ is a control gain.

The dynamic model of the continuum robot is highly nonlinear and has an uncertain structure; hence, the strategy developed by Xian *et al.* [40], can be utilized for the controller. This controller is chosen because it is continuous, it does not require the dynamic model of the manipulator to be known and yet it provides semi-global asymptotic tracking. Specifically, the control objective described in (3.5) can be met with the following controller [40]

$$\begin{aligned} u(t) \triangleq\ & (K_s + I)e_2(t) - (K_s + I)e_2(t_0) + \int_{t_0}^{t} \hat{f}(\tau)d\tau \\ & + \int_{t_0}^{t} \left(\lambda_2(K_s + I)e_2(\tau) + \beta sgn(e_2(\tau))\right) d\tau \end{aligned} \tag{3.9}$$

where $u(t) \in \mathbb{R}^9$ is the control input defined in (3.1), $\lambda_2 \in \mathbb{R}^+$ is a control gain, K_s, $\beta \in \mathbb{R}^{9\times 9}$ are positive definite diagonal control gain matrices, $\hat{f}(t) \in \mathbb{R}^9$ is the neural network feedforward component, and $sgn(\cdot) : \mathbb{R}^9 \mapsto \mathbb{R}^9$ denotes the vector signum function defined as $sgn(\xi) = [sgn(\xi_1), \cdots, sgn(\xi_9)]^T \ \forall \xi = [\xi_1, \cdots, \xi_9]^T \in \mathbb{R}^9$. As shown in [40], the controller presented in (3.9), provides semi-global asymptotic convergence of the actuator length tracking error, (i.e. $\|e_1(t)\| \to 0$ as $t \to \infty$). For brevity in this presentation, the stability analysis is omitted, for a complete analysis of the controller the reader is referred to [40].

Remark 1 *The design of the neural network feedforward component, $\hat{f}(t)$, is presented in the subsequent section. The only restriction imposed on the neural network component by the selection of the feedback controller in (3.9) is that $\hat{f}(t) \in \mathcal{L}_\infty$, i.e. the output from the neural network must be bounded.*

Neural Network Feedforward Design

The neural network feedforward component $\hat{f}(t) \in R^9$ is computed using a two layer network with 15 neurons. The weights are computed using a modified version of the back propagation algorithm presented in [33]. Given Remark 1, an important consideration regarding the design of the neural network feedforward component is that the output from the neural network must always be bounded (i.e. $\hat{f}(t) \in \mathcal{L}_\infty$). To this end the neural network component is defined as follows

$$\hat{f} = \hat{W}^T \bar{\sigma}\left(\hat{V}^T x\right). \tag{3.10}$$

where $\hat{W}(t) \in \mathbb{R}^{15\times 9}$ and $\hat{V}(t) \in \mathbb{R}^{37\times 15}$ are estimated weight matrices, and $x(t) \in \mathbb{R}^{37}$ is the input vector to the neural network which is selected as

$$x = \left[\ 1, \ \ q_d^T, \ \ \dot{q}_d^T, \ \ \ddot{q}_d^T, \ \ \dddot{q}_d^T \ \right]^T \tag{3.11}$$

where $q_d(t), \dot{q}_d(t), \ddot{q}_d(t), \dddot{q}_d(t)$ were defined *a priori*. The vector activation function $\bar{\sigma}(\cdot) \in \mathbb{R}^{15} \mapsto \mathbb{R}^{15}$ is defined as follows

$$\bar{\sigma}(\omega) = [\sigma(\omega_1), \sigma(\omega_2), \cdots, \sigma(\omega_{15})]^T \tag{3.12}$$

where $\omega = [\omega_1, \omega_2, \cdots, \omega_{15}]^T$ and $\sigma(s) : \mathbb{R} \mapsto \mathbb{R}$ is the sigmoid activation function defined as

$$\sigma(s) = \frac{1}{1 + \exp(-s)} \ . \tag{3.13}$$

The gradient of the vector activation function, denoted by $\bar{\sigma}'(\cdot) \in \mathbb{R}^{15\times 15}$ can be expressed in closed form as follows, [33]

$$\bar{\sigma}(\omega)' = diag\{\bar{\sigma}(\omega)\}\left[I - diag\{\bar{\sigma}(\omega)\}\right]. \tag{3.14}$$

If we were to design the weight update laws according to the augmented backpropagation algorithm [33], we would use the following update rules

$$\dot{\hat{W}} = -\kappa F \left\| r \right\| \hat{W} - F\bar{\sigma}'(\cdot)\hat{V}^T x r^T + F\bar{\sigma}(\cdot) r^T$$

$$\dot{\hat{V}} = -\kappa G \left\| r \right\| \hat{W} + Gx\left(\bar{\sigma}'^T(\cdot)\hat{W} r\right)^T$$

where $\kappa \in \mathbb{R}^+$ is selected to be a small constant, $F \in \mathbb{R}^{15\times 15}$, $G \in \mathbb{R}^{37\times 37}$ are positive definite gain matrices, $x(t)$ is the input vector defined in (3.11), and $r(t)$ is the filtered tracking error signal defined in (3.8). Here, the filtered tracking error signal $r(t)$ is required in the update laws which requires the measurement of the acceleration $\ddot{q}(t)$, and hence, is undesirable. To ensure that the weights generated from these laws are bounded, and that acceleration measurements are not required, we redefine the update laws as follows

$$\dot{\hat{W}} = -\alpha_w \hat{W} + \gamma_1 \bar{\sigma}\left(\hat{V}^T x\right) \ \mathrm{sat}\left(e_2 + \zeta\right)^T \tag{3.15}$$

$$\dot{\hat{V}} = -\alpha_v \hat{V} + \gamma_2 x\left[\bar{\sigma}'\left(\hat{V}^T x\right)\hat{W}\mathrm{sat}\left(e_2 + \zeta\right)\right]^T \tag{3.16}$$

where α_v, $\alpha_w \in \mathbb{R}^+$ are small constants, γ_1, $\gamma_2 \in \mathbb{R}^+$ are control gains which effect the learning speed, the function $\mathrm{sat}(\xi) : \mathbb{R}^9 \mapsto \mathbb{R}^9$ is defined as $\mathrm{sat}(\xi) = [\mathrm{sat}(\xi_1), \cdots, \mathrm{sat}(\xi_9)]^T \ \forall \xi = [\xi_1, \cdots, \xi_9]^T \in \mathbb{R}^9$ where $\mathrm{sat}(\xi_i) \in \mathbb{R} \ \forall i = 1, \cdots, 9$ is the following saturation function

$$\mathrm{sat}(\xi_i) = \begin{cases} -\xi_{min} & \text{if} \quad \xi_i \leq -\xi_{min} \\ \xi_i & \text{if} \quad \xi_i > -\xi_{min} \quad \text{or} \quad \xi_i < \xi_{max} \\ \xi_{max} & \text{if} \quad \xi_i \geq \xi_{max} \end{cases}$$

where $\xi_{min}, \xi_{max} \in \mathbb{R}^+$ are constants. In (3.15) and (3.16) the auxiliary signal $\zeta(t) \in \mathbb{R}^9$ is an approximation (i.e. a dirty derivative operation) for the signal $\dot{e}_2(t)$ which is defined as follows

$$\zeta = \frac{1}{\varepsilon}(e_2 - \eta) \tag{3.17}$$

where $\varepsilon \in \mathbb{R}^+$ is a small constant, and the signal $\eta(t) \in \mathbb{R}^9$ is updated according to the following expression

$$\dot{\eta} = \frac{1}{\varepsilon}(e_2 - \eta). \tag{3.18}$$

From equations (3.10)-(3.18) and the fact that the input vector to the neural network is bounded, it is easy to show that the weight matrices $\hat{W}(t)$ and $\hat{V}(t)$ are bounded, and hence, the output from the neural network, $\hat{f}(t)$, is bounded.

Experimental Results

To verify the performance of the controller with the neural network feedforward component, the controller was implemented on the OCTARM VI continuum robot manipulator. In this section, we first provide a description of the control system for the OCTARM VI continuum robot manipulator, then experimental results are described which illustrate the effectiveness of the neural network feedforward tracking controller.

OCTARM Control System

Figure 3.2 shows an overview of the control system of the OCTARM. The control system consists of a commercial off-the-shelf Pentium III EBX form-factor Single Board Computer (SBC) with two ServoToGo data acquisition boards which provide analog and digital I/O. The computer runs the QNX Neutrino real-time operating system and QMotor 3.0 [65], a real-time control software which facilitates online parameter tuning and data logging for the implemented control programs. Data acquisition and closed loop control were performed at a frequency of 500 Hz. There are nine pressure regulators, one for each actuator on the manipulator. The air pressure

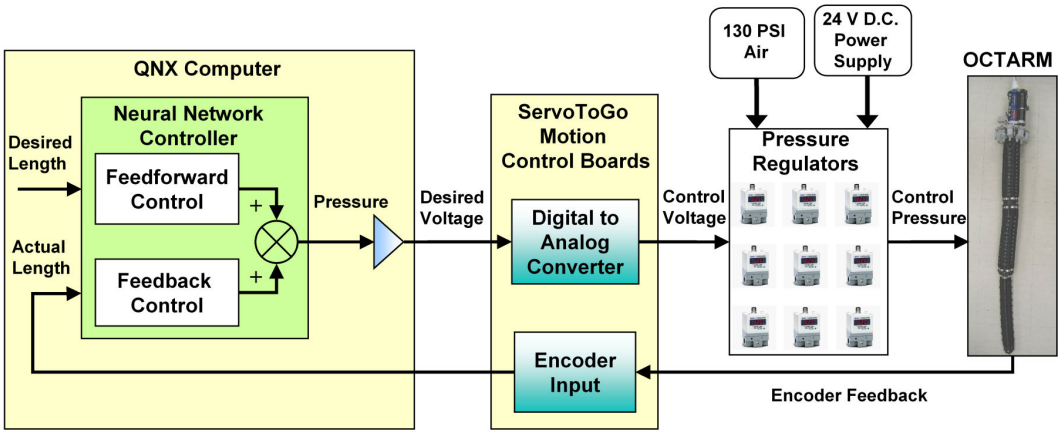

Figure 3.2 Block diagram showing an overview of the OCTARM VI control system.

for each actuator is determined by the neural network controller. The pressure specified by the controller is converted to a corresponding voltage level (using a linear relationship between pressure and voltage) and this drives the pressure regulators which control the air flow to the actuators.

For closed loop control of the OCTARM VI manipulator accurate sensing of the actuator lengths is essential. To measure the length of each of the nine actuators, there are nine string encoders arranged around the base of section one (see Fig. 3.3). The cables from each of the string encoders run the entire length of the actuator they are assigned to measure. This configuration enables the length of each of the actuators on the OCTARM VI manipulator to be determined. Since the string encoders have a relatively low resolution, velocities obtained through differentiation of the position measurements are noisy; hence, a variable structure velocity observer [40], was utilized to obtain estimates of the velocity.

Trajectory Tracking Experiment Description

To test the low level controller with the neural network component given in (3.9), a sinusoidal trajectory was selected for the actuator lengths. Since, the three actuators on a section are 120 degrees apart mechanically, the desired trajectory for each

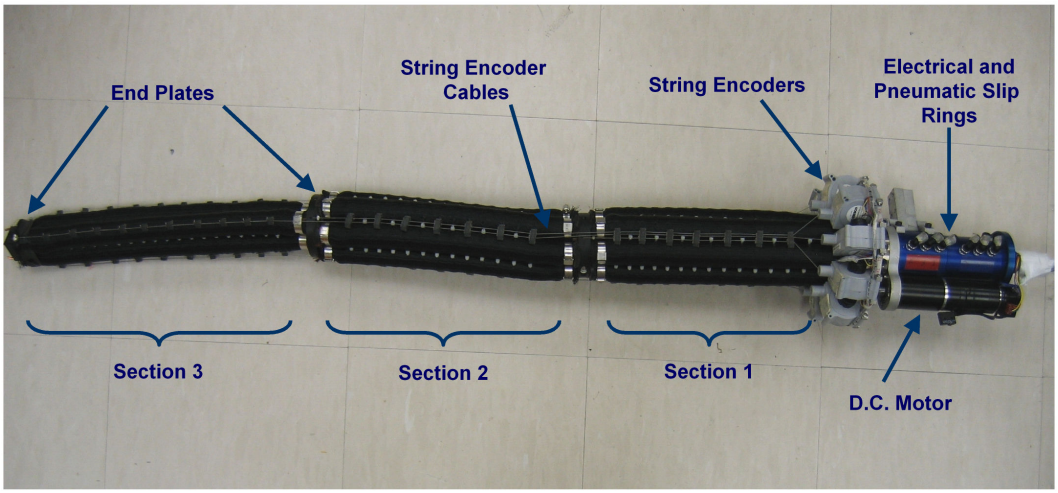

Figure 3.3 The OCTARM VI robotic manipulator.

actuator in a section is shifted 120 degrees in phase from the trajectory of the previous actuator in that section. The trajectory for section one was selected as follows

$$\begin{aligned} q_{d1k} &= l_{min_1} + (1 - \exp(-0.5t))l_1 \\ &\quad +3(1 - \exp(-0.5t))\sin\left(0.03\pi t + \frac{2}{3}\pi k\right) \end{aligned}$$

where $k = 1, 2, 3$, $q_{d1} = [q_{d11}, q_{d12}, q_{d13}] \in \mathbb{R}^3$ represents the desired trajectory for the actuators on section one, $l_{min_1} \in \mathbb{R}$ represents the minimum lengths of the actuators on section one, and $l_1 = 8$ [cm] is the initial extension of the actuators on section one. The initial extension was selected to keep the operating pressure close to its nominal value. For sections two and three, the initial extensions were $l_2 = 7$ [cm], $l_3 = 5$ [cm] and the frequency of the sinusoidal trajectory for each section was twice that of the previous section. The minimum and maximum lengths of the sections corresponding to the minimum pressure (0 psi) and maximum pressure (130 psi) respectively are physical limitations of the actuators and were found to be $l_{min_1} = 28$ [cm], $l_{min_2} = 26.5$ [cm], $l_{min_3} = 32.5$ [cm], $l_{max_1} = 42$ [cm], $l_{max_2} = 44$ [cm], $l_{max_3} = 54$ [cm].

Control Parameter Tuning

To test the effectiveness of the neural network feedforward component, we compared the performance of the controller in (3.9), with and without the neural network

component. The performance of the system was first tested without the neural network component. The control gains for the feedback controller were adjusted in such a manner as to obtain a fast system response while minimizing the overshoot and oscillations, since the actuators have very low damping. The control gains which gave the best performance were as follows

$$K_s = diag\{1,1,1,1,1,1,1,1,1\}, \;\; \lambda_1 = 1, \;\; \lambda_2 = 1,$$
$$\beta = diag\{1,1,1,1,1,1,0.5,0.5,0.5\}.$$

It is interesting to note that when β and $\hat{f}(t)$ in (3.9) are zero, the control law is essentially a PID controller and hence, the parameters $K_s, \lambda_1, \lambda_2$, can be tuned in a conceptually similar manner to a PID controller. However, the addition of the control gain β into the controller provides a significant improvement in the tracking performance. Also, the last three elements of the β vector, corresponding to the third section of the manipulator, are smaller, since the third section requires less control effort to extend due to its relatively smaller size.

With the feedback controller now tuned, the neural network feedforward was added to the controller while keeping the gains on the feedback controller the same. The number of neurons required for the system was determined experimentally by noting the performance achievable with a given number of neurons and then increasing the number of neurons until the desired tracking performance level was obtained. Initially 5 neurons were utilized and this was subsequently increased to 15 to get better performance. Note that we cannot keep increasing the number of neurons since we require the control algorithm to be as computationally efficient as possible for it to run in real-time on the SBC. There was no off-line training period utilized for the neural network feedforward component, the weight matrices $\hat{W}(t)$ and $\hat{V}(t)$ in the neural network were initialized to zero. The gains in the neural network weight update law were adjusted to be as follows

$$\gamma_1 = 10, \;\; \gamma_2 = 500, \;\; \alpha_w = 0.001, \;\; \alpha_v = 0.001.$$

The constants α_w and α_v (sometimes called forgetting factors) were kept small to provide stability to the weight estimation laws. The constants γ_1 and γ_2 affect the weight adaptation speed. It was interesting to note that γ_2 was required to be much larger than γ_1 to provide stable yet fast convergence of the weights. This is because the gain γ_2 only affects the inner layer of the neural network, whereas γ_1 would affect the output layer. The constant ε must be close to zero and was selected as follows $\varepsilon = 0.01$.

Experiment Results and Discussion

To provide a means to quantify the performance of the controller for each configuration, we computed the following measures

$$M_e \triangleq \int_{t_0}^{t} \|e_1(\tau)\|^2 d\tau \tag{3.19}$$

$$M_u \triangleq \int_{t_0}^{t} \|u(\tau)\|^2 d\tau \tag{3.20}$$

where $M_u(t)$ is a measure of the energy expended by the controller, and $M_e(t)$ is a measure of the magnitude of the tracking error over the period of operation of the system.

Figures 3.4, 3.5 and 3.6, show the actual and desired length trajectories, tracking error, and the input pressure, for the controller without the neural network component. Figures 3.7, 3.8 and 3.9, show the actual and desired length trajectories, tracking error, and the input pressure, for the controller with the neural network feedforward component. It can be seen from Figure 3.8, that the tracking error with the neural network feedforward component settles out to ± 0.3 [cm] in under 5 seconds.

To compare the controller performance with and without the neural network feedforward component, the measures (3.19), (3.20), were computed for the two configurations. Tables 3.1, 3.2, 3.3, show a comparison of the performance of these two controller configurations over different time periods of the experimental trajectory. It can clearly be seen from these tables that although the control input measure remains

almost the same over the different time periods, the error measure shows a significant drop when the neural network feedforward component is introduced. Also, note that as the neural network feedforward continues to "learn" the performance of the controller improves. From the tables we note that after 5 seconds there is a 38.66% improvement, at 10 seconds there is a 66.36% improvement, and after 60 seconds the performance improvement is 87.82%. Thus we can conclude that improved tracking performance is achieved by adding the neural network feedforward to the controller.

Table 3.1 Performance measures for the controller with and without the feedforward component calculated for the first 5 seconds of the experimental trajectory

	Without feedforward	With feedforward
M_e	13.14042	8.06437
M_u	$54,808.85$	$52,331.17$

Table 3.2 Performance measures for the controller with and without the feedforward component calculated for the first 10 seconds of the experimental trajectory

	Without feedforward	With feedforward
M_e	27.32518	9.19339
M_u	$148,143.52$	$138,589.19$

Table 3.3 Performance measures for the controller with and without the feedforward component calculated for the first 60 seconds of the experimental trajectory

	Without feedforward	With feedforward
M_e	145.51935	17.7180
M_u	1.0463×10^6	1.0301×10^6

The controller described in this chapter has been implemented on the OCTARM VI control system replacing the original PID controller. The group has adapted the controller as the default controller for the OCTARM series of robots. It has been utilized for a number of applications such as whole arm grasping, inspection

and tunnel navigation where it has shown excellent performance. All of these tasks were performed by an operator commanding the continuum arm's motion using a joystick interface (see [74], for a description of the operator interface). The tunnel navigation task is one application where the improved tracking performance provided by the neural network controller significantly reduces the time taken by the operator to navigate down the tunnel. Through our experimentation we have noted that the fine motion control possible with the controller is extremely useful for the tasks mentioned above as it enables all the capabilities of the continuum arm to be explored and exploited.

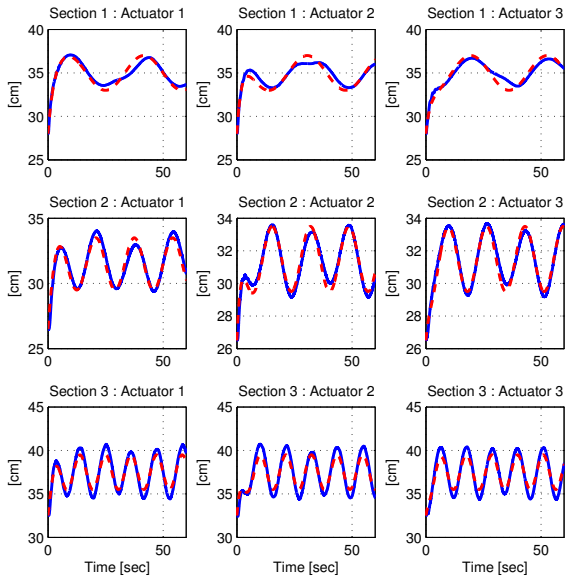

Figure 3.4 Actual and desired length trajectory without neural network component, solid line represents the actual length trajectory, dashed line represents the desired length trajectory.

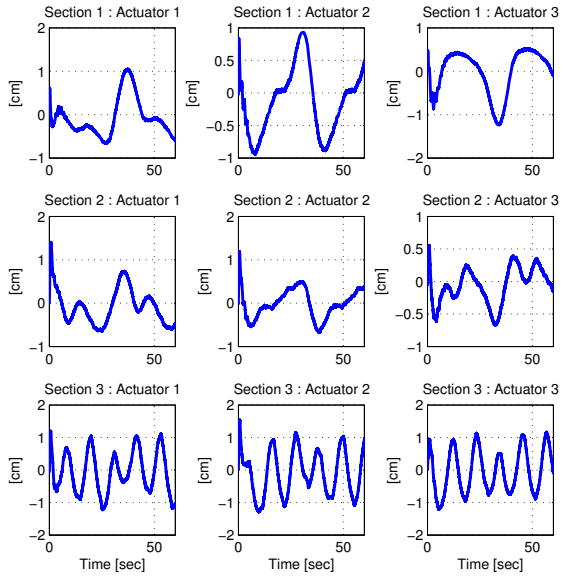

Figure 3.5 Tracking error without neural network component.

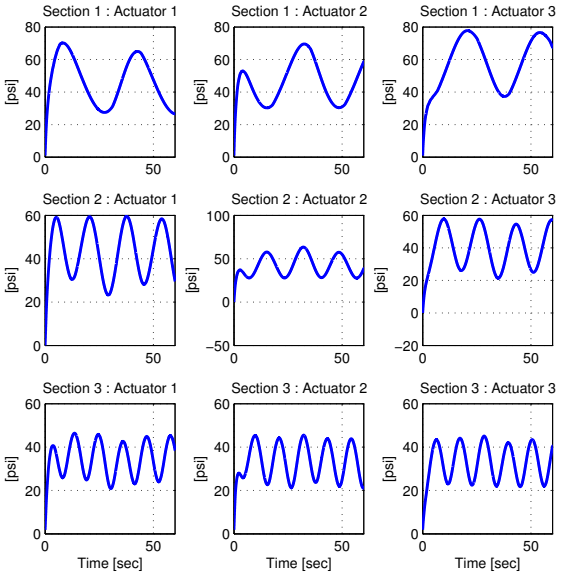

Figure 3.6 Control pressure without neural network component.

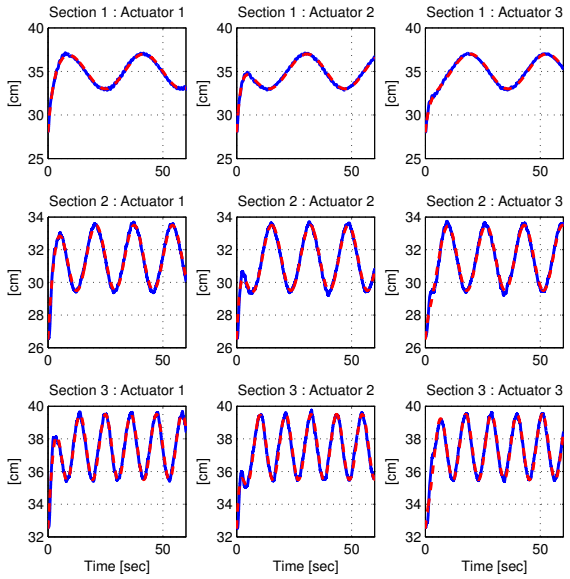

Figure 3.7 Actual and desired length trajectory with neural network component, solid line represents the actual length trajectory, dashed line represents the desired length trajectory.

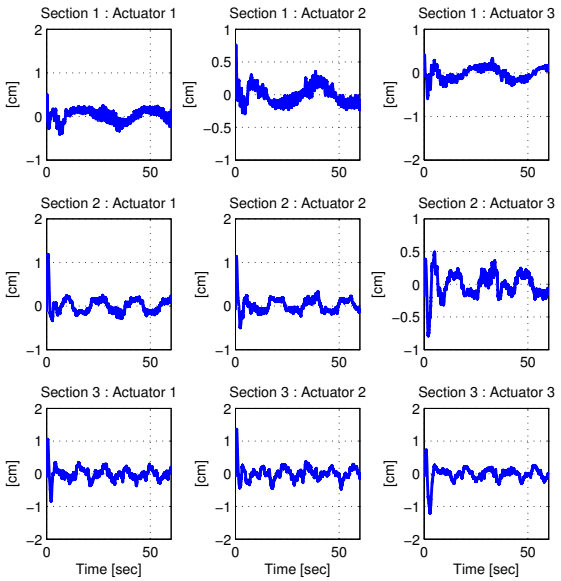

Figure 3.8 Tracking error with neural network component.

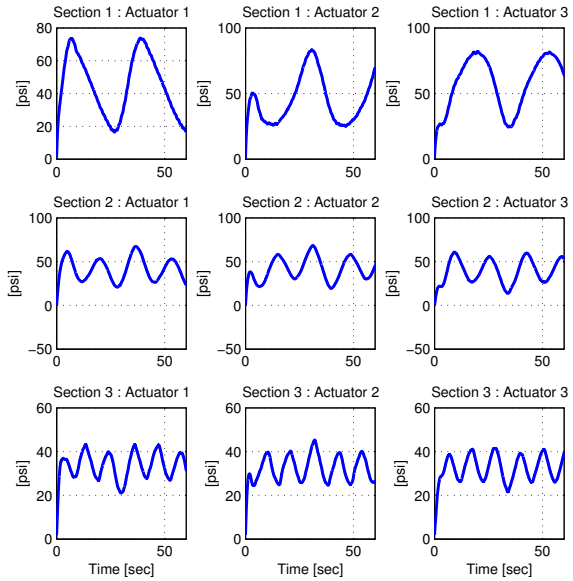

Figure 3.9 Control pressure with neural network component.

CHAPTER 4
WHOLE ARM GRASPING CONTROL FOR REDUNDANT ROBOT MANIPULATORS

An application which is well suited for kinematically redundant robot manipulators is the utilization of the manipulators body to grasp objects, also known as whole arm grasping. In this chapter an approach to whole arm grasping of rigid objects using kinematically redundant robot manipulators is presented. Roughly, the whole arm grasping objective is achieved by integrating the path planner and the controller such that two tasks, end-effector positioning and body self-motion positioning, are accomplished simultaneously. The end-effector positioning controller forces the end-effector to follow a path around the object which in turn, forces the robot's body to wrap itself around the object to be grasped. The body self-motion positioning controller "repels" the body of the manipulator away from the object while the end-effector moves around the object. This control-induced repulsion-like property facilitates object avoidance as well as removes the "slack" from the robot body as the robot begins to move into the grasping position. When all possible slack is removed, the manipulators body makes contact with the object, hence, completing the whole arm grasp of the object.

The whole arm grasping control problem can thus be broken down into two stages; first, a kinematic level path planner is designed which facilitates the encoding of both the end-effector position as well as the manipulators self-motion positioning information as a desired trajectory for the manipulator joints. The trajectory from the high level path planner is then filtered to produce an appropriate joint level trajectory. Then, a joint space controller, which is based on the work in Chapter 3, is utilized to provide asymptotic tracking of the encoded desired joint trajectory in the presence of dynamic uncertainties. Experimental results for a planar application of the whole arm grasping technique using the Barrett whole arm manipulator are described to illustrate the performance of the controller.

Manipulator Models

In this section the kinematic and dynamic models for an n-joint ($n \geq 6$), revolute, direct drive robot manipulator are presented. The subsequent development of the whole arm grasping controller is based on these models.

Kinematic Model

The Denavit-Hartenberg based forward kinematic model for an n-segment redundant manipulator can be developed as follows

$$x_n = f_n(q) \tag{4.1}$$

where $x_n(t) \in \mathbb{R}^p$ represents the robot end-effector's task-space vector, $q(t) \in \mathbb{R}^n$ denotes the joint position, and $f_n(q) \in \mathbb{R}^p$ denotes the forward kinematics of the manipulator. The velocity kinematics for the manipulator can be developed as follows

$$\dot{x}_n = J_n(q)\dot{q}(t) \tag{4.2}$$

where $\dot{x}_n(t) \in \mathbb{R}^p$ represents the task-space velocity, $\dot{q}(t) \in \mathbb{R}^n$ denotes the joint velocity, and $J_n(q) \triangleq \dfrac{\partial f_n(q)}{\partial q} \in \mathbb{R}^{p \times n}$ denotes the manipulator Jacobian.

Dynamic Model

The dynamic model for an n-joint ($n \geq 6$), revolute, direct drive robot manipulator is described by the following expression [57]

$$M(q)\ddot{q} + N(q, \dot{q}) + F_e(q, \dot{q}) = \tau \tag{4.3}$$

where $M(q) \in \mathbb{R}^{n \times n}$ represents the inertia effects, $N(q, \dot{q}) \in \mathbb{R}^n$ represents the remaining dynamic terms, such as the centripetal-Coriolis effects, gravitational effects, and frictional effects, $F_e(q, \dot{q}) \in \mathbb{R}^n$ represents the contact forces placed on the robot manipulator by the environment, $\tau(t) \in \mathbb{R}^n$ represents the input torque vector. The subsequent development is based on the assumptions that $q(t)$ and $\dot{q}(t)$ are measurable, $M(q)$, $N(q, \dot{q})$, and $F_e(q, \dot{q})$ are unknown, second order differentiable, functions of $q(t)$ and $\dot{q}(t)$, and the inertia matrix satisfies the following property [57],

Property 1 *The inertia matrix $M(q)$ is symmetric and positive-definite, and satisfies the following inequalities*

$$m_1 \|\xi\|^2 \leq \xi^T M(q)\xi \leq m_2 \|\xi\|^2 \qquad \forall \xi \in \mathbb{R}^n \tag{4.4}$$

where m_1, $m_2 \in \mathbb{R}$ are positive constants, and $\|\cdot\|$ denotes the standard Euclidean norm.

Remark 1 *Since this development is only concerned with revolute robot manipulators, the kinematic and dynamic terms denoted by $M(q)$, $N(q, \dot{q})$, and $J(q)$, are assumed to be bounded for all possible $q(t)$ (i.e., these kinematic and dynamic terms only depend on $q(t)$ as arguments of trigonometric functions).*

High Level Path Planning

As has been described previously, the whole arm grasping objective is achieved by designing the high level path planner such that two tasks, end-effector positioning and body self-motion positioning, are accomplished simultaneously. To facilitate the development of the path planner, the pseudo-inverse of the manipulator Jacobian, $J_n(q)$, denoted by $J_n^+(q) \in \mathbb{R}^{n \times p}$, is defined as follows

$$J_n^+ \triangleq J_n^T \left(J_n J_n^T\right)^{-1} \tag{4.5}$$

where $J_n^+(q)$ satisfies the following equality

$$J_n J_n^+ = I_p \tag{4.6}$$

where $I_p \in \mathbb{R}^{p \times p}$ is the standard identity matrix. The pseudo-inverse defined as in (4.5) satisfies the following properties [75]

$$\begin{array}{ll} J_n J_n^+ J_n = J_n & J_n^+ J_n J_n^+ = J_n^+ \\ \left(J_n^+ J_n\right)^T = J_n^+ J_n & \left(J_n J_n^+\right)^T = J_n J_n^+. \end{array} \tag{4.7}$$

In addition to the above properties, the matrix $(I_n - J_n^+ J_n)$ satisfies the following useful properties

$$\begin{gathered}(I_n - J_n^+ J_n)(I_n - J_n^+ J_n) = I_n - J_n^+ J_n \\ (I_n - J_n^+ J_n)^T = (I_n - J_n^+ J_n) \\ J_n (I_n - J_n^+ J_n) = 0 \\ (I_n - J_n^+ J_n) J_n^+ = 0\end{gathered} \tag{4.8}$$

where $I_n \in \mathbb{R}^{n \times n}$ is the standard identity matrix.

Remark 2 *During the control development, the assumption is made that the minimum singular value of the manipulator Jacobian, denoted by σ_m is greater than a known, small positive constant $\delta > 0$, such that $\max\{\|J_n^+(q)\|\}$ is known a priori and all kinematic singularities are always avoided.*

Based on (4.2) and the properties mentioned above, the kinematic level path planner, denoted by $U(t) \in \mathbb{R}^n$, which enables the whole arm grasping objective is designed as follows

$$\dot{q}(t) \triangleq J_n^+ U_e + \left(I_n - J_n^+ J_n\right) U_m \tag{4.9}$$

where $U_e(t) \in \mathbb{R}^p$ is the *end-effector positioning* controller, and $U_m(t) \in \mathbb{R}^n$ is the *robot body self-motion* controller. In the remainder of this section, the design of the robot *end-effector positioning* controller and the *robot body self-motion* controller will be discussed in detail.

End-Effector Positioning

The objective of the *end-effector positioning* controller is to force the end-effector to track a desired trajectory that encompasses the surface of the object to be grasped. For this type of problem, instead of a time based trajectory, a velocity field control (VFC) is utilized because it more effectively penalizes the end-effector for leaving the contour ([76], [77], and [78]). The VFC will also not exhibit the *radial reduction* phenomenon which is common with traditional control methods ([76] and [77]). For example, when the object to be grasped is circular, the velocity field generates a desired trajectory that forces the end-effector to spiral inwards, toward and around the surface of the object.

Remark 3 *A velocity field specifies a desired velocity $\dot{x}_d(t)$ at each displacement position $x_n(t)$ on the task space of the system [77]. In [77] and [78], the authors provide specific information about the construction of velocity fields. Also see [76] and [79] for details of circular velocity fields. The velocity field for a specific planar application is presented subsequently in the experimental section of this chapter.*

The *end-effector positioning* controller $U_e(t) \in \mathbb{R}^p$ which enables the path planning objective is developed based on a Lyapunov analysis and is defined as follows

$$U_e \triangleq \vartheta(x_n) + K_e e + k_n \left\| \frac{\partial V(x_d)}{\partial x_d} \right\|^2 \rho^2(x_n, x_d)\, e \tag{4.10}$$

where $\vartheta(x_n) \in \mathbb{R}^p$ is a task-space velocity field, $K_e \in \mathbb{R}^{p\times p}$ is a positive definite diagonal gain matrix, $k_n \in \mathbb{R}^+$ is a scalar gain parameter, $e(t) \in \mathbb{R}^p$ is the task space position tracking error defined as follows

$$e \triangleq x_d - x_n, \tag{4.11}$$

where $x_d(t) \in \mathbb{R}^p$ is the desired task-space position, and $x_n(t)$ was introduced in (4.1). In (4.10), $V(x_d) \in \mathbb{R}$ is a first order differentiable, nonnegative function and $\rho(\cdot) \in \mathbb{R}$ is a known positive function that is assumed to be bounded provided $x_n(t)$ and $x_d(t)$ are bounded. The function $V(x_d)$ is defined for a given application subsequently in experimental section of this chapter. For details on how to construct $\rho(x_n, x_d)$ for a specific application, the reader is referred to [79].

For the whole arm grasping objective, the desired task space velocity trajectory is defined as

$$\dot{x}_d(t) \triangleq \vartheta(x_n) \tag{4.12}$$

where $\vartheta(x_n)$ is the velocity field generated by the task-space position $x_n(t)$. The velocity tracking error is derived by taking the first derivative of (4.11) and using (4.12)

$$\dot{e} = \vartheta(x_n) - \dot{x}_n. \tag{4.13}$$

After utilizing (4.2), the expression in (4.13) can be written as follows

$$\dot{e} = \vartheta(x_n) - J_n \dot{q}. \tag{4.14}$$

Then, after utilizing (4.9), (4.14) can be written as follows

$$\dot{e} = \vartheta(x_n) - U_e \tag{4.15}$$

where (4.6) and (4.8) have also been used. Substituting (4.10), the closed loop error system can be written as follows

$$\dot{e} = -K_e e - k_n \left\| \frac{\partial V(x_d)}{\partial x_d} \right\|^2 \rho^2 (x_n, x_d)\, e. \tag{4.16}$$

Theorem 1 *The control law described by (4.10) guarantees that* $e(t)$, $\dot{e}(t)$ *and* $U_e(t) \in \mathcal{L}_\infty$ *and* $\|e(t)\| \to 0$ *as* $t \to \infty$.

Proof. For brevity in this presentation the proof of the theorem is omitted. Refer to [79], which describes a similar result. ■

The result of Theorem 1 proves that $\|e(t)\| \to 0$ as $t \to \infty$ and that $U_e(t) \in \mathcal{L}_\infty$. Thus, the control law defined in (4.10) guarantees that the manipulators end-effector follows the desired contour while also ensuring that all signals remain bounded. If the controller defined in (4.10) is used alone (i.e. $\dot{q}(t) = J_n^+ U_e$), the joint space desired trajectory that is tracked may take a path such that the end-effector and body of the manipulator make contact with the object while the end-effector tries to follow the contour of the object to be grasped. Since this is an undesirable effect, the *robot body self-motion positioning* controller must be designed in such a manner to provide object avoidance as the body of the manipulator moves around the object to be grasped.

Body Self-Motion Positioning

The objective of the *body self-motion positioning* controller is to "repel" the end-effector and the body of the manipulator away from the object to be grasped while the end-effector moves around the object. This control-induced repulsion-like property

not only facilitates object avoidance but also removes the "slack" from the body of the manipulator as the robot moves into the grasping position. When all possible slack is removed, the manipulator body makes contact with the object, hence completing the whole arm grasp of the object. Following this line of reasoning, the *body self-motion positioning* controller $U_m(t) \in \mathbb{R}^n$ in (4.9), is designed as follows

$$U_m \triangleq -k_m \left[J_s \left(I_n - J_n^+ J_n \right) \right]^T y_a \tag{4.17}$$

where $k_m \in \mathbb{R}^+$ is a control gain, $J_s \in \mathbb{R}^{1\times n}$ is a subsequently designed Jacobian-like vector, $I_n \in \mathbb{R}^{n\times n}$ was defined in (4.8), and $y_a(t) \in \mathbb{R}$ is an auxiliary scalar signal which encodes the object's geometric information. The function $y_a(t)$ is designed to keep the body of the manipulator away from the object as the robot end-effecor encircles the object. See [63] for details of a general auxiliary self-motion control signal of a redundant robot manipulator.

For whole arm grasping, a specific auxiliary signal $y_a(t)$ is designed as follows

$$y_a \triangleq \sum_{i=1}^{n} h_{ai}(x_i) \tag{4.18}$$

where n is the number of joints of the manipulator, $x_i = \begin{bmatrix} \bar{x}_{i1} & \bar{x}_{i2} & \dots & \bar{x}_{ip} \end{bmatrix}^T \in \mathbb{R}^p$ is the Euclidean-space coordinate for the i^{th} joint, and $h_{ai}(x_i) \in \mathbb{R}$ is the repulsion function for the i^{th} joint which encodes geometric information about the surface of the object with respect to the i^{th} joint's Euclidean position. The repulsion function $h_{ai}(x_i)$ is defined as follows

$$h_{ai}(x_i) = k_{hi} \exp\left(-\alpha_i \beta_i^2(x_i)\right), \qquad \forall\, i = 1, .., n \tag{4.19}$$

where k_{hi}, $\alpha_i \in \mathbb{R}^+$ are constants, and $\beta_i(x_i) \in \mathbb{R}$ is the joint specific geometric function. The function $\beta_i(x_i)$ should be designed to be positive when the manipulator is not touching the object as well as that $\beta_i(x_i) \in \mathcal{L}_\infty$, if $x_i(t) \in \mathcal{L}_\infty$. For example, given a spherical object in three dimensional Euclidean-space, $\beta_i(x_i)$ could be defined as follows

$$\beta_i(x_i) \triangleq (\bar{x}_{i1} - \bar{x}_{c1})^2 + (\bar{x}_{i2} - \bar{x}_{c2})^2 + (\bar{x}_{i3} - \bar{x}_{c3})^2 - r_o^2$$

where $\bar{x}_{c1}$, $\bar{x}_{c2}$, $\bar{x}_{c3}$, $r_o \in \mathbb{R}$ are the Euclidean coordinates of the center of the spherical object and its radius, respectively.

To determine the dynamics of $y_a(t)$, the time derivative of (4.18) is written as follows

$$\dot{y}_a = J_s \dot{q} \tag{4.20}$$

where a Jacobian-type vector $J_s(t) \in \mathbb{R}^{1\times n}$ is defined as follows

$$J_s = \frac{\partial y_a \left(\begin{array}{cccc} x_1 & x_2 & .. & x_n \end{array} \right)}{\partial \left[\begin{array}{cccc} x_1^T & x_2^T & .. & x_n^T \end{array} \right]} \left[\begin{array}{c} J_1 \\ : \\ J_n \end{array} \right] \tag{4.21}$$

where $\dot{x}_i = J_i \dot{q}$ and $J_i \in \mathbb{R}^{p\times n}$ is the Jacobian matrix relating the joint velocities and the Euclidean velocities for the i^{th} joint. Substituting for $\dot{q}(t)$ from (4.9), the expression for $\dot{y}_a(t)$ can be written as follows

$$\dot{y}_a = J_s J_n^+ U_e + J_s \left(I_n - J_n^+ J_n \right) U_m. \tag{4.22}$$

After substituting for $U_m(t)$ as defined in (4.17), $\dot{y}_a(t)$ of (4.22) can be further expressed as

$$\dot{y}_a = -k_m \left\| J_s \left(I_n - J_n^+ J_n \right) \right\|^2 y_a + J_s J_n^+ U_e. \tag{4.23}$$

Theorem 2 *The control law described by (4.17) guarantees that $y_a(t)$ is practically regulated (i.e., ultimately bounded) in the following sense*

$$|y_a(t)| \leq \sqrt{|y_a(t_0)|^2 \exp\left(-2\mu t\right) + \frac{\omega}{\mu}} \tag{4.24}$$

provided the following sufficient conditions are true

$$\left\| J_s \left(I_n - J^+ J \right) \right\|^2 > \bar{\delta} \tag{4.25}$$

and

$$k_m > \frac{1}{\bar{\delta}\delta_2} \tag{4.26}$$

where ω, μ, $\bar{\delta}$, $\delta_2 \in \mathbb{R}$ are positive constants.

Proof. For brevity in this presentation the proof of the theorem is omitted. Refer to [63], which describes a similar result. ■

Remark 4 *From (4.18) and (4.19), it is clear that* $0 < y_a(t) \leq \sum_{i=1}^{n} k_{hi}$ *and that as* $\beta_i(\cdot)$ *increases,* $h_{ai}(t)$ *decreases, and hence,* $y_a(t)$ *decreases. In addition each* $\beta_i(\cdot)$ *is designed such that* $\beta_i(\cdot) > 0$ *if the manipulators links are outside the object. From (4.24), it can be shown that the initial conditions of the manipulator and the bounding constants can be selected such that* $y_a(t) < \sum_{i=1}^{n} k_{hi}$*, hence, it is clear from (4.18) and (4.19) that* $\beta_i(t) > 0 \, \forall t$*.*

The result of Theorem 2 illustrates how the repulsive term $y_a(t)$ can be bounded by an exponentially decreasing function. This implies that when all the manipulators links are in contact with the object the auxiliary repulsion function $y_a(t)$ will approach a constant value ($\sum_{i=1}^{n} k_{hi}$), hence $\beta_i(t) \approx 0$. Interestingly, as the slack in the robot body is removed, the effect of the control term $U_m(t)$ is automatically reduced. This is because as the manipulator links make contact with the object, the number of redundant degrees of freedom available to accomplish the task space objective reduces. As a consequence, the self-motion component of the control input becomes almost zero (i.e., the null space projection $\|(I_n - J_n^+ J_n)\|$ approaches zero), and hence, (4.25) is no longer satisfied.

Low Level Control

In the previous section, a high level path planner was presented which enabled the whole arm grasping objective to be encoded as a desired trajectory signal which can be fed low level joint controller. This desired joint trajectory signal encodes information from the two auxiliary controllers, the *end-effector positioning* controller, and the *body self-motion positioning* controller. In this section, a filter is utilized to fuse the high level path planner with the low level joint tracking controller.

Fusing the Planner with the Controller

Traditionally for torque based control, the desired trajectory and its higher order derivatives are required in the control implementation. It is assumed that the desired trajectory and its higher order derivatives are always bounded for this problem to be tractable. In this section, a desired trajectory filter which generates bounded desired joint space trajectories for the joint space tracking controller is provided. The structure of the desired trajectory generator is motivated by the choice of the joint space controller [40], which is a continuous, nonlinear integral feedback controller and requires the desired trajectory to be bounded upto its fourth derivative. This controller was selected because of its ability to meet the tracking objective in the presence of system uncertainties (i.e. uncertainty in the robot dynamic model and unmeasurable contact forces).

To ensure that the desired joint space velocity trajectory is bounded, we could use the following expression

$$\dot{q}_d(t) \triangleq sat\,(\text{RHS of (4.9)}) \tag{4.27}$$

where RHS denotes the right hand side of the equation, $sat(\xi) \in \mathbb{R}^n$ is defined as $sat(\xi) = [sat(\xi_1), sat(\xi_2), \cdots, sat(\xi_n)]^T \; \forall \xi = [\xi_1, \xi_2, \cdots, \xi_n]^T \in \mathbb{R}^n$ where $sat(\xi_i) \in \mathbb{R} \; \forall i = 1, \cdots, n$ is the following saturation function

$$sat(\xi_i) = \begin{cases} -\xi_{min} & \text{if} \quad \xi_i \leq -\xi_{min} \\ \xi_i & \text{if} \quad \xi_i > -\xi_{min} \quad \text{or} \quad \xi_i < \xi_{max} \\ \xi_{max} & \text{if} \quad \xi_i \geq \xi_{max} \end{cases}$$

where $\xi_{min}, \xi_{max} \in \mathbb{R}^+$ are constants. If (4.27) is used to generate the desired trajectory, we cannot prove that $q_d(t)$ is bounded, so we could use the following filtering operation

$$q_d(s) \triangleq \frac{1}{\left(\frac{s}{\epsilon} + 1\right)} sat\,(\text{RHS of (4.9)}) \tag{4.28}$$

where $s \in \mathbb{C}$ is the standard Laplace variable, and $\epsilon \in \mathbb{R}^+$ is an integration constant selected very close to zero. However, in the case of (4.28), we cannot prove that the

higher order derivatives of $q_d(t)$ will remain bounded. So the desired trajectory $q_d(t)$ for the manipulator joint angles are generated by the following expression

$$q_d(s) \triangleq \frac{1}{\left(\frac{s}{\epsilon}+1\right)\left(\frac{s}{\kappa}+1\right)^3} sat\left(\text{RHS of (4.9)}\right) \tag{4.29}$$

where $\kappa \in \mathbb{R}^+$ is an integration constant selected to be very large. From (4.29), it is clear that $q_d(t)$, $\dot{q}_d(t)$, $\ddot{q}_d(t)$, $\dddot{q}_d(t)$, and $\ddddot{q}_d(t) \in \mathcal{L}_\infty$.

Low Level Controller

The objective of the closed-loop system is to ensure asymptotic tracking between the manipulator and the desired trajectory in the sense that

$$q(t) \to q_d(t) \text{ as } t \to \infty \tag{4.30}$$

where $q_d(t) \in \mathbb{R}^n$ is obtained from (4.29). To quantify the control objective, an error signal $e_1(t) \in \mathbb{R}^n$ is defined as follows

$$e_1 \triangleq q_d - q. \tag{4.31}$$

Furthermore, a tracking error signal $e_2(t) \in \mathbb{R}^n$ is defined as follows

$$e_2 \triangleq \dot{e}_1 + \gamma_1 e_1 \tag{4.32}$$

where $\gamma_1 \in \mathbb{R}^+$ is a control gain.

Since the robot dynamic model is a nonlinear uncertain multi-input multi-output system, the strategy developed by Xian *et al.* [40], can be used for the continuous low level controller. The control objective of (4.30) can be met with the following controller

$$\begin{aligned} \tau \triangleq\ & (K_s + I_n)\left[e_2(t) - e_2(t_0) + \gamma_2 \int_{t_0}^{t} e_2(\tau) d\tau\right] \\ & + \int_{t_0}^{t} \left[\Gamma sgn(e_2(\tau))\right] d\tau \end{aligned} \tag{4.33}$$

where $\tau(t) \in \mathbb{R}^n$ is the control input defined in (4.3), $K_s, \Gamma \in \mathbb{R}^{n \times n}$ are positive diagonal control gain matrices, and $sgn(\cdot) \in \mathbb{R}^n$ denotes the vector signum function defined as $sgn(\xi) = [sgn(\xi_1), sgn(\xi_2), \cdots, sgn(\xi_n)]^T \ \forall \xi = [\xi_1, \xi_2, \cdots, \xi_n]^T \in \mathbb{R}^n$. The controller presented in (4.33), provides asymptotic convergence of the joint tracking error, i.e. $\|e_1(t)\| \to 0$ as $t \to \infty$. For a detailed analysis of the controller the reader is referred to [40].

Remark 5 *For the experimental results presented in the subsequent section of this chapter the controller in (4.33) was utilized as the low level joint space controller. However, it is a simple matter to replace this controller with the neural network controller designed in the previous chapter, i.e. equation (3.9) could be utilized with the feedforward neural network component.*

Remark 6 *The trajectory generator defined in (4.29) generates a filtered version of (4.9). This filtered signal is used as a desired trajectory for the joint space controller defined in (4.33). The joint space controller (4.33), forces the actual robot joint angles to track the filtered desired trajectory of (4.29). However, we cannot show that the actual robot joint velocities track the kinematic velocity signal defined in (4.9). Thus, the results of Theorem 1 and Theorem 2 may not be technically valid, however, the validity of the approach is illustrated through the experimental results presented in the subsequent section.*

Experimental Results

To evaluate the performance of the proposed approach experiments were conducted using a planar, three link configuration of the Barrett whole arm manipulator (WAM). In this section indicative results are presented to demonstrate the validity of the whole arm grasping controller.

Experimental Testbed

The Barrett WAM is a seven degree of freedom (d.o.f.), highly dexterous and back-drivable robotic manipulator. To simplify the controller implementation, four joints of the robot were locked at fixed angles and the remaining links of the manipulator were used as a three d.o.f. planar robot manipulator. Figure 4.1, shows the experimental setup with a circular object to be grasped.

The control algorithm was written in "C++" and hosted on an AMD Athlon 1.2 GHz PC running the QNX 6.2.1 real-time operating system. Data logging and on-line gain tuning was performed using Qmotor 3.0 control software [65]. Data acquisition and control implementation was performed at a frequency of 1.0 [kHz] using the ServoToGo I/O board. Joint positions were measured using the optical encoders located at the motor shaft of each axis and joint velocity measurements were obtained using a filtered backwards difference algorithm. In this demonstration of the whole arm grasping controller a rigid cylindrical object was selected.

Path Planner and Controller Setup

Refer to Figure 4.2 for an explanation of the notations used in this section. $X_c = [x_c, y_c]^T \in \mathbb{R}^2$ represents the co-ordinates of the center of the object and $r_0 \in \mathbb{R}$ represents the object radius. We define the task space variable for each of the three links and the mid-point[5] of each of the three links as $X_i = [x_i, y_i]^T \in \mathbb{R}^2 \; \forall \, i = 1, 2, .., 6$. The joint angles for the three links are represented by $q = [q_1, q_2, q_3]^T \in \mathbb{R}^3$. The object specific functions defined for each of the three links and the mid-points of the three links are defined as $\beta_1(\cdot), \beta_2(\cdot), \cdots, \beta_6(\cdot) \in \mathbb{R}$.

The object specific functions for this planar application were defined as follows

$$\beta_i\left(X_i\right) \triangleq \left(x_i - x_c\right)^2 + \left(y_i - y_c\right)^2 - r_o^2 \; \forall \, i = 1, .., 6. \tag{4.34}$$

[5]The object function for the mid-points of each of the manipulator links was used for this three link application, since it provides more control over the body self-motion positioning control (i.e. the repulsion functions) than just using the link end-points alone.

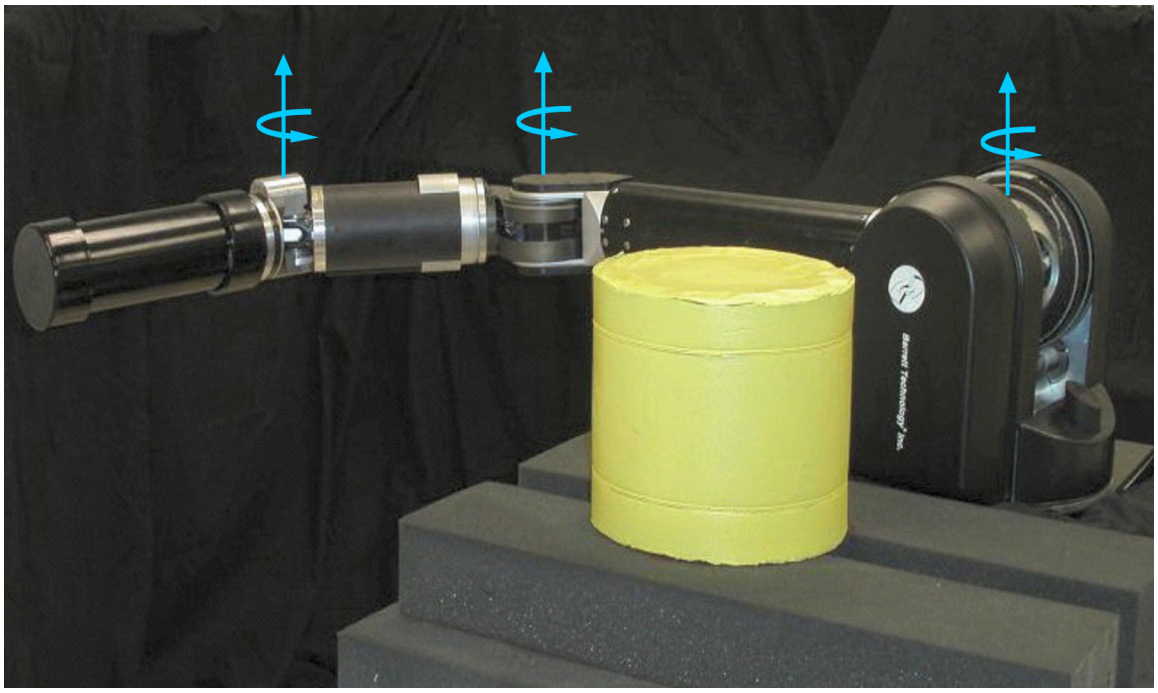

Figure 4.1 Experiment setup showing the Barrett Whole Arm manipulator and object to the grasped.

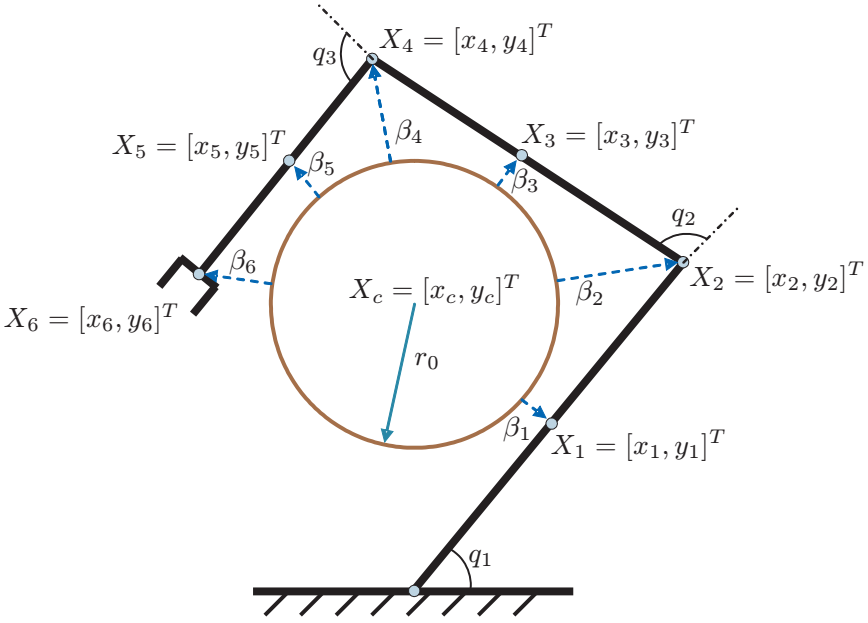

Figure 4.2 Planar configuration for the three link robot with a circular object.

The task-space velocity field which provides the desired end effector trajectory for a planar, circular contour, was defined as follows [76]

$$\begin{aligned}\dot{X}_d = \vartheta(X_6) = -2K(X_6)f(X_6)\begin{bmatrix}(x_6 - x_c)\\(y_6 - y_c)\end{bmatrix}\\ +2c(X_6)\begin{bmatrix}-(y_6 - y_c)\\(x_6 - x_c)\end{bmatrix}\end{aligned} \tag{4.35}$$

where the functions $f(X_6)$, $K(X_6)$, and $c(X_6) \in \mathbb{R}$ are defined according to [76] as follows

$$\begin{aligned} f(X_6) &= (x_6 - x_c)^2 + (y_6 - y_c)^2 - r_0^2 \\ K(X_6) &= \frac{k_0^*}{\sqrt{f^2(X_6)}\left\|\frac{\partial f(X_6)}{\partial X_6}\right\| + \epsilon_0} \\ c(X_6) &= \frac{c_0 \exp\left(-\mu_0\sqrt{f^2(X_6)}\right)}{\left\|\frac{\partial f(X_6)}{\partial X_6}\right\|}. \end{aligned} \tag{4.36}$$

In (4.36), the constant parameters were selected as $\epsilon_0 = 0.005$ [m^3], $\mu_0 = 20$ [m^{-1}], $k_0^* = 0.1$ [ms^{-1}], and $c_0 = 0.1$ [ms^{-1}]. These constants were selected in such a manner as to provide smooth motion of the end effector. The constants can be modified to either increase or decrease the velocity of the desired end effector trajectory. The desired position for the end-effector was represented as $X_d = [x_d, y_d]^T \in \mathbb{R}^2$. The controller defined in (4.10) was implemented with $e = X_d - X_6$, $V(X_d) \triangleq 4\|X_d\|^2$, and $\rho(\cdot) = 1$.

The initial joint angles were $q_1(0) = 98$[deg], $q_2(0) = 45.8$[deg], $q_3(0) = 31$[deg], which corresponds to a position of $x_6(0) = 0.368$ [m] $\;\; y_6(0) = -0.883$ [m] for the end-effector in the task space. The position of the object center in the task space was $x_c = 0.307$ [m], $y_c = -0.117$ [m], and the radius of the circular object was found to be 0.12 [m]. To take into account the width of the manipulator arm, the radius of the object was augmented to $r_0 = 0.16$ [m] in the implementation of the repulsion function.

Control Parameter Tuning

The control gains which gave the good performance were found to be as follows

$$K_e = diag\{800, 800\},\ k_n = 1,\ k_m = 100,$$
$$k_h = \{0.001, 0.01, 0.01, 0.05, 8.5, 8\},$$
$$\alpha_i = \{3, 3, 3, 3, 5, 5\},\ Ks = diag\{16, 9, 6\},$$
$$\Gamma = diag\{5, 5, 2\},\ \gamma_1 = diag\{1, 1, 1\},$$
$$\gamma_2 = diag\{2, 2, 2\},\ \epsilon = 0.01,\ \kappa = 500.$$

These control gains were determined in the following manner; first the gains for the path planner were set to some arbitrary values and the gains in the low level controller, $K_s, \Gamma, \gamma_1, \gamma_2$, were adjusted till the joint space tracking error was within a satisfactory limit. At the same time the gains for the joint trajectory filter were tuned such that the desired joint trajectory being provided to the low level controller was bounded and smooth. In the filter, the value for κ must be selected to be very large compared to the value of ϵ. Once the low level controller was tuned satisfactorily the gains in the path planner were adjusted. In the path planner, first the end-effector positioning control gains K_e, k_n, were adjusted such that the end-effector tracked the trajectory being generated from the velocity field. Then the gains in the body self-motion control were adjusted so that the individual links of the manipulator avoided contact with the object while the end-effector was moving around the object. Notice that the gains for the last two elements of k_h, which are the repulsion function gains for the third link, were much higher than the other links as the third link had to travel much further around the object before the grasp could be completed.

Results and Discussion

Figure 4.3 shows the actual and desired joint angles of the three links and figure 4.4 shows the value of the joint tracking error. From figures 4.3 and 4.4 we see that the joint tracking error is within $3 - 4$ degrees. Figure 4.5 shows the joint control

torques during the whole arm grasp. Figure 4.6 shows the time evolution of the spatial position of each of the links and their mid-points during the grasp.

Since the desired trajectory for the end-effector is being generated using the velocity field, it will continuously generate the trajectory even after the manipulator has moved into a optimal grasp configuration. To determine when the desired trajectory generation must be stopped, the norm of the following vector $\beta(\cdot) = [\beta_1(\cdot), \beta_2(\cdot), \cdots, \beta_6(\cdot)] \in \mathbb{R}^6$ was used to determine when when all the links of the manipulator make contact with the object. Notice that as the links of the manipulator move closer to the object boundary, $\|\beta(\cdot)\|$ approaches zero, and this gives an estimate of how close the manipulators links are to the object. For the experiment, we stopped the trajectory generation by setting $\dot{X}_d = 0$ when $\|\beta(\cdot)\| \leq \eta_0$, where the constant $\eta_0 = 0.01$ was determined experimentally.

Remark 7 *The value of $\|\beta(\cdot)\|$ at which we stop the generation of the desired trajectory is specific to a particular grasp configuration. It will change if the object is re-positioned in the task space. However, if we use a highly redundant robot arm which can wrap its entire body around the object, then $\|\beta(\cdot)\|$ will approach zero when the arm grasps the object, since the entire body of the arm will be in contact with the object.*

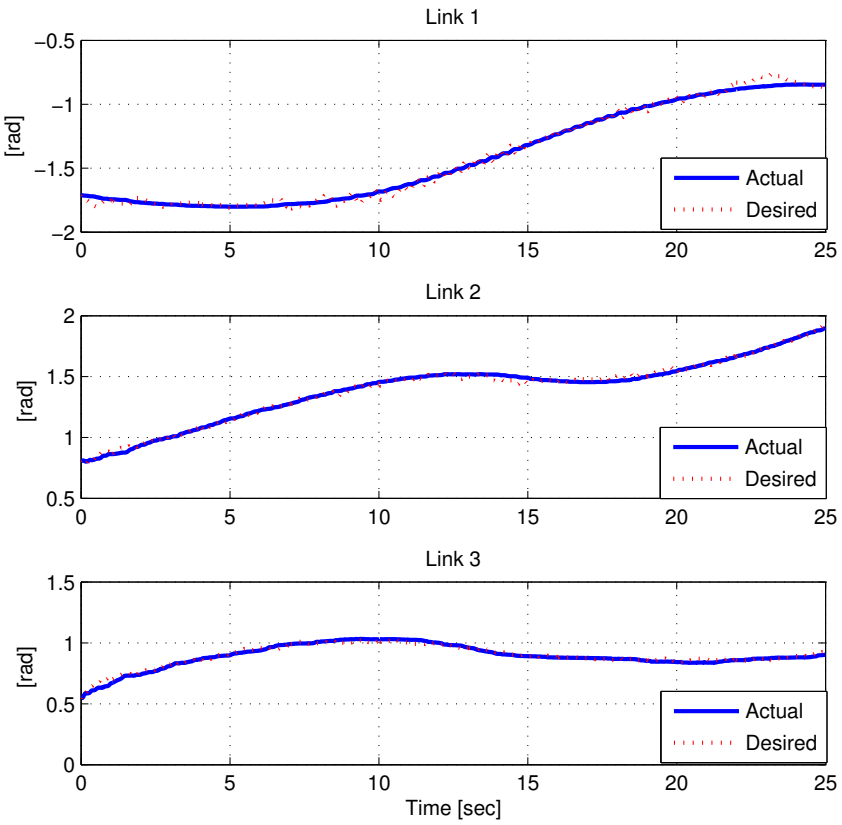

Figure 4.3 Desired joint angles $q_d(t)$ and actual joint angles $q(t)$.

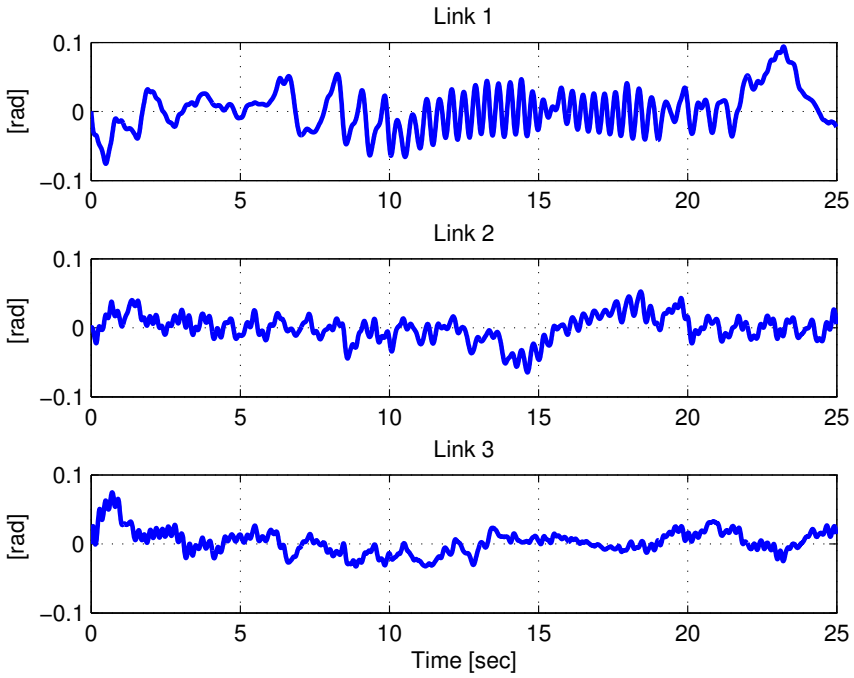

Figure 4.4 Joint space position tracking error $e_1(t)$.

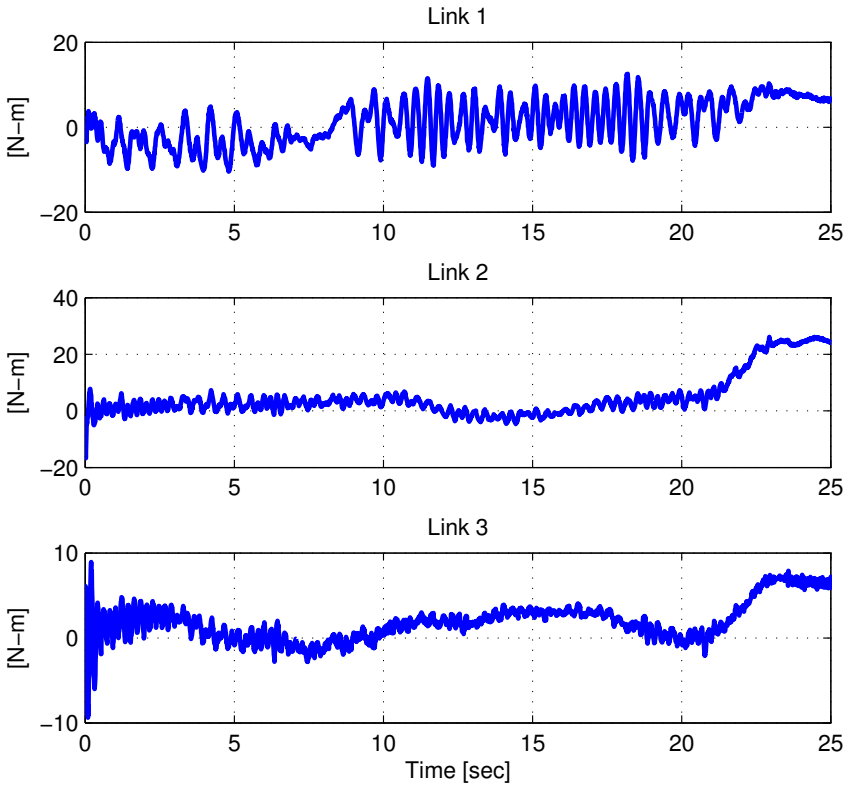

Figure 4.5 Joint space control torques $\tau(t)$.

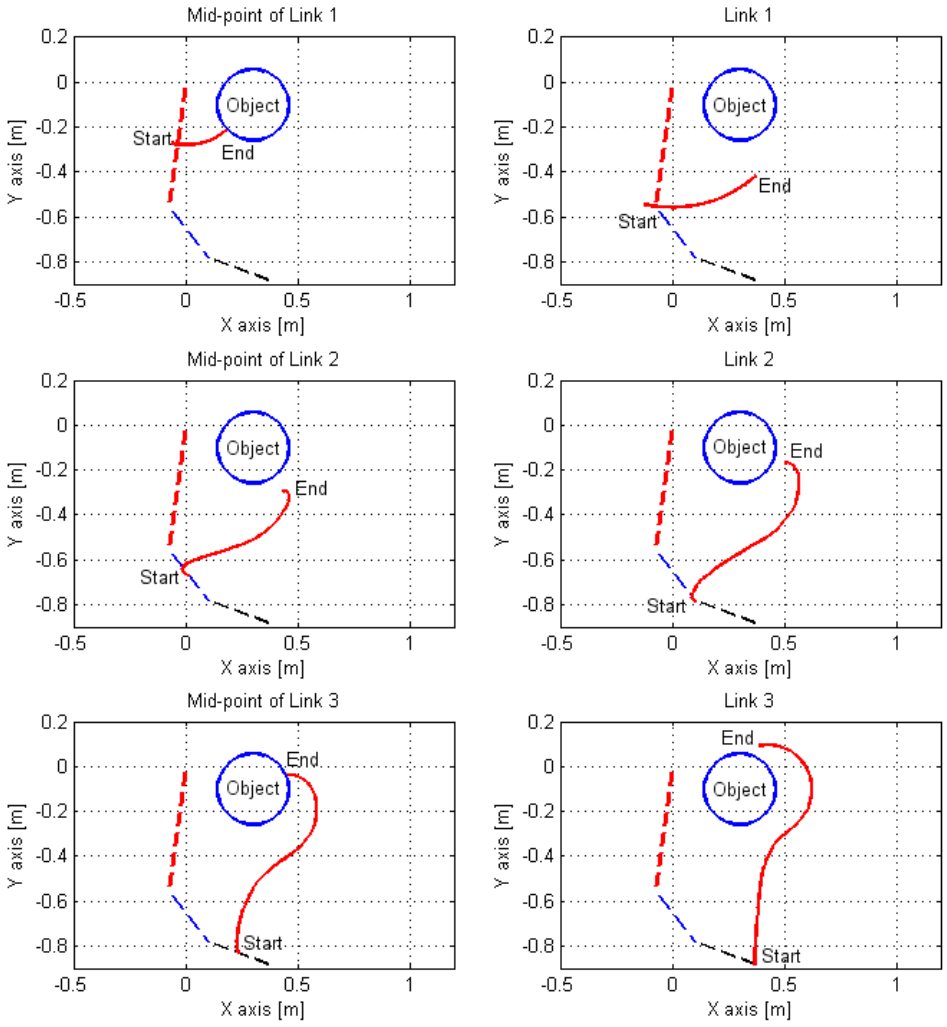

Figure 4.6 Spatial position $X_i \; \forall \, i = 1, \cdots, 6$ (each link and mid-point of the link).

CHAPTER 5
CONCLUSION

In Chapter 2, a task-space, adaptive tracking controller for robot manipulators with uncertainty in both the kinematic and the dynamic models was developed. The controller yields asymptotic regulation of the end-effector position and orientation tracking errors. The advantages of the proposed controller are that singularities associated with the three parameter representation are avoided, and unlike previous research the controller does not require the measurement of the task-space velocity. The experiment carried out on a planar, two link configuration of the Barrett WAM validates the performance of the adaptive controller.

In Chapter 3, a neural network based joint tracking controller for continuum robot manipulators was developed. Unlike previous research, the proposed approach does not require that an accurate dynamic model of the continuum manipulator be known. The feedforward neural network estimation scheme was used to compensate for the uncertain nonlinear dynamics of the continuum manipulator. Experimental results for the OCTARM continuum robot manipulator were presented to demonstrate the significant performance improvement provided by the neural network feedforward estimation technique. The developed approach is not dependent on any specific manipulator model and hence it can be easily adapted to a wide range of continuum manipulator designs.

Finally in Chapter 4, a whole arm grasping controller for kinematically redundant robot manipulators was presented. A high level path planner which enables *end-effector position* tracking as well as *body self-motion positioning* control to be encoded as the desired joint trajectory was developed. Then, the joint tracking controller developed in Chapter 3, which enables the robot to track a desired trajectory in the presence of system uncertainties and unmeasurable contact forces was utilized. The controller provides asymptotic tracking which enables the whole arm grasping objective to be completed. Experimental results for a planar, three link configura-

tion of the Barrett WAM are provided to demonstrate the efficacy of the whole arm grasping controller.

BIBLIOGRAPHY

[1] F. Caccavale, C. Natale, and L. Villani, "Task-space control without velocity measurements," in *Proc. IEEE Int. Conf. Robot. Autom.*, Detroit, MI, 1999, pp. 512–517.

[2] R. Campa, R. Kelly, and E. Garcia, "On stability of the resolved acceleration control," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, 2001, pp. 3523–3528.

[3] F. Lizarralde and J. T. Wen, "Attitude control without angular velocity measurement: A passivity approach," *IEEE Trans. Automat. Contr.*, vol. 41, no. 3, pp. 468–472, 1996.

[4] B. Xian, M. S. de Queiroz, D. Dawson, and I. Walker, "Task-space tracking control of robot manipulators via quaternion feedback," *IEEE Trans. Robot. Automat.*, vol. 20, no. 1, pp. 160–167, 2004.

[5] J. S. C. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE Trans. Robot. Automat.*, vol. 4, no. 4, pp. 434–440, 1988.

[6] C. C. Cheah, M. Hirano, S. Kawamura, and S. Arimoto, "Approximate jacobian control for robots with uncertain kinematics and dynamics," *IEEE Trans. Robot. Automat.*, vol. 19, no. 4, pp. 692–702, 2003.

[7] ——, "Approximate jacobian control with task-space damping for robot manipulators," *IEEE Trans. Automat. Contr.*, vol. 49, no. 5, pp. 752–757, 2004.

[8] C. C. Cheah, S. Kawamura, and S. Arimoto, "Feedback control for robotic manipulators with uncertain kinematics and dynamics," in *Proc. IEEE Int. Conf. Robot. Autom.*, Leuven, Belgium, 1998, pp. 3607–3612.

[9] ——, "Feedback control for robotic manipulators with an uncertain jacobian matrix," *Int. Journal of Robotic Systems*, vol. 12, no. 2, pp. 119–134, 1999.

[10] ——, "Pid control for robotic manipulator with uncertain jacobian matrix," in *Proc. IEEE Int. Conf. Robot. Autom.*, Detroit, MI, 1999, pp. 494–499.

[11] C. C. Cheah, S. Kawamura, S. Arimoto, and K. Lee, "H_∞ tuning for task-space feedback control of robot with uncertain jacobian matrix," *IEEE Trans. Automat. Contr.*, vol. 46, no. 8, pp. 1313–1318, 2001.

[12] H. Yazarel and C. C. Cheah, "Task-space adaptive control of robotic manipulators with uncertainties in gravity regressor matrix and kinematics," *IEEE Trans. Automat. Contr.*, vol. 47, no. 9, pp. 1580–1585, 2002.

[13] W. E. Dixon, "Adaptive regulation of amplitude limited robot manipulators with uncertain kinematics and dynamics," in *Proc. American Control Conf.*, Boston, MA, 2004, pp. 3839–3844.

[14] C. Liu and C. C. Cheah, "Adaptive regulation of rigid-link electrically driven robots with uncertain kinematics," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, 2005, pp. 3273–3278.

[15] C. C. Cheah, C. Liu, and J. J. Slotine, "Approximate jacobian adaptive control for robot manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, La, 2004, pp. 3075–3080.

[16] ——, "Adaptive jacobian tracking control of robots based on visual task-space information," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, 2005, pp. 3509–3514.

[17] ——, "Adaptive jacobian tracking control of robots with uncertainties in kinematic, dynamic and actuator models," *IEEE Trans. Automat. Contr.*, vol. 51, no. 6, pp. 1024–1029, 2006.

[18] G. Robinson and J. B. C. Davies, "Continuum robots - a state of the art," in *Proc. IEEE Int. Conf. Robot. Autom.*, Detroit, Michigan, USA, 1999, pp. 2849–2854.

[19] I. D. Walker, D. M. Dawson, T. Flash, F. Grasso, R. Hanlon, B. Hochner, W. Kier, C. Pagano, C. D. Rahn, and Q. Zhang, "Continuum robot arms inspired by cephalopods," in *Proc. 2005 SPIE Conf. Unmanned Ground Vehicle Technology IV*, Orlando, Florida, USA, Mar. 2005, pp. 303–314.

[20] R. Cieslak and A. Morecki, "Elephant trunk type elastic manipulator - a tool for bulk and liquid materials transportation," *Robotica*, vol. 17, pp. 11–16, 1999.

[21] H. Tsukagoshi, A. Kitagawa, and M. Segawa, "Active hose: An artificial elephant's nose with maneuverability for rescue operation," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, 2001, pp. 2454–2459.

[22] M. A. Hannan and I. D. Walker, "Kinematics and the implementation of an elephant's trunk manipulator and other continuum style robots," *Journal of Robotic Systems*, vol. 20, no. 2, pp. 45–63, Feb. 2003.

[23] J. Yang, E. P. Pitarch, J. Potratz, S. Beck, and K. Abdel-Malek, "Synthesis and analysis of a flexible elephant trunk robot," *Advanced Robotics*, vol. 20, no. 6, p. 631659, 2006.

[24] W. McMahan, V. Chitrakaran, M. Csencsits, D. M. Dawson, I. D. Walker, B. Jones, M. Pritts, D. Dienno, M. Grissom, and C. Rahn, "Field trials and testing of the octarm continuum manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, 2006, pp. 2336–2341.

[25] S. Hirose, *Biologically Inspired Robots*. New York, NY: Oxford University Press, 1993.

[26] F. Matsuno and K. Suenga, "Control of redundant snake robot based on kinematic model," in *Proc. 41st SICE Annual Conference*, Osaka, Japan, 2002, pp. 1481–1486.

[27] M. Ivanescu, "Position dynamic control for a tentacle manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, 2002, pp. 1531–1538.

[28] M. Ivanescu, N. Popescu, and D. Popescu, "A variable length tentacle manipulator control system," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, 2005, pp. 3285–3290.

[29] I. A. Gravagne and I. D. Walker, "Uniform regulation for a multi-section continuum manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Washington, DC, USA, 2002, pp. 1519–1524.

[30] T. Suzuki, K. Shintani, and H. Mochiyama, "Control methods of hyper-flexible manipulators using their dynamical features," in *Proc. 41st SICE Annual Conference*, Osaka, Japan, 2002, pp. 1511–1516.

[31] H. Mochiyama, E. Shimemura, and H. Kobayashi, "Control of manipulators with hyper degrees of freedom: shape tracking using only joint angle information," *International Journal of Systems Science*, vol. 30, no. 1, pp. 77–85, 1999.

[32] F. Matsuno and H. Sato, "Trajectory tracking control of snake robots based on dynamic model," in *Proc. IEEE Int. Conf. Robot. Autom.*, Barcelona, Spain, 2005, pp. 3040–3045.

[33] F. L. Lewis, S. Jagannathan, and A. Yesildirek, *Neural Network Control of Robot Manipulators and Nonlinear Systems*. London: Taylor and Francis, June 1999.

[34] S. S. Ge, T. H. Lee, and C. J. Harris, *Adaptive Neural Network Control of Robotic Manipulators*. London, UK: World Scientific Publishing Company, Feb. 1999.

[35] Y. H. Kim and F. L. Lewis, "Neural network output feedback control of robot manipulators," *IEEE Trans. Robot. Automat.*, vol. 15, no. 2, pp. 301–309, Apr. 1999.

[36] F. C. Sun, Z. Q. Sun, R. J. Zhang, and Y. B. Chen, "Neural adaptive tracking controller for robot manipulators with unknown dynamics," *IEE Proc. Control Theory and Applications*, vol. 147, no. 3, pp. 366–370, 2000.

[37] H. D. Patino, R. Carelli, and B. R. Kuchen, "Neural networks for advanced control of robot manipulators," *IEEE Trans. Neural Networks*, vol. 13, no. 2, pp. 343–354, Mar. 2002.

[38] M. K. Ciliz, "Adaptive control of robot manipulators with neural network based compensation of frictional uncertainties," *Robotica*, vol. 23, pp. 159–167, 2005.

[39] D. Braganza, D. M. Dawson, I. D. Walker, and N. Nath, "Neural network grasping controller for continuum robots," in *Proc. IEEE Int. Conf. Decision and Control*, San Diego, CA, 2006, pp. 6445–6449.

[40] B. Xian, D. M. Dawson, M. S. de. Queiroz, and J. Chen, "A continuous asymptotic tracking control strategy for uncertain nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 49, no. 7, pp. 1206–1211, 2004.

[41] O. M. Omidvar and D. L. Elliott, Eds., *Neural Systems for Control*. Boston, MA: Academic Press, 1997, ch. 7.

[42] K. Salisbury, "Whole arm manipulation," in *Proc. 4th Int. Symposium Robotics Research*, 1987, pp. 183–189.

[43] A. Bicchi, "Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity," *IEEE Trans. Robot. Automat.*, vol. 16, no. 6, pp. 652–662, 2000.

[44] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *Proc. IEEE Int. Conf. Robot. Autom.*, San Francisco, CA, 2000, pp. 348–353.

[45] C. Melchiorri and G. Vassura, "Implementation of whole-hand manipulation capability in the ub hand system design," *Robotics Society of Japan - Advanced Robotics*, vol. 9, no. 5, pp. 547–560, 1995.

[46] H. V. B. D. Reynaerts, "Whole-finger manipulation with a two-fingered robot hand," *Robotics Society of Japan - Advanced Robotics*, vol. 9, no. 5, pp. 505–518, 1995.

[47] K. Mirza and D. E. Orin, "Force distribution for power grasp in the digits system," *CSIM-IFToMM Symp, Theory and Practice of Robots and Manipulators*, 1990.

[48] M. J. Sheridan, S. C. Ahalt, and D. E. Orin, "Fuzzy control for robotic power grasp," *Robotics Society of Japan - Advanced Robotics*, vol. 9, no. 5, pp. 535–546, 1995.

[49] J. C. Trinkle, J. M. Abel, and R. P. Paul, "An investigation of frictionless enveloping grasping in the plane," *Int. J. Robot. Res.*, vol. 7, no. 3, pp. 33–51, 1988.

[50] F. Asano, Z. Luo, M. Yamakita, and S. Hosoe, "Dynamic modeling and control for whole body manipulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Las Vegas, NV, 2003, pp. 3162–3167.

[51] G. Chirikjian and J. W. Burdick, "Design and experiments with a 30 dof robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Atlanta, GA, 1993, pp. 113–119.

[52] H. Mochiyama, "Whole-arm impedance of a serial-chain manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, 2001, pp. 2223–2228.

[53] R. Platt, A. H. Fagg, and R. A. Grupen, "Nullspace composition of control laws for grasping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Lausanne, Switzerland, 2002, pp. 1717–1723.

[54] P. Song, M. Yashima, and V. Kumar, "Dynamics and control of whole arm grasps," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seoul, Korea, 2001, pp. 2229–2234.

[55] F. Asano, Z. Luo, M. Yamakita, K. Tahara, and S. Hosoe, "Bio-mimetic control for whole arm cooperative manipulation," in *Proc. IEEE Int. Conf. Robot. Autom.*, The Hague, The Netherlands, 2004, pp. 704–709.

[56] R. Platt, A. H. Fagg, and R. A. Grupen, "Extending fingertip grasping to whole body grasping," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, 2003, pp. 2677–2682.

[57] F. L. Lewis, D. M. Dawson, and C. Abdallah, *Robot Manipulator Control: Theory and Practice*. New York, NY: Marcel Dekker, 2003.

[58] P. C. Hughes, *Spacecraft Attitude Dynamics*. New York, NY: Wiley, 1994.

[59] J. B. Kuipers, *Quaternions and Rotation Sequences*. Princeton, NJ: Princeton University Press, 1999.

[60] W. E. Dixon, A. Behal, D. M. Dawson, and S. Nagarkatti, *Nonlinear control of engineering systems: A lyapunov-based approach*. Boston, MA: Birkhauser, 2003.

[61] M. Spong and M. Vidyasagar, *Robot Dynamics and Control*. New York, NY: John Wiley & Sons Inc., 1989.

[62] M. Krstic, I. Kanellakopoulus, and P. Kokotovic, *Nonlinear and Adaptive Control Design*. New York, NY: John Wiley & Sons, 1995.

[63] E. Tatlicioglu, M. McIntyre, D. Dawson, and I. Walker, "Adaptive nonlinear tracking control of kinematically redundant robot manipulators with subtask extensions," in *Proc. IEEE Int. Conf. Decision and Control*, Seville, Spain, 2005, pp. 4373–4378.

[64] J. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliff, NJ: Prentice Hall Inc., 1991.

[65] M. Loffler, N. Costescu, and D. M. Dawson, "Qmotor 3.0 and the qmotor robotic toolkit - an advanced pc-based real-time control platform," *IEEE Control Syst. Mag.*, vol. 22, no. 3, pp. 12–26, June 2002.

[66] W. McMahan, B. Jones, I. D. Walker, V. Chitrakaran, A. Seshadri, and D. Dawson, "Robotic manipulators inspired by cephalopod limbs," in *CDEN Symposium on Biomimicry, Bionics and Biomechanics*, Montreal, Canada, 2004, pp. 1–10.

[67] M. B. Pritts and C. D. Rahn, "Design of an artificial muscle continuum robot," in *Proc. IEEE Int. Conf. Robot. Autom.*, New Orleans, LA, USA, 2004, pp. 4742–4746.

[68] B. A. Jones and I. D. Walker, "Kinematics for multisection continuum robots," *IEEE Trans. Robot.*, vol. 22, no. 1, pp. 43–55, Feb. 2006.

[69] I. Gravagne, C. Rahn, and I. D. Walker, "Large deflection dynamics and control for planar continuum robots," *IEEE/ASME Trans. Mechatron.*, vol. 8, no. 2, pp. 299–307, June 2003.

[70] Y. Yekutieli, R. Sagiv-Zohar, R. Aharonov, Y. Engel, B. Hochner, and T. Flash, "Dynamic model of the octopus arm. i. biomechanics of the octopus arm reaching movement," *Journal of Neurophysiology*, vol. 94, pp. 1443–1458, 2005.

[71] E. Tatlicioglu, I. D. Walker, and D. M. Dawson, "Dynamic modeling for planar extensible continuum robot manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, Rome, Italy, 2007, pp. 1357–1362.

[72] H. Mochiyama and T. Suzuki, "Dynamic modeling of a hyper-flexible manipulator," in *Proc. 41st SICE Annual Conference*, Osaka, Japan, 2002, pp. 1505–1510.

[73] ——, "Kinematics and dynamics of a cable-like hyper-flexible manipulator," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, 2003, pp. 3672–3677.

[74] M. Cscenstis, B. A. Jones, W. McMahan, V. Iyengar, and I. D. Walker, "User interfaces for continuum robot arms," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Edmonton, AB, Canada, 2005, p. 30113018.

[75] Y. Nakamura, *Advanced Robotics Redundancy and Optimization*. Reading, MA: Addison-Wesley, 1991.

[76] I. Cervantes, R. Kelly, J. Alvarez-Ramirez, and J. Moreno, "A robust velocity field control," *IEEE Trans. Contr. Syst. Technol.*, vol. 10, no. 6, pp. 888–894, 2002.

[77] J. Li and P. Li, "Passive velocity field control (pvfc) approach to robot force control and contour following," in *Proc. Japan/USA Symposium on Flexible Automation*, Ann Arbor, Michigan, 2000.

[78] P. Li and R. Horowitz, "Passive velocity field control of mechanical manipulators," *IEEE Trans. Robot. Automat.*, vol. 15, no. 4, pp. 751–763, 1999.

[79] M. McIntyre, W. Dixon, D. Dawson, and B. Xian, "Adaptive tracking control of on-line path planners: Velocity fields and navigation functions," in *Proc. American Control Conf.*, Portland, Oregon, 2005, pp. 3168–3173.