KINEMATIC MODELING, IDENTIFICATION, AND CONTROL OF ROBOTIC MANIPULATORS

KINEMATIC MODELING, IDENTIFICATION, AND CONTROL OF ROBOTIC MANIPULATORS

Henry W. Stone
Carnegie-Mellon University

KLUWER ACADEMIC PUBLISHERS
Boston/Dordrecht/Lancaster

Distributors for North America:
Kluwer Academic Publishers
101 Philip Drive
Assinippi Park
Norwell, MA 02061, USA

Distributors for the UK and Ireland:
Kluwer Academic Publishers
MTP Press Limited
Falcon House, Queen Square
Lancaster LA1 1RN, UNITED KINGDOM

Distributors outside North America:
Kluwer Academic Publishers Group
Distribution Centre
P.O. Box 322
3300 AH Dordrecht, THE NETHERLANDS

Library of Congress Cataloging-in-Publication Data

Stone, Henry W.
Kinematic modeling, identification, and control of robotic manipulators.

(Kluwer international series in engineering and computer science ; SECS 29. Robotics)
Based on the author's dissertation—Carnegie Mellon University, 1986.
Bibliography: p.
Includes index.
1. Robotics. 2. Manipulators (Mechanism) I. Title. II. Series: Kluwer international series in engineering and computer science ; SECS 29. III. Series: Kluwer international series in engineering and computer science. Robotics.
TH211.S76 1987 629.8'92 87-3122

ISBN-13: 978-1-4612-9193-0 e-ISBN-13: 978-1-4613-1999-3
DOI: 10.1007/978-1-4613-1999-3

TO GRETCHEN

Table of Contents

List of Figures

List of Tables

Preface

The objective of this dissertation is to advance the state-of-the-art in the kinematic modeling, identification, and control of robotic manipulators with rigid links in an effort to improve robot kinematic performance.

The positioning accuracy of commercially-available industrial robotic manipulators depends upon a kinematic model which describes the robot geometry in a parametric form. Manufacturing error in the machining and assembly of manipulators lead to discrepancies between the design parameters and the physical structure. Improving the kinematic performance thus requires the identification of the *actual* kinematic parameters of each individual robot. The identified kinematic parameters are referred to as the ***arm signature***.

Existing robot kinematic models, such as the Denavit-Hartenberg model, are not directly applicable to kinematic parameter identification. In this dissertation we introduce a new kinematic model, called the S-Model, which is applicable to kinematic parameter identification, and use it as the foundation for our development of a general technique for identifying the kinematic parameters of any robot with rigid links.

The objective of our S-Model identification algorithm is to estimate the S-Model kinematic parameters from a set of mechanical features which are inherent to the manipulator. Each revolute joint possesses two such features and each prismatic joint possesses one. These features contain the essential information to model completely the kinematics of a manipulator. The initial step of the algorithm involves the explicit identification of the feature parameters. Each feature is identified in an independent procedure and is based upon measurements of the three-dimensional Cartesian positions of target points mounted on each of the links of the manipulator. A relatively simple and systematic method for collecting these measurements is one of the practical advantages of our approach. The identified feature parameters are then used to establish the positions and orientations of Cartesian coordinate frames fixed relative to each link of the manipulator in accordance with the definition of the S-Model. The parameters of the S-Model are then computed from the estimated link coordinate frame locations. Finally, the Denavit-Hartenberg parameters for the manipulator are extracted from the identified S-Model parameters.

We have implemented a complete prototype arm signature identification system and have applied it to identify the signatures and control the end-effector of seven Unimation/Westinghouse Puma 560 robots. Evaluation of the experimental results has demonstrated consistent and significant improvements in the kinematic performance of all the robots tested.

Acknowledgements

I would like to express my sincere thanks to Professor Arthur C. Sanderson, my thesis advisor, for his guidance, encouragement, and patience during my many years at CMU. His thorough reading and constructive criticism have greatly improved this work. In addition, I am grateful for the financial support which he has made available. I thank Professors Bruce H. Krogh, Friedrich B. Prinz, and Marc H. Raibert for their reading of the dissertation and their suggestions which have also contributed to improving this work. I thank Dr. Gene Bartel for his assistance in developing the sensor system and associated hardware utilized in our laboratory experiments.

Special thanks go to my colleagues, Lee Weiss, Vassilios Tourassis, and Pradeep Khosla with whom I have had numerous and informative discussions, and general commiseration sessions. I would also like to thank my office mates of late, Nancy Cornelius and Dave Simon, and Angie and Joel Ferguson , Bill Birmingham, and Ben Motazed for their support and friendship.

A very special thanks to my parents whom have given me their love and encouragement.

Finally, and most importantly, I wish to express my love and appreciation to my wife, Gretchen, for her immeasureable patience, encouragement, and support. Without her this dissertation would not have been possible.

KINEMATIC MODELING, IDENTIFICATION, AND CONTROL OF ROBOTIC MANIPULATORS

Chapter 1
Introduction

1.1. Overview

This dissertation describes the development of a general technique for identifying kinematic parameters of serial link robotic manipulators and for improving their kinematic performance. The research described here addresses three problems in robot kinematics: ***modeling, parameter identification,*** and *control.* Accurate robot kinematic models are needed to improve the positioning and orienting accuracy of commercially available robotic manipulators. The strong inter-relationships between the parameters of currently formulated kinematic models makes it extremely difficult to apply and guarantee the convergence of standard parameter estimation techniques such as least-squares. Thus, kinematic models which simplify the identification process are needed. Kinematic models of manipulators with revolute joints are inherently nonlinear and identification algorithms must be capable of accurately identifying the nonlinear kinematic parameters. Since the identified models will *not* possess closed-form inverse solutions, new methods for designing and implementing robot kinematic control algorithms must be developed, which incorporate the identified kinematic parameters, to improve robot kinematic performance.

This introduction discusses the significance of these problems and the motivation for the research (in Section 1.2), and highlights goals and contributions (in Section 1.3). The introduction closes with an outline of the dissertation (in Section 1.4).

1.2. Motivation

Robotic manipulators are articulated open chains of serially connected links. An n *degree-of-freedom* (DOF) manipulator has n independent joints and $n+1$ links. The i th joint connects the i-1 th and i th links and has one degree-of-freedom. The joints can either be prismatic or revolute. Actuation of a prismatic joint translates the link along the joint axis, while actuation of a revolute joint rotates the link about the joint axis. The base link, Link 0, is rigidly attached to a mounting surface and the end-effector, Link n, is free to move in accordance with the actuation of the n joints. Industrial manipulators are currently used primarily as positioning devices (e.g., for pick and place operations). In these applications, accuracy of the Cartesian position and orientation of the end-effector is the control objective.

The design and implementation of robot manipulators require planning of the desired trajectory, followed by analysis of the kinematic and dynamic characteristics to develop a control system [18]. Trajectory planners operate in one of two modes depending upon whether the robot is programmed with respect to a coordinate frame or programmed in a move-and-teach scenario. When they are programmed with respect to a coordinate frame, heuristics are applied to compute temporal sequences of the end-effector's Cartesian position and orientation from high-level task descriptions. In a move-and-teach scenario, curve-fitting algorithms are used to compute a joint space trajectory from a series of manually-recorded robot configurations. Kinematic control, which is the solution of the *inverse kinematic problem* (IKP), computes the set of joint positions from the desired Cartesian position of the end-effector. Dynamic control computes the control forces and/or torques, which are required to produce the motion specified by the joint space trajectories. The performance of robotic manipulators depends upon all three of these factors. This dissertation focuses upon the kinematic modeling, identification, and control of industrial robotic manipulators for improved positioning accuracy. The design and analysis of trajectory planners and dynamic controllers are beyond the scope of this dissertation.

Kinematic models are required for the analysis and design of robot controllers. Kinematic models describe the *static* relationships between the joint positions and the Cartesian position and orientation of the end-effector. These relationships are usually expressed in terms of parametric relations among joint positions and orientations [5, 20, 26, 32]. Increasing the number of revolute joints in a manipulator dramatically increases the complexity of the kinematics. The physical interpretation and systematic structure of the *Denavit-Hartenberg* model [5] have led to its widespread use in robot control. The inverse kinematic problem described above uses the parametric kinematic model to compute the required joint positions for positioning of the robot arm.

Conventional robot positioning systems rely on the kinematic model to predict the end-effector position and orientation when only the joint positions are known. Whereas sensory feedback techniques are more appealing, the real-time measurement of the end-effector's position and orientation is impractical. In practice, engineers neglect gear backlash, friction, link compliance, encoder resolution, joint wobble, and manufacturing errors in formulating a kinematic robot model. Standard practice in robot control is to use the kinematic parameters specified in the robot manufacturer's design, rather than the actual kinematic parameters that emerge from the manufacturing process. This approach simplifies the modeling task and facilitates the implementation of *ideal* inverse kinematic algorithms for control. In particular, use of an idealized model leads to a closed-form inverse kinematic solution. This dissertation describes a method for identifying the kinematic parameters of robotic manipulators, evaluates an implementation of the method, and considers the overall improvement in end-effector positioning and orienting accuracy.

In the past, positioning errors introduced by the mismatch between the actual kinematics and the kinematic model have been overshadowed by more fundamental inadequacies of robotic systems. Over the past decade, however, marked improvements have been made in the accuracy, reliability, and dynamic performance of robots. There now exists a need to develop kinematic control algorithms to improve robot performance [23]. The fact that engineers often program robots in a move-and-teach mode, thereby circumventing the kinematic controller, to increase positioning accuracy reaffirms this realization and provides further motivation for this dissertation. Interest in the kinematic modeling problem has led to the analysis of kinematic modeling errors and a number of schemes have been proposed for their identification [4, 8, 13, 14, 15, 30, 31]. Most of these schemes have not been implemented and evaluated.

1.3. Dissertation Goals and Contributions

This dissertation describes the development of new kinematic robot models and kinematic parameter identification algorithms for the design of controllers to improve robot kinematic performance. These models and algorithms have been implemented and evaluated on standard industrial robots. The effects of manufacturing errors upon robot kinematic behavior and the design of kinematic control algorithms for industrial robots have been analyzed using simulation tools and compared with experimental results.

This dissertation contributes to four areas of robotics:

- *Kinematic Modeling* used for the analysis and design of robot controllers. Our research in this area has led to the development of a new robot kinematic model, called the S-Model, which is directly applicable to the kinematic parameter identification problem.
- *Identification* of robot kinematic parameters. Our research in this area has led to the development of a practical kinematic parameter identification algorithm called the S-Model identification algorithm.
- *Kinematic Performance Evaluation* for the assessment and specification of robot kinematic accuracy.
- *Kinematic Control* for the design and evaluation of recursive control algorithms for robotic manipulators.

We anticipate that our research will impact both the mechanical design and manufacture of robots and the implementation of advanced control algorithms for industrial manipulators. We have shown that implementation of our proposed identification and control algorithms can increase robot performance and thereby expand the range of robotic applications.

1.4. Dissertation Outline

This dissertation is organized as follows. In Chapter 2, we review robot kinematics, identification, and control. In Chapter 3, we introduce the new kinematic model, the S-Model, and derive its relationship to the Denavit-Hartenberg model. Then, in Chapter 4, we apply the S-Model to formulate the robot arm signature identification algorithm. In Chapter 5, we describe two algorithms for solving the inverse kinematic problem and evaluate both their computational complexity and numerical accuracy. The software and hardware implementation of the S-Model identification algorithm and the subsequent control performance evaluation is presented in Chapter 6. In Chapter 7 we use Monte-Carlo simulation techniques to evaluate the statistical performance properties of the S-Model identification algorithm. Chapters 3 through 7, which highlight our research, are the major contributions of this dissertation. Finally, in Chapter 8, we summarize our contributions and identify areas for further research.

Chapter 2

Review of Robot Kinematics, Identification, and Control

2.1. Overview

In this chapter, we review the formulation of robot kinematic models and establish robot kinematic control as a framework for the development of the dissertation. The fundamental problem in the development of robot kinematic models is the use of geometric and trigonometric principles to systematically specify the relative positions and orientations of robot joints. Approaches to this problem are reviewed in this chapter. Standard terminology is used throughout the dissertation.

2.2. Coordinate Frame Kinematic Models

Coordinate frame kinematic models are based, conceptually, upon the assignment of Cartesian coordinate frames fixed relative to each of the links. The spatial transformation between two consecutive link coordinate frames is a function of the position of the joint which connects the two links together. In robotics, the (4x4) *homogeneous* transformation matrix introduced by Denavit and Hartenberg [5] and later adopted by Pieper [20] and Paul [18] has become the most common approach to describing these spatial transformations. A general homogeneous transformation matrix has the form

$$\begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.1}$$

or, in terms of its vector components,

$$\begin{bmatrix} \vec{n} & \vec{o} & \vec{a} & \vec{p} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

where

$$\vec{n} = [n_x\, n_y\, n_z]^T \;, \tag{2.3}$$

$$\vec{o} = [o_x\, o_y\, o_z]^T \;, \tag{2.4}$$

$$\vec{a} = [a_x\, a_y\, a_z]^T \;, \text{and} \tag{2.5}$$

$$\vec{p} = [p_x\, p_y\, p_z]^T \;. \tag{2.6}$$

Consider the two arbitrary cartesian coordinate frames depicted in Figure 2-1. If the matrix[1] **C** defines the spatial transformation from coordinate frame 1 to coordinate frame 2, then the unit direction vectors $\vec{n}$, $\vec{o}$, and $\vec{a}$ specify the orientation of the X, Y, and Z axes of coordinate frame 2 in terms of coordinate frame 1, respectively. The vector $\vec{p}$ specifies the position of the origin of coordinate frame 2 with respect to coordinate frame 1.

The (3x3) orthogonal rotation matrix $\mathbf{R}_{(3x3)}$ formed by $\vec{n}$, $\vec{o}$, and $\vec{a}$ is the orientational component of (2.1) and $\vec{p}$ is the translational component of (2.1). The coordinates of a point P in frame 1, $\vec{x}_1 = [x_1\; y_1\; z_1\; 1]^T$, expressed in terms of its coordinates relative to coordinate frame 2, $\vec{x}_2 = [x_2\; y_2\; z_2\; 1]^T$, is

[1]Uppercase boldface letters (e.g., $\mathbf{T}_i$ and $\mathbf{S}_i$) denote (4x4) homogeneous transformation matrices.

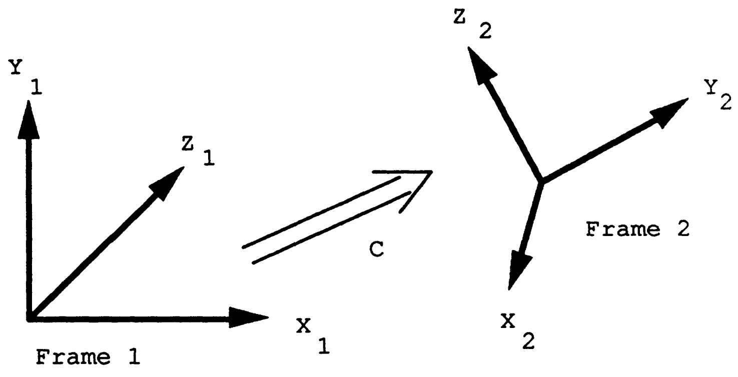


Figure 2-1: Two Arbitrary Cartesian Coordinate Frames

$$\vec{x}_1 = C\vec{x}_2 \tag{2.7}$$

The homogeneous transformation matrix (2.1) has a variety of mathematical properties which are used extensively in robotics and are presented in [18].

The transformation between any two cartesian coordinate frames can always be decomposed into a combination of *primitive* transformations. Using Paul's notation, the *six* primitive transformations are

- Trans(x,0,0) – Translate x units along the X axis.
- Trans(0,y,0) – Translate y units along the Y axis.
- Trans(0,0,z) – Translate z units along the Z axis.
- Rot(x, θ) – Rotate theta degrees about the X axis.
- Rot(y, θ) – Rotate theta degrees about the Y axis.
- Rot(z, θ) – Rotate theta degrees about the Z axis.

Paul's notation will be used throughout this dissertation. The elements of the primitive transformations are evaluated in Appendix A to indicate their structure.

2.2.1. Denavit-Hartenberg Model

The *Denavit-Hartenberg* model [5]

$$\mathbf{T}_n = \mathbf{A}_1 \cdot \mathbf{A}_2 \cdot \ldots \cdot \mathbf{A}_n \tag{2.8}$$

has become the standard robot kinematic model because of its physical interpretation, strict definition, and multiplicative structure. In (2.8), the (4x4) homogeneous transformation matrix $\mathbf{T}_n$ defines the position and orientation of a coordinate frame fixed relative to the last link (n th link or end-effector) of a manipulator with respect to a coordinate frame fixed relative to the base of the manipulator. Conceptually, the right-hand side of (2.8) describes the spatial relationships between coordinate frames fixed relative to each of the manipulator links. The Denavit-Hartenberg link coordinate frames, $\mathcal{T}_i$[2] for $i=1, \ldots ,n$, are specified such that the *forward transformation matrices* $\mathbf{A}_i$ are prescribed by

$$\begin{aligned}\mathbf{A}_i = \mathbf{A}_i(\mathbf{q}_i) \equiv\ & Rot(z, \theta_i) Trans(0, 0, \mathbf{d}_i) Trans(\mathbf{a}_i, 0, 0) \\ & Rot(x, \alpha_i) \ .\end{aligned} \tag{2.9}$$

The input parameters to the model (2.8) are the n generalized joint coordinates q_i . The generalized coordinates are used to represent the joint positions without explicitly specifying the physical nature of the joint (i.e., whether revolute or prismatic). Expanding (2.9) yields

[2]Uppercase script letters (e.g., $\mathcal{T}_i$ and $\mathcal{S}_i$) denote the symbolic name of a cartesian coordinate system.

$$\mathbf{A}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \cos\alpha_i & \sin\theta_i \sin\alpha_i & a_i \cos\theta_i \\ \sin\theta_i & \cos\theta_i \cos\alpha_i & -\cos\theta_i \sin\alpha_i & a_i \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

In (2.9), the transformation $\mathbf{A}_i$ from coordinate frame $\mathcal{T}_{i-1}$ to coordinate frame $\mathcal{T}_i$ is a function of the *four* Denavit-Hartenberg parameters, θ_i, $\mathbf{d}_i$, $\mathbf{a}_i$, and α_i[3]. An n degree-of-freedom manipulator requires the specification of $4 \cdot n$ parameters, n of which are the controllable joint positions. Geometrically, the link length, $\mathbf{a}_i$, is the length of the common normal between the joint i and joint $i+1$ axes. The link twist, α_i, is the angle between the joint i and joint $i+1$ axes measured in the plane perpendicular to the common normal to the joint axes. The parameter θ_i is the angle, measured in the plane perpendicular to the joint i axis, between the joint axis common normals of the i-1 th and i th link. The fourth parameter, $\mathbf{d}_i$, is the linear displacement between the intersections of the link i-1 and i axis common normals with the joint i axis. Figures 2-2 and 2-3 illustrate the physical definition of these parameters for both revolute and prismatic joints, respectively.

The characteristics of the Denavit-Hartenberg model are immediate consequences of the Denavit-Hartenberg convention applied to specify the link coordinate frames. The Denavit-Hartenberg convention follows from a geometrical analysis of the spatial relationships between consecutive joint axes. The Denavit-Hartenberg convention specifies the following link coordinate frame assignments:

- The Z axis of coordinate frame $\mathcal{T}_{i-1}$ must be parallel to the joint i axis.

[3]Lowercase letters (e.g., $\mathbf{d}_i$ and β_i) denote scalar parameters.

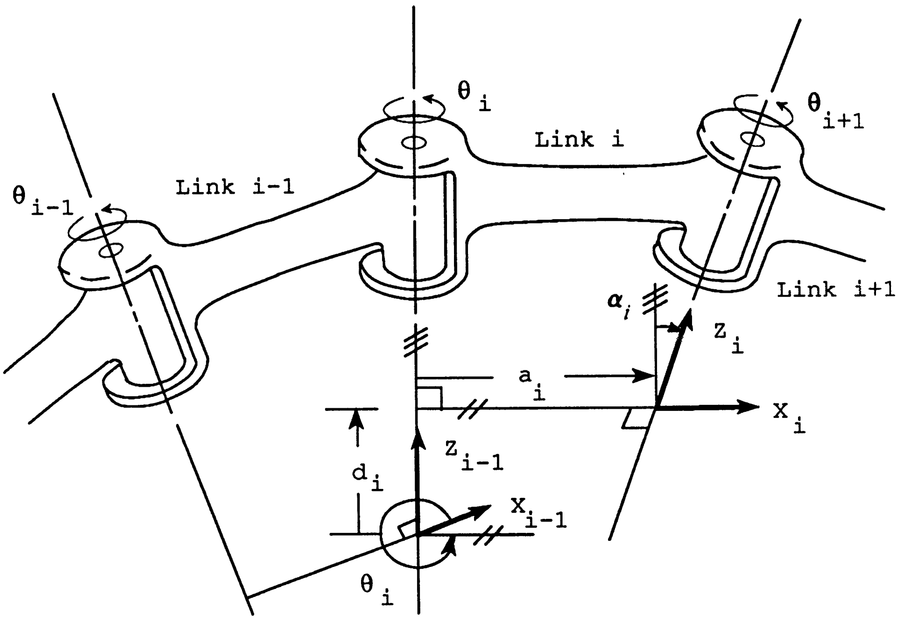


Figure 2-2: Denavit-Hartenberg Parameters for a Revolute Joint. Reprinted with permission [18].

- The origin of coordinate frame $\mathcal{T}_{i-1}$ must lie on the joint *i* axis at the intersection point of the common normal between the joint *i-1* and joint *i* axes, and the joint *i* axis.
- The X axis of coordinate frame $\mathcal{T}_{i-1}$ must be parallel to the common normal between the joint *i-1* and joint *i* axes. The positive direction of the X axis points towards the joint *i* axis.
- The Y axis of coordinate frame $\mathcal{T}_{i-1}$ is defined by the vector cross product of the Z axis unit direction vector with the X axis unit direction vector.
- If the joint *i* and joint *i+1* axes intersect, the point of intersection is the origin of the $\mathcal{T}_{i-1}$ coordinate frame.

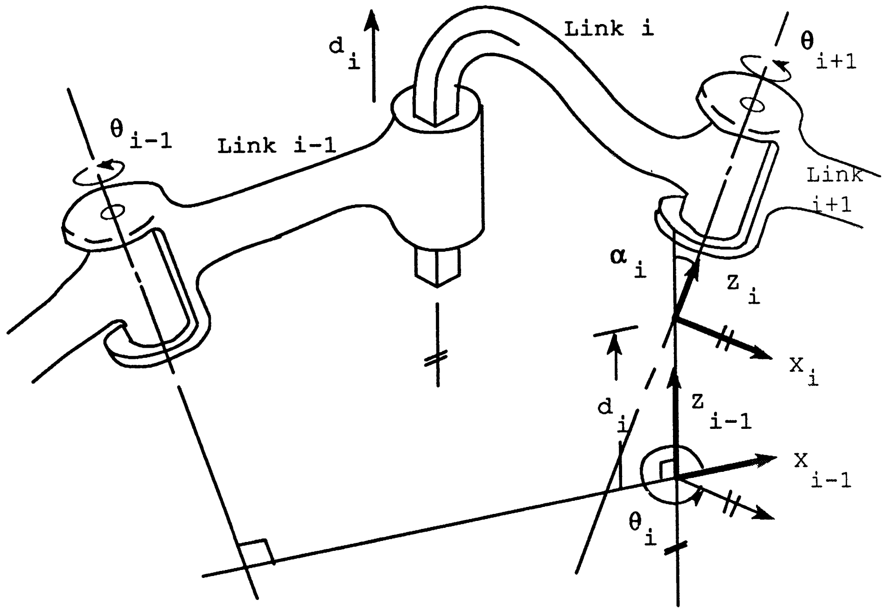


Figure 2-3: Denavit-Hartenberg Parameters for a Prismatic Joint. Reprinted with permission [18].

- If the joint *i* and joint *i+1* axes are parallel the origin of the coordinate frame, $\mathcal{T}_{i-1}$ is chosen so that the joint distance d_{i+1} for the next link is equal to zero.
- The origin of the base link coordinate frame, $\mathcal{T}_0$, coincides with the origin of the link 1 coordinate frame $\mathcal{T}_1$.
- The origin of the last coordinate frame, $\mathcal{T}_n$, coincides with the origin of the next to last coordinate frame $\mathcal{T}_{n-1}$.

These assignments guarantee the functional form of the Denavit-Hartenberg model in (2.8). By using different conventions to specify the link coordinate frames, alternative kinematic models with the same multiplicative structure as in (2.8) can be formulated. For robots with rigid links and single degree-of-freedom joints, the model in (2.8) is exact.

For a *revolute* joint, the generalized coordinate q_i is the joint i position θ_i and the three parameters, $\mathbf{d}_i$, $\mathbf{a}_i$, and α_i are constants. For a *prismatic* joint, q_i is the joint i position $\mathbf{d}_i$ and the three parameters θ_i, $\mathbf{a}_i$, and α_i are constants. In practice, the joint encoders of a manipulator are typically calibrated such that the encoder outputs match the Denavit-Hartenberg joint positions (i.e., θ_i for a revolute joint and $\mathbf{d}_i$ for a prismatic joint). Without this calibration, constant offsets must be introduced to specify the difference between the joint positions measured by the encoder and the joint positions defined by the Denavit-Hartenberg model. When all of the joint positions are *zero*, we say that the manipulator is in the *Denavit-Hartenberg Zero Configuration.*

From a modeling point of view, the Denavit-Hartenberg model has at least one potential disadvantage. It can be demonstrated [15, 30] that for some manipulator geometries the definition of the parameter $\mathbf{d}_i$ may cause elements of the $\mathbf{A}_i$ matrices to be extremely large and hence ill-conditioned. We have observed such ill-conditioned $\mathbf{A}_i$ matrices with very large elements in our identification of real robot kinematic model parameters. This situation can lead to numerical instabilities depending upon the application and the available numerical precision. The ramifications of potentially ill-conditioned transformation matrices upon the kinematic parameter identification problem are discussed in Chapter 4.

The Denavit-Hartenberg parameters are defined according to a conceptual model of the kinematics rather than a purely physical model. Thus, the link coordinate frames $\mathcal{T}_i$ may be located at a point external to the physical links of the manipulator. This is often the case for manipulators with near parallel consecutive joint axes.

2.2.2. Whitney-Lozinski Model

In formulating an approach to the kinematic identification problem, Whitney and Lozinski [30] use the model

$$\mathbf{T}_n = \mathbf{W}_1 \cdot \mathbf{W}_2 \cdot \ldots \cdot \mathbf{W}_n \ , \tag{2.11}$$

where

$$\begin{aligned}\mathbf{W}_i = \mathbf{W}_i(\mathbf{q}_i) \equiv\ & Rot(y,\theta_i) Trans(0,\mathbf{y}_i,\mathbf{z}_i) Rot(z,\phi_i) \\ & Rot(y,\Omega_i) Rot(x,\psi_i) \ ,\end{aligned} \tag{2.12}$$

to describe the rigid body geometric component of the kinematics[4]. The final three transformations in (2.12) constitute a general *Roll-Pitch-Yaw* (RPY) rotational transformation. Expanding (2.12) yields

$$\mathbf{W}_i = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

where

$$n_x \equiv \cos\theta_i \cos\phi_i \cos\Omega_i - \sin\theta_i \sin\Omega \ , \tag{2.14}$$

$$n_y \equiv \sin\phi_i \cos\Omega_i \ , \tag{2.15}$$

$$n_z \equiv -\sin\theta_i \cos\phi_i \cos\Omega_i - \cos\theta_i \sin\Omega_i \ , \tag{2.16}$$

$$\begin{aligned} o_x \equiv\ & \cos\theta_i [\cos\phi_i \sin\Omega_i \sin\psi_i - \sin\phi_i \cos\psi_i] + \\ & \sin\theta_i [\cos\Omega_i \sin\psi_i] \ ,\end{aligned} \tag{2.17}$$

$$o_y \equiv \sin\phi_i \sin\Omega_i \sin\psi_i + \cos\phi_i \cos\psi_i \ , \tag{2.18}$$

$$\begin{aligned} o_z \equiv\ & -\sin\theta_i [\cos\phi_i \sin\Omega_i \sin\psi_i - \sin\phi_i \cos\psi_i] + \\ & \cos\theta_i [\cos\Omega_i \sin\psi_i] \ ,\end{aligned} \tag{2.19}$$

[4]The complete model used by Whitney and Lozinski for identification also contains a non-geometric component.

$$\begin{aligned}
a_x &\equiv \cos\theta_i[\cos\phi_i\sin\Omega_i\cos\psi_i+\sin\phi_i\sin\psi_i] + \\
&\qquad \sin\theta_i[\cos\Omega_i\cos\psi_i] \ , && (2.20) \\
a_y &\equiv \sin\phi_i\sin\Omega_i\cos\psi_i - \cos\phi_i\sin\psi_i \ , && (2.21) \\
a_z &\equiv -\sin\theta_i[\cos\phi_i\sin\Omega_i\cos\psi_i+\sin\phi_i\sin\psi_i] + \\
&\qquad \cos\theta_i[\cos\Omega_i\cos\psi_i] \ , && (2.22) \\
p_x &\equiv z_i\sin\theta_i \ , && (2.23) \\
p_y &\equiv y_i \ \text{,and} && (2.24) \\
p_z &\equiv z_i\cos\theta_i \ . && (2.25)
\end{aligned}$$

In (2.12), the transformation $\mathbf{W}_i$ from coordinate frame $\mathcal{T}_{i-1}$ to coordinate frame $\mathcal{T}_i$ is a function of the *six* parameters, θ_i, y_i, z_i, ϕ_i, Ω_i, and ψ_i. An n degree-of-freedom manipulator requires the specification of $6 \cdot n$ parameters, n of which are the controllable joint positions. In contrast to the Denavit-Hartenberg model which requires four parameters per joint, the model in (2.11) is a non-minimum realization requiring six parameters per joint.

For a *revolute* joint, the generalized coordinate q_i is the joint i position θ_i and the five parameters, y_i, z_i, ϕ_i, Ω_i, and ψ_i are constants. For a *prismatic* joint, q_i is the joint i position y_i and the five parameters θ_i, z_i, ϕ_i, Ω_i, and ψ_i are constants. For a prismatic joint, the parameter θ_i can be arbitrarily set to zero. Like (2.8), the model in (2.11) is exact for robots with rigid links and perfect joints. With two additional parameters for each link transformation matrix, the model in (2.11) conceptually introduces a greater degree of flexibility in assigning the locations of the link coordinate frames W_i. This flexibility is reflected in the rules applied to specify the locations of the link coordinate frames, $\mathcal{T}_i$:

- The Y axis of the link coordinate frame $\mathcal{T}_{i-1}$ must be parallel to the joint i axis in the direction defined by the positive sense of the rotation or translation of the i th joint.
- The origin of the coordinate frame $\mathcal{T}_{i-1}$ must lie on the joint i axis.

- The Z axis of the link coordinate frame $\mathcal{T}_{i-1}$ must point in the direction such that the origin of coordinate frame $\mathcal{T}_i$ lies in the plane formed by the Y and Z axis of coordinate frame $\mathcal{T}_{i-1}$ when joint i is in its zero position.
- The origin of coordinate frame $\mathcal{T}_n$ lies on the joint n axis.

The parameters of the model (2.11) have no real physical significance. Consequently, the model (2.11) lacks the apparent elegance of the Denavit-Hartenberg model. The expression (2.12) for the link transformation matrix $\mathbf{W}_i$ is considerably more complex than the link transformation matrix $\mathbf{A}_i$ in (2.10).

2.3. Models of Revolute Joint Manipulators

In contrast to *coordinate frame* kinematic models, the models introduced for *rotary* manipulators by Mooring [15], Sugimoto and Duffy [25], and Suh and Radcliffe [26] find their roots in the *theory of screws* [2]. These models possess the same multiplicative structure as in (2.8) and (2.11). The Cartesian position and orientation of the end-effector with respect to the base coordinate frame is specified by the transformation matrix

$$\mathbf{T}_n = \mathbf{D}_1 \cdot \mathbf{D}_2 \cdot \ldots \cdot \mathbf{D}_n \cdot \mathbf{T}_n^0 \tag{2.26}$$

where the constant matrix $\mathbf{T}_n^0$ is the location of the end-effector when all of the joint positions are zero. For the model (2.26), the *zero configuration* can be associated with any arbitrary physical configuration of the manipulator. The simplest approach is to let the zero positions of the joints correspond directly to the physical joint encoder readings. In the literature [15], the homogeneous transformation matrices $\mathbf{D}_i$ are called *displacement matrices* and are functions of the joint positions

$$\mathbf{D}_i = \mathbf{D}_i(\theta_i) \ . \tag{2.27}$$

A displacement matrix specifies the spatial transformation undergone by a point when it is rotated about an axis in space. The general form of the displacement matrix is

$$\mathbf{D}_i = \begin{bmatrix} u_x^2 V\theta + C\theta & u_x u_y V\theta - u_z S\theta & u_x u_z V\theta + u_y S\theta & d_{14} \\ u_x u_y V\theta + u_z S\theta & u_y^2 V\theta + C\theta & u_y u_z V\theta - u_x S\theta & d_{24} \\ u_x u_z V\theta - u_y S\theta & u_y u_z V\theta + u_x S\theta & u_z^2 V\theta + C\theta & d_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.28}$$

where

$$d_{14} = (1-d_{11})p_x - d_{12}p_y - d_{13}p_z \ , \tag{2.29}$$

$$d_{24} = -d_{21}p_x + (1-d_{22})p_y - d_{23}p_z \ , \tag{2.30}$$

$$d_{34} = -d_{31}p_x - d_{32}p_y + (1-d_{33})p_z \ , \tag{2.31}$$

and d_{jk} is the (j,k) element of the displacement matrix **D**, $S(\theta) = \sin\theta_i$, $C(\theta) = \cos\theta_i$, and $V(\theta) = 1 - \cos\theta_i$. The matrix $\mathbf{D}_i$ is parameterized by the *six* components, u_x, u_y , u_z , p_x , p_y , and p_z of the vectors $\vec{u}_i$ and $\vec{p}_i$. In (2.26), the unit direction vector $\vec{u}_i$ points along the joint i axis of rotation and is referenced with respect to an arbitrarily located base coordinate frame. The vector $\vec{p}_i$ specifies the location of an arbitrary point on the joint i axis with respect to the base frame. Together the two vectors, $\vec{u}_i$ and $\vec{p}_i$, locate the joint axis in space.

2.4. Modeling Assumptions

The accuracy of a mathematical model of a manipulator is determined by the validity of the assumptions upon which it is formulated. These assumptions are introduced to insure the tractability of the modeling task. In practice, however, these assumptions are not always satisfied. For robot control, the resulting inaccuracies give rise to discrepancies between the predicted and actual end-effector position. The kinematic models reviewed in Sections 2.2 and 2.3 are based upon the following seven assumptions:

- Link compliance is negligible (A-1).
- Gear train compliance is negligible (A-2).
- Motor-bearing wobble is negligible (A-3).
- Gear backlash is negligible (A-4).
- Link deformation, due to such environmental effects as temperature variations, is negligible (A-5).
- Encoder resolution is infinite (A-6).
- The kinematic parameters of the actual manipulator are known exactly (A-7).

The difference between the predicted and the actual end-effector positions can be reduced by ensuring the validity of underlying modeling assumptions. Reduction of this error is critical to improving the kinematic performance of robotic manipulators. Two approaches are commonly taken to reduce manipulator-model mismatch. The model can be expanded to incorporate previously unmodeled features or the manipulator can be modified to more closely resemble the desired model. For the kinematic modeling and control of robots, a combination of these approaches appears to be reasonable.

The complexity in modeling link compliance, gear train compliance, bearing wobble, gear backlash, gear friction, link deformation, and encoder resolution suggests that to incorporate these features into a kinematic model would be impractical, if not infeasible. Furthermore, since an extremely wide range of mechanical components are used in the design of manipulator drive mechanisms, there is little hope that a general model or set of models could be developed to adequately describe these effects for an arbitrary manipulator. Commercially, a case by case analysis of these effects on each individual manipulator after it is manufactured would be too costly.

Recent advances in robot actuator and sensor technology and composite materials for robotic applications improve the validity of many of these assumptions. Determination of the actual kinematic parameters is a different issue. Manufacturing errors introduce errors between the actual kinematic parameters and the design parameters. Unfortunately for many industrial robots, economic reality precludes the further reduction of manufacturing tolerances. It thus becomes essential to expand the kinematic model to account for these manufacturing errors. We must identify either the manufacturing errors or the actual kinematic parameters. The identification algorithm developed in this dissertation employs the latter approach. The observation that the actual kinematic parameters of a manipulator can vary significantly from the design parameters due to the presence of manufacturing errors is the principal motivation for the development of this dissertation. The dissertation relies on Assumptions (A-1) - (A-6).

2.5. Kinematic Identification

The goal of a kinematic identification algorithm is to identify the parameters of a kinematic model which describe the actual position and orientation of the end-effector in terms of the measured joint positions, and which incorporates the geometrical variations in the structure caused by manufacturing errors. Several identification algorithms have been proposed in the literature [4, 13, 14, 15, 30], but to our knowledge only the algorithm proposed by Whitney and Lozinski [30] has been actually implemented and evaluated.

The algorithm developed by Whitney and Lozinski is designed to identify both geometric and nongeometric parameters. The geometric parameters correspond to the kinematic parameters of a rigid body description of the robot while the nongeometric parameters represent compliance, gear transmission errors, and backlash. The model (2.11), presented in Section 2.2.2, constitutes their description of the geometrical portion of the complete robot model. The mathematical form of the non-geometrical component is derived separately from experiments performed on the robot. The non-geometrical model maps measured joint positions into actual joint positions. In the example cited, the gear transmission error for one of the joints is modeled as a sinusoid. The geometrical and non-geometrical models are combined to form the complete robot kinematic model.

In this approach, a theodolite is used to measure the position of the robot's end-effector corresponding to various joint configurations. Additional parameters are introduced into the robot model in order to relate the Cartesian position of the end-effector to the angular coordinates of the theodolite. For instance, the coordinates defining the position and orientation of the theodolite, relative to the robot, must be included into the new model. The model parameters are estimated by minimizing the cost function

$$\Xi = \sum_{i=1}^{n} (\vec{z}_i - \vec{f}(\vec{\alpha}_i, \vec{\phi}))^T (\vec{z}_i - \vec{f}(\vec{\alpha}_i, \vec{\phi})) \tag{2.32}$$

where $\vec{z}_i$ is the vector of measured theodolite angular readings, $\vec{f}(\vec{\alpha}_i, \vec{\phi})$ is the vector of predicted theodolite angular readings, $\vec{\alpha}_i$ is the vector of joint encoder readings, $\vec{\phi}$ is the vector of model parameters, and n is the number of configurations at which measurements are made. A nonlinear least-squares numerical search algorithm is applied to minimize (2.32).

The authors have implemented their algorithm and have applied it to improve the kinematic performance of a Puma 560 robot. Twenty eight geometric parameters and eight non-geometric parameters were identified using 60 sets of theodolite readings. It appears, however, that in order for the numerical search to converge, certain parameters within the original model had to be set to predefined values. Nevertheless substantial improvements in performance were obtained. For instance, in one particular case the relative positioning error of the end-effector was reduced from 4.8 mm to .3 mm.

The identification algorithm developed by Whitney and Lozinski has several disadvantages.

- The nonlinear minimization algorithm used to estimate the model parameters is not necessarily guaranteed to converge.
- The process of measuring the position of the end-effector is extremely time-consuming and requires a highly skilled operator.
- Analytic models of the non-geometric errors must be developed for each individual robot.
- There is no mechanism for determining at which joint configurations the end-effector's position should be measured to obtain the best estimates.

- Separate procedures must be used to establish a length standard since a theodolite only measures angular displacements.

Further analysis and evaluation will be required to fully assess the capabilities and limitations of this approach.

Mooring [15] has proposed a method for identifying the kinematic parameters of a revolute joint manipulator using the model (2.26). The basis for his algorithm lies in the form of the transformation matrix $\mathbf{D}_i$ in (2.28). If the position of joint i is zero, the transformation matrix $\mathbf{D}_i$ becomes the identity matrix regardless of the values of $\vec{u}$ or $\vec{p}$ in (2.28). This property provides a mechanism to partially decouple the identification problem. If, with the exception of joint i, the positions of all the joints are zero, then the matrix $\mathbf{T}_n$ will be equal to the matrix $\mathbf{D}_i$.

The first step of the identification procedure is to move all joints to their zero positions. Then, the positions of three points on the end-effector must be measured and the position of an arbitrary fourth point computed. The measured positions are combined to form the matrix of measured positions,

$$\mathbf{X}_1 = \begin{bmatrix} x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} \\ y_{1,1} & y_{1,2} & y_{1,3} & y_{1,4} \\ z_{1,1} & z_{1,2} & z_{1,3} & z_{1,4} \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{2.33}$$

After rotating joint i to another position, the positions of the three points on the gripper again are measured, the position of the fourth point is computed, and the matrix $\mathbf{X}_2$ analogous to (2.33) is formed. The eight measurements correspond to the positions of the same four physical points on the end-effector and thus

$$\mathbf{X}_2 = \mathbf{D}_1 \cdot \mathbf{D}_2 \cdot \ldots \cdot \mathbf{D}_n \mathbf{X}_1 \ . \tag{2.34}$$

Since only joint i has a nonzero position, $\mathbf{D}_j$ in (2.34) is equivalent to $\mathbf{I}$ for $j \neq i$ and

$$\mathbf{D}_i = \mathbf{X}_2 \cdot \mathbf{X}_1^{-1} \ . \tag{2.35}$$

The elements of $\mathbf{D}_i$ in (2.35) can be applied to solve for the kinematic model parameters (i.e., the elements of the vectors $\vec{u}$ and $\vec{p}$). The process is then repeated for each of the remaining joints.

The scheme presented in [15] has neither been implemented nor evaluated using either simulated data or real data. A variation of this scheme was tested using simulated data and was shown to identify the robot's true kinematic parameters within "acceptable limits" in [15]. However, this paper did not discuss whether or not the simulated end-effector measurements were corrupted with measurement noise and/or how measurement noise effects the accuracy of the identified parameters. Further analysis is required before a judgement as to its potential for improving the kinematic performance of robots can be made. The development of a sensor system which can accurately measure the positions of the three points on the robot's end-effector over a large volume appears to be a major factor in determining the feasibility and practicality of this approach.

2.6. Kinematic Control

The kinematic control problem focuses upon the computation of the joint positions required to locate the end-effector at a desired Cartesian position and orientation. Since feedback, which requires the measurement of the Cartesian position and orientation of the end-effector, is often infeasible, we must implement *open-loop* feedforward control. In the design of feedforward control algorithms for industrial robots, we must address the trade-off between algorithm complexity and controller performance. Since the control algorithms for robots are derived directly from the forward kinematic model, their complexity increases as the complexity of the forward kinematic model increases. Incorporating the effects of manufacturing errors into a kinematic model will lead to increased robot performance at the expense of controller complexity.

The design of kinematic control algorithms for industrial robots involves:

- Formulation of a kinematic model
- Inversion of the forward kinematic model
- Implementation of the inverse kinematic algorithms.

This straightforward approach leads to a relatively simple control algorithm provided that the initial kinematic model possesses a *simple structure.* Kinematic models with closed-form inverses are defined as *simple structures.* The fact that relatively few of the possible manipulator configurations possess a closed-form inverse kinematic model has had a strong influence upon the mechanical design and kinematic modeling of robotic manipulators. For instance, basic geometrical features, such as parallel or perpendicular joint axes, are incorporated into the mechanical design of a robot to guarantee the existence of a simply-structured kinematic model. In the design of controllers for these robots, it is conveniently assumed that the manipulator has been manufactured with negligible machining and assembly errors. For high precision applications, the failure of this assumption often accounts for the observed end-effector positioning errors.

The most difficult task in the design of kinematic control algorithms has been the formulation of the inverse model. If the Denavit-Hartenberg formulation is applied to model the kinematics, the ***backward multiplication technique*** of Paul [18] can be applied to derive the inverse kinematics of simply-structured manipulators. Paul's backward multiplication technique has contributed significantly to the widespread appeal of the Denavit-Hartenberg formulation. Consider the model in (2.8) where the elements of the matrix $\mathbf{T}_n$, the desired position and orientation of the end-effector, are known. Premultiplying both sides of (2.8) yields

$$\mathbf{A}_1^{-1} \cdot \mathbf{T}_n = \mathbf{A}_2 \cdot \mathbf{A}_3 \cdot \ldots \cdot \mathbf{A}_n \equiv \mathbf{U}_2 \tag{2.36}$$

The left-hand side of (2.36) is a function of the unknown joint position q_1 and known elements of $\mathbf{T}_n$. (The inverse of the homogeneous transformation matrix $\mathbf{A}_1$ can be expressed analytically in closed form.) The objective in Paul's technique is to symbolically expand both sides of (2.36) and to equate elements on the left-hand side which involve q_1 with elements on the right-hand side which are independent of the remaining joint positions. The resulting set of equations are then solved to determine the joint position q_1. The next step uses the solution q_1 to evaluate the left-hand side of (2.36) which thus becomes a matrix of known constants. Similarly, by premultiplying (2.36) by the inverse of $\mathbf{A}_2$, we obtain

$$\mathbf{A}_2^{-1} \cdot \mathbf{A}_1^{-1} \cdot \mathbf{T}_n = \mathbf{A}_3 \cdot \mathbf{A}_4 \cdot \ldots \cdot \mathbf{A}_n \equiv \mathbf{U}_3 \ . \tag{2.37}$$

The situation is analogous to that in (2.36). The left-hand side of (2.37) is now only a function of the unknown position q_2 and known constants. Matrix element equality is then employed to isolate and solve for q_2. This procedure is repeated to sequentially solve for the remaining unknown joint positions. For revolute joints, the equations involving q_i are trigonometric. The arc tangent function which has two arguments, the ordinate y, and the abscissa x should be used to solve for q_i as opposed to the arc sine or arc cosine functions. The accuracy of the arc tangent function is uniform over its full range of definition [18]. Based upon the sign of its arguments, the arc tangent function returns an angle in the interval $-\pi$ to π.

The premultiplication procedure described above will require modification if two or more of the joint axes are parallel. For instance, if the joints *i* and *i+1* are parallel, the sum of the two joint positions q_i and q_{i+1} must first be determined in one step followed by determination of either q_i or q_{i+1} in the following step. Hence, after solving for q_{i-1}, we expand

$$\mathbf{A}_{i+1}^{-1} \cdot \mathbf{A}_i^{-1} \cdot \mathbf{A}_{i-1}^{-1} \cdot \ldots \cdot \mathbf{A}_1^{-1} \cdot \mathbf{T}_n = \mathbf{U}_{i+2} \tag{2.38}$$

The elements of $\mathbf{A}_{i+1}^{-1} \cdot \mathbf{A}_i^{-1}$ will be a function of the sum $(q_i + q_{i+1})$. After solving for this sum, other elements of (2.38) can be used to solve for the individual joint positions.

Most manipulators with revolute joints, especially anthropomorphic manipulators, can reach a desired end-effector location in one of several distinct joint configurations. The Puma 560, which has *six* revolute joints, has, in general, *eight* distinct solutions. The multiplicity of inverse kinematic solutions for a given end-effector location can be easily computed using Paul's technique. However, since the model (2.8) does not account for the mechanical joint limitations, not all the multiple solutions are physically achievable. The ability to reach a particular joint configuration can only be determined after the solution has been evaluated.

We indicated in Section 2.4 that kinematic models of increasing complexity and accuracy are needed to improve robot positioning performance. Since these models will not have closed-form inverses, numerical algorithms must be applied to solve the inverse kinematic problem. Application of these algorithms gives rise to such issues as the rate of convergence, convergence to a global versus a local minimum, and the feasibility of real-time implementation.

2.7. Conclusions

In this chapter, we have reviewed robot kinematic modeling, identification, and control techniques, and established a framework in which to present our research contributions.

We introduced the concept of *coordinate frame* kinematic models (based upon the conceptual notion of fixing Cartesian coordinate frames to the various links of a robot). The analytic properties and physical interpretation of the Denavit-Hartenberg model which has become widely used in both industry and academia for modeling robot kinematics were discussed. The model used by Whitney and Lozinski [30] and the *displacement matrix model* used to model the kinematics of revolute joint manipulators were also presented. We then delineated the engineering assumptions upon which these models are formulated. We reviewed the kinematic parameter identification algorithms proposed by Whitney and Lozinski [30] and Mooring [15], and outlined some of practical problems with each of these approaches. Finally, we formulated the robot kinematic control problem and reviewed the *Backward Multiplication Technique* developed by Paul [18].

In this dissertation, we:

- Develop (in Chapter 3) a new robot kinematic model, called the *S-Model,* whose analytic properties and conceptual formulation make it directly amenable to identification.
- Develop (in Chapter 4) the S-Model identification algorithm which can be applied to identify the kinematic parameters of any robotic manipulator with rigid links.
- Synthesize and evaluate (in Chapter 5) two methods for inverting identified arm signature models.
- Develop (in Chapter 6) a prototype arm signature identification system and apply it to significantly improve the performance of several standard robotic manipulators.
- Evaluate and compare *Chapter 7* the statistical performance of the *design model* based and *signature-based* approach to robot kinematic control.

Chapter 3
Formulation of the S-Model

3.1. Overview

In this chapter, we introduce the formulation and properties of a new kinematic model for describing robot kinematics. This kinematic model, which we call the S-Model, was designed to facilitate kinematic parameter identification. This model can be applied to model the kinematics of all robotic manipulators which satisfy assumptions (A-1) through (A-7) (refer to Chapter 2).

3.2. S-Model

Like the Denavit-Hartenberg model, the S-Model is a general method of describing and characterizing kinematics of robotic manipulators. In the S-Model, the matrix

$$\mathbf{S}_n = \mathbf{B}_1 \cdot \mathbf{B}_2 \cdot \ldots \cdot \mathbf{B}_n \tag{3.1}$$

defines the position and orientation of a coordinate frame fixed relative to the last link of a manipulator with respect to a coordinate frame fixed relative to the base link. The *general transformation matrices,* $\mathbf{B}_i$, in (3.1) are (4x4) homogeneous transformation matrices. The $\mathbf{B}_i$ and $\mathbf{S}_n$ matrices in (3.1) are analogous to the $\mathbf{A}_i$ and $\mathbf{T}_n$ matrices of the Denavit-Hartenberg model in (2.8). The symbolic name $\mathcal{S}_i$ signifies the i^{th} link coordinate frame defined by the S-Model. The transformation matrix, $\mathbf{B}_i$, describes the relative transformation between the $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$ coordinate frames (measured with respect to the $\mathcal{S}_{i-1}$ coordinate frame). In the S-Model, *six* parameters, β_i, $\bar{\mathbf{d}}_i$, $\bar{\mathbf{a}}_i$, $\bar{\alpha}_i$, γ_i, and $\mathbf{b}_i$, define the transformation matrix

$$\begin{aligned}\mathbf{B}_i \equiv\ & Rot(z,\beta_i)Trans(0,0,\bar{\mathbf{d}}_i)Trans(\bar{\mathbf{a}}_i,0,0)Rot(x,\bar{\alpha}_i)\\ & Rot(z,\gamma_i)Trans(0,0,\mathbf{b}_i)\ .\end{aligned} \tag{3.2}$$

Expanding (3.2) yields

$$\begin{bmatrix} n_x & o_x & \sin\beta_i \sin\alpha_i & \mathbf{b}_i \sin\beta_i \sin\alpha_i + \mathbf{a}_i \cos\beta_i \\ n_y & o_y & -\cos\beta_i \sin\alpha_i & -\mathbf{b}_i \cos\beta_i \sin\alpha_i + \mathbf{a}_i \sin\beta_i \\ \sin\alpha_i \sin\gamma_i & \sin\alpha_i \cos\gamma_i & \cos\alpha_i & \mathbf{b}_i \cos\alpha_i + \mathbf{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

where

$$n_x \equiv \cos\beta_i \cos\gamma_i - \sin\beta_i \cos\alpha_i \sin\gamma_i\ , \tag{3.4}$$
$$n_y \equiv \sin\beta_i \cos\gamma_i + \cos\beta_i \cos\alpha_i \sin\gamma_i\ , \tag{3.5}$$
$$o_x \equiv -\cos\beta_i \sin\gamma_i - \sin\beta_i \cos\alpha_i \cos\gamma_i\ , \text{and} \tag{3.6}$$
$$o_y \equiv -\sin\beta_i \sin\gamma_i + \cos\beta_i \cos\alpha_i \cos\gamma_i\ . \tag{3.7}$$

To specify the S-Model for an n degree-of-freedom manipulator thus requires $6 \cdot n$ parameters.

To insure that the manipulator's kinematics can be modeled by (3.1), we introduce an S-Model convention to define the allowable locations of the link coordinate frames. Because each link transformation matrix is specified by six parameters rather than by four, the S-Model convention is less restrictive than the Denavit-Hartenberg convention.

The following four assignments, which are a subset of the Denavit-

Hartenberg convention reviewed in Section 2.2.1, specify the locations of the S-Model link coordinate frames:

- The Z axis of the link coordinate frame, $\mathcal{S}_{i-1}$, must be parallel to the joint i axis in the direction defined by the positive sense of the rotation or translation of the i^{th} joint.
- The origin of the coordinate frame, $\mathcal{S}_{i-1}$, must lie on the joint i axis.
- The Z axis of the last coordinate frame, $\mathcal{S}_n$, is parallel to the Z axis of the next to the last coordinate frame, $\mathcal{S}_{n-1}$.
- The origin of the last coordinate frame, $\mathcal{S}_n$, lies on the joint n-1 axis.

There are two fundamental distinctions between the Denavit-Hartenberg link coordinate frames, $\mathcal{T}_i$, and the S-Model link coordinate frames, $\mathcal{S}_i$. First, in contrast to the origin of $\mathcal{T}_i$ which is fixed, the location of the origin of $\mathcal{S}_i$ on the joint i+1 axis is arbitrary. Second, the direction of the X axis of $\mathcal{S}_i$ must only be orthogonal to the Z axis. The arbitrary location of the origin of $\mathcal{S}_i$ along the joint axis and the arbitrary orientation of the X axis of $\mathcal{S}_i$ provide an infinite number of link coordinate frames, $\mathcal{S}_0$ through $\mathcal{S}_n$, which satisfy the S-Model convention.

The transformation matrix, $\mathbf{B}_i$, can be formulated from the geometry of either Figure 3-1 or Figure 3-2. In Figures 3-1 and 3-2, we apply the S-Model convention to define a pair of link coordinate frames, $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$. For comparison, we also depict the Denavit-Hartenberg coordinate frames $\mathcal{T}_{i-1}$ and $\mathcal{T}_i$. The angle γ_i is defined as the *angular* displacement between the X axes of the Denavit-Hartenberg coordinate frame $\mathcal{T}_i$ and the S-Model coordinate frame $\mathcal{S}_i$. The parameter $\mathbf{b}_i$ is defined as the *linear* displacement between the origins of the Denavit-Hartenberg coordinate frame $\mathcal{T}_i$ and the S-Model link coordinate frame $\mathcal{S}_i$. If the displacement is in the direction of the Z axis of joint i, then γ_i is positive.

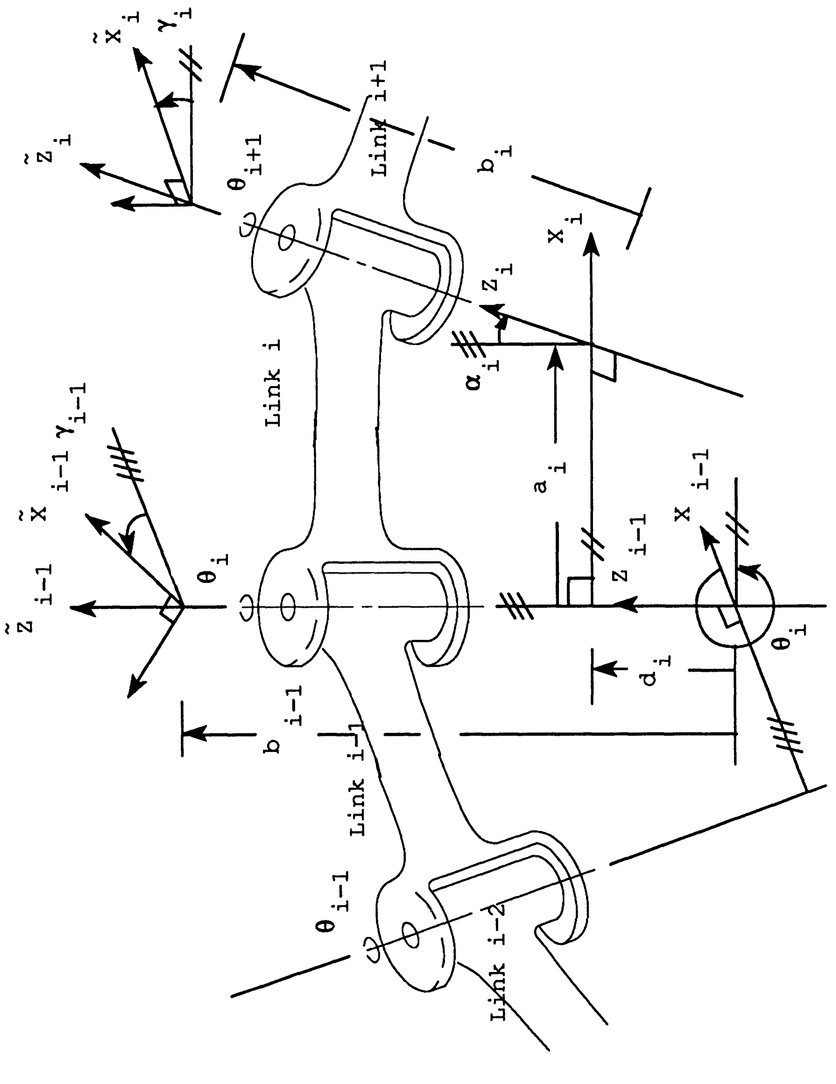


Figure 3-1: S-Model Parameters for a Revolute Joint

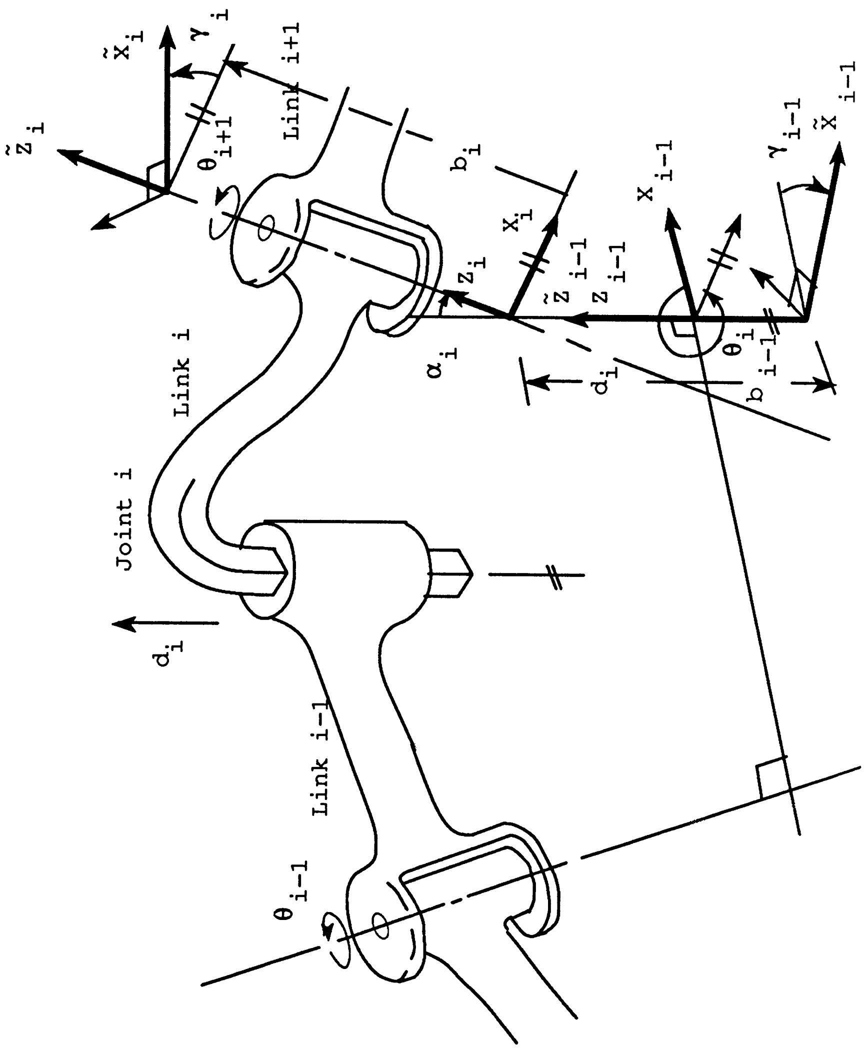


Figure 3-2: S-Model Parameters for a Prismatic Joint

The transformation matrix, $\mathbf{B}_i$, specifies the spatial transformation between the $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$ link coordinate frames for both prismatic and revolute joints. From Figure 3-1 and the definitions of γ_i, $\mathbf{b}_i$, and $\mathbf{A}_i$, the transformation matrix, $\mathbf{B}_i$, is the product

$$\begin{aligned}\mathbf{B}_i &= Rot(z,-\gamma_{i-1})Trans(0,0,-\mathbf{b}_{i-1})\,[Rot(z,\theta_i)Trans(0,0,\mathbf{d}_i)\\ &\quad Trans(\mathbf{a}_i,0,0)Rot(x,\alpha_i)]\,Rot(z,\gamma_i)Trans(0,0,\mathbf{b}_i)\ .\end{aligned} \tag{3.8}$$

The first transformation, $Rot(z,-\gamma_{i-1})$, aligns the orientation of the axes of coordinate frames $\mathcal{S}_{i-1}$ and $\mathcal{T}_{i-1}$. The second transformation, $Trans(0,0,-\mathbf{b}_{i-1})$, translates the origin of $\mathcal{S}_{i-1}$ so that it coincides with the origin of the Denavit-Hartenberg coordinate frame $\mathcal{T}_{i-1}$. The four bracketed transformations define the Denavit-Hartenberg matrix $\mathbf{A}_i$. (The parameters θ_i, $\mathbf{d}_i$, $\mathbf{a}_i$, and α_i are the Denavit-Hartenberg parameters for link *i*.) These four transformations transform coordinate frame $\mathcal{T}_{i-1}$ to the Denavit-Hartenberg coordinate frame $\mathcal{T}_i$. In analogy with the first two transformations, the cascade $Rot(z,\gamma_i)Trans(0,0,\mathbf{b}_i)$ transforms the Denavit-Hartenberg coordinate frame $\mathcal{T}_i$ to the S-Model link coordinate frame $\mathcal{S}_i$.

By combining terms according to the rules of homogeneous transformations [18], $\mathbf{B}_i$ in (3.8) simplifies to

$$\begin{aligned}\mathbf{B}_i &= Rot(z,\theta_i-\gamma_{i-1})Trans(0,0,\mathbf{d}_i-\mathbf{b}_{i-1})Trans(\mathbf{a}_i,0,0)\\ &\quad Rot(x,\alpha_i)Rot(z,-\gamma_i)Trans(0,0,\mathbf{b}_i)\ .\end{aligned} \tag{3.9}$$

Since (3.2) and (3.8) are equivalent,

$$\beta_i = \theta_i - \gamma_{i-1} \tag{3.10}$$

$$\overline{\mathbf{d}}_i = \mathbf{d}_i - \mathbf{b}_{i-1}\ , \tag{3.11}$$

where the joint rotational offset, γ_i, and the joint translational offset, $\mathbf{b}_i$, are constant parameters.

If joint i is *revolute,* β_i is a function of the joint position θ_i and the remaining five parameters, $\bar{\mathbf{d}}_i$, $\bar{\mathbf{a}}_i$, $\bar{\alpha}_i$, γ_i, and $\mathbf{b}_i$, are constants. If joint i is *prismatic,* $\bar{\mathbf{d}}_i$ is a function of the joint position $\mathbf{d}_i$ and the remaining five parameters, β_i, $\bar{\mathbf{a}}_i$, $\bar{\alpha}_i$, γ_i, and $\mathbf{b}_i$, are constants. For a manipulator with revolute joints, the four Denavit-Hartenberg parameters are extracted from the six S-Model parameters according to

$$\theta_i = \beta_i + \gamma_{i-1} \tag{3.12}$$

$$\mathbf{d}_i = \bar{\mathbf{d}}_i + \mathbf{b}_{i-1} \tag{3.13}$$

$$\mathbf{a}_i = \bar{\mathbf{a}}_i \tag{3.14}$$

$$\alpha_i = \bar{\alpha}_i \ . \tag{3.15}$$

According to the Denavit-Hartenberg model, the link transformation matrix

$$\mathbf{A}_i = \mathbf{A}_i(\mathbf{q}_i) \equiv Rot(z,\theta_i)Trans(0,0,\mathbf{d}_i)Rot(x,\alpha_i) \ , \tag{3.16}$$

for a prismatic joint and the parameter $\mathbf{a}_i$, is by definition zero. This condition is guaranteed by requiring that the axis of the prismatic joint, joint i, be chosen to intersect with the joint $i+1$ axis, as illustrated in Figure 2-3. Thus, the location of the coordinate frame $\mathcal{T}_{i-1}$ is constrained even more for a prismatic joint than for a revolute joint. This is not the case in the S-Model. From Figure 3-2, it is, in general, impossible to model the spatial transformation between the $\mathcal{S}_{i-1}$ and $\mathcal{S}_i$ link coordinate frames by the general transformation matrix in (3.2) with the parameter $\bar{\mathbf{a}}_i$ set to zero. Thus, if we were to apply the relations (3.12) - (3.15), the computed Denavit-Hartenberg parameter $\mathbf{a}_i$ would be nonzero. The parameters θ_i, $\mathbf{d}_i$, $\mathbf{a}_i$, and α_i obtained in this way for a prismatic joint are called the *modified* Denavit-Hartenberg parameters. In the modified model, the origin of the coordinate frame $\mathcal{T}_{i-1}$ is arbitrary. In Chapter 4, we present a method for determining the true Denavit-Hartenberg parameters for manipulators with prismatic joints.

3.3. Computing S-Model Parameters

In this section, we apply the backward multiplication technique [18] to derive the closed-form expressions for the S-Model parameters, β_i , $\bar{\mathbf{d}}_i$, $\bar{\mathbf{a}}_i$, $\bar{\alpha}_i$, γ_i , and $\mathbf{b}_i$, in terms of the elements of the general transformation matrix $\mathbf{B}_i$.

The transformation matrix $\mathbf{B}_i$ is given by

$$\mathbf{B}_i = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

where the individual elements are known. Premultiplying (3.2) by $Rot^{-1}(z,\beta_i)$ yields

$$\begin{aligned} Rot^{-1}(z,\beta_i)\cdot\mathbf{B}_i &= Trans(0,0,\bar{\mathbf{d}}_i)Trans(\bar{\mathbf{a}}_i,0,0)Rot(x,\bar{\alpha}_i) \\ &\quad Rot(z,\gamma_i)Trans(0,0,\mathbf{b}_i) \ , \end{aligned} \tag{3.18}$$

which when expanded becomes

$$\begin{bmatrix} \cos\beta_i & \sin\beta_i & 0 & 0 \\ -\sin\beta_i & \cos\beta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{B}_i =$$

$$\begin{bmatrix} \cos\gamma_i & -\sin\gamma_i & 0 & \mathbf{a}_i \\ \cos\alpha_i \sin\gamma_i & \cos\alpha_i \cos\gamma_i & -\sin\alpha_i & -\mathbf{b}_i \sin\alpha_i \\ \sin\alpha_i \sin\gamma_i & \sin\gamma_i & \cos\alpha_i & \mathbf{b}_i \cos\alpha_i + \mathbf{d}_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

An expression for β_i is obtained by equating the (1,3) elements in (3.19)

$$a_x \cos\beta_i + a_y \sin\beta_i = 0 \ . \tag{3.20}$$

From (3.20), we obtain the two solutions

$$\beta_i = \operatorname{atan}\frac{-a_x}{a_y} \qquad \text{and} \tag{3.21}$$

$$\beta_i = \operatorname{atan}\frac{a_x}{-a_y} \ , \tag{3.22}$$

which differ by 180 degrees. If both a_x and a_y are zero, the *i* th and *i+1* th joint axes are parallel and the parameters β_i and γ_i are redundant. When this situation occurs, we can arbitrarily set β_i to zero. Thus,

$$\beta_i = 0 \tag{3.23}$$

when $a_x = a_y = 0$. The rotational parameter $\bar{\alpha}_i$ can be expressed in terms of β_i by equating the (2,3) and (3,3) elements in (3.19). The expressions are

$$-\sin\bar{\alpha}_i = -a_x \sin\beta_i + a_y \cos\beta_i \tag{3.24}$$

$$\cos\bar{\alpha}_i = a_z \ . \tag{3.25}$$

Having computed β_i using either (3.21) or (3.22), the unique solution for $\bar{\alpha}_i$ obtained from (3.24) and (3.25), is

$$\bar{\alpha}_i = \operatorname{atan}\frac{a_x \sin\beta_i - a_y \cos\beta_i}{a_z} . \tag{3.26}$$

Two expressions involving γ_i and β_i can also be obtained from (3.19). Equating the (1,1) and (1,2) elements in (3.19) yields

$$\cos\gamma_i = n_x \cos\beta_i + n_y \sin\beta_i , \tag{3.27}$$

$$-\sin\gamma_i = o_x \cos\beta_i + o_y \sin\beta_i . \tag{3.28}$$

Using (3.27), (3.28), and β_i, the unique solution for γ_i is

$$\gamma_i = \operatorname{atan}\frac{-o_x \cos\beta_i - o_y \sin\beta_i}{n_x \cos\beta_i + n_y \sin\beta_i} . \tag{3.29}$$

The solution for $\bar{\alpha}_i$ is obtained by equating the (1,4) elements in (3.19). Thus,

$$\bar{\mathbf{a}}_i = p_x \cos\beta_i + p_y \sin\beta_i . \tag{3.30}$$

Equating the (2,4) elements in (3.19) yields

$$-p_x \sin\beta_i + p_y \cos\beta_i = -\mathbf{b}_i \sin\alpha_i \tag{3.31}$$

from which we obtain the solution

$$\mathbf{b}_i = \frac{-p_x \sin\beta_i + p_y \cos\beta_i}{\sin\alpha_i} \qquad \text{if } \sin\alpha_i \neq 0$$

$$\mathbf{b}_i = 0 \qquad \text{if } \sin\alpha_i = 0 . \tag{3.32}$$

Finally, equating the (3,4) elements in (3.19) and rearranging, yields the solution for $\bar{\mathbf{d}}_i$, namely

$$\bar{\mathbf{d}}_i = p_z - \mathbf{b}_i \cos\alpha_i . \tag{3.33}$$

In chapter 4, we apply the solutions (3.21), (3.26), (3.29), (3.30), (3.32), and (3.33) in formulating the S-Model identification algorithm.

3.4. Conclusions

The S-Model described in this chapter offers several advantages for the development of a kinematic identification algorithm:

- The flexibility in assigning link coordinate frames leads to a simple, efficient, and accurate algorithm for identifying the location of the S-Model link coordinate frames S_i for $i=0, \ldots, n-1$
- The Denavit-Hartenberg model parameters may be extracted from the S-Model parameters according to (3.12) - (3.15)

The development of a kinematic identification algorithm using these principles is described in the next chapter.

Chapter 4

Kinematic Identification

4.1. Overview

The goal of a kinematic identification algorithm is to identify the parameters of a kinematic model which describes the actual position and orientation of the end-effector in terms of the measured joint positions, and which incorporates the geometrical variations in the structure caused by manufacturing errors. Either the Denavit-Hartenberg model or the S-Model are adequate to provide an *exact* description of the actual robot kinematics. Identification of these parametric models, however, requires detailed consideration of the structure of the models as well as an adequate procedure to measure robot configurations. Because of manufacturing errors, *all* exact kinematic models will possess *non*-simple structures leading to more complex control algorithm design and implementation tasks.

While the Denavit-Hartenberg model is specified by a minimum number of parameters, it possesses a rigid structure and is not amenable to direct identification. The term "*rigid*" signifies that all the parameters and components to the model are precisely defined and are unique. In contrast, the S-Model is directly applicable to kinematic identification. The relationships in (3.12) - (3.15) provide a mechanism for calculating the Denavit-Hartenberg parameters from the identified S-Model parameters.

In Section 4.2, we describe the intrinsic properties of mechanical joints, referred to as the *kinematic features*. Then, in Section 4.3, we develop the S-Model identification algorithm.

4.2. Kinematic Features

The objective of *S-Model Identification* is to estimate the S-Model kinematic parameters from a set of $2n_r + n_p$ mechanical features inherent to the manipulator, where n_r is the number of revolute joints and n_p is the number of prismatic joints. (The number of degrees-of-freedom $n = n_r + n_p$). The two features of a revolute joint are called the *center-of-rotation* and the *plane-of-rotation,* and the feature of a prismatic joint is called the *line-of-translation*.

These features are derived from basic geometric considerations of the joints. The locus of a point rotating about an axis is a circle lying in a plane, called the *plane-of-rotation* and the normal to this plane is a vector which is parallel to the axis of rotation. The center of the circle is a point, called the *center-of-rotation* which lies on the axis of rotation. When joint *i-1* of a manipulator is rotated, any point which is fixed relative to the *i* th link defines a plane-of-rotation and a center-of-rotation, under the assumption that the positions of joints *1* through *i-2* remain fixed. We associate this plane-of-rotation and center-of-rotation with the (*i-1*) th joint and the *i* th link.

The line-of-translation is a feature of a *prismatic* joint. When a point is displaced linearly, its trajectory is a straight line which is parallel to the vector which indicates the direction of the displacement. For a manipulator, any point which is fixed relative to link *i* defines a line-of-translation when joint *i-1* is moved, under the assumption that the positions of joints *1* through *i-2* remain fixed.

4.3. S-Model Identification

4.3.1. Overview

The approaches to solving the manipulator kinematic parameter identification problem discussed in Chapter 2 all propose to identify the kinematic parameters *directly* and *explicitly* from a set of observed measurements, typically the position and/or orientation of the end-effector. Since the kinematics of any manipulator with at least one revolute joint will be nonlinear, such a direct method inevitably leads to a nonlinear minimization problem. In contrast, the S-Model identification algorithm is an *indirect method* of identification which leads naturally to a separation of the identification problem into a set of independent, less complex minimization problems.

In this section, we describe our solution to the kinematic parameter identification problem. The detailed formulation and implementation are presented in subsequent sections. The S-Model identification algorithm includes four steps:

1. Feature identification
2. Link coordinate frame specification
3. S-Model parameter computation
4. Denavit-Hartenberg parameter extraction

4.3.1.1. Feature Identification

In the first step, *feature identification* the kinematic features are identified from measurements of the Cartesian position of targets physically attached to the robots links. The positions of these targets, relative to an independent fixed coordinate frame, vary as the manipulator changes its configuration. During data collection the manipulator is programmed to move through a sequence of joint configurations. At each configuration the position of a target is measured. The kinematic features are described analytically by an algebraic equation, and the coefficients of these equations are the feature parameters which are estimated.

4.3.1.2. Link Coordinate Frame Specification

The second step in our identification algorithm is *link coordinate frame specification*. If the manipulator configuration is known in terms of the joint positions a set of link coordinate frames which satisfy the S-Model convention (in Section 3.2) may be defined. By collecting target measurements and identifying the parameters of the kinematic features which correspond to this same manipulator configuration, we can readily establish a valid set of S-Model link coordinate frames. Then, we can apply the estimated feature parameters to construct and evaluate the elements of the matrices $\mathbf{S}_i$ in (3.1).

4.3.1.3. S-Model Parameter Computation

The third step, *S-Model parameter computation* applies the application of the inverse kinematic parameter relationships developed in Section 3.3. The elements of the general transformation matrices $\mathbf{B}_i$, which are the arguments to these inverse relationships, are a function of the transformation matrices $\mathbf{S}_i$ determined in the previous step. Identifying the S-Model parameters requires straightforward numerical evaluation.

4.3.1.4. Denavit-Hartenberg Parameter Extraction

In the fourth and final step, *Denavit-Hartenberg parameter extraction* the relationships developed in Section 3.2 are applied to determine the manipulators Denavit-Hartenberg kinematic parameters from the identified S-Model parameters.

The procedure for varying the configuration of the manipulator in order to obtain the necessary measurements of the target positions and the analytic techniques for solving for the estimated feature parameters are presented in Section 4.3.2. In Section 4.3.3, we introduce the rules for establishing valid S-Model link coordinate frames and constructing the matrices $\mathbf{S}_i$. Evaluation of the general transformation matrices and the S-Model parameters are described in Section 4.3.4 with special attention given to the determination of the first and last link parameters. Finally, in Section 4.3.5, we review the procedure for extracting the Denavit-Hartenberg parameters and introduce the notion of a pseudo Denavit-Hartenberg model to account for physical alignment constraints.

4.3.2. Feature Identification

Identifying a plane-of-rotation and a center-of-rotation is conceptually straightforward. Imagine an *arbitrary* target point fixed relative to link i+1. When joint i is rotated, this target point traces a circle in space. The plane in which the circle lies is, by definition, a *plane-of-rotation*. The coefficients of the equation of this plane can be estimated from a curve fit of m measured Cartesian positions of the target along the circle corresponding to m different positions of joint i. (It is assumed that all Cartesian measurements are made with respect to the sensor frame). The center of the traced circle, defined by its three-dimensional Cartesian coordinates, is the *center-of-rotation*. We can extract the coordinates of the center-of-rotation from an estimate of the parameters of the equation for the traced circle. Identifying a line-of-translation is also straightforward. When joint i is translated, the target point traces a line in space. The parameters of the line, which constitute the *line-of-translation*, can be estimated by fitting the measured target locations to the equation of a line.

In practice, the target point may be a physical location on the $i+1$ st link or a location of a point on a body which is attached rigidly to link $i+1$. We will refer to the target point attached to link $i+1$ as the i th target point or as target point i. The flexibility in choosing the location of the target point is another important advantage of our approach. For a revolute joint, the nominal radius of the target point to the joint axis can be controlled to accommodate the sensor system constraints and to enhance feature estimate accuracy. We will return to the issue of enhancing estimate accuracy once we have formulated and presented solutions to the feature parameter estimation problem.

Identifying a plane-of-rotation and a center-of-rotation from a set of points in three dimensions corresponds to the problem of fitting such points to a circle. In this problem, the parameters to be identified are the coordinates of the center of the circle, the circle's radius, the unit normal vector to the plane in which the circle lies, and the coordinates of n points lying on the circle corresponding to each of the n measured points. Given the measured positions of the target points, $\vec{p}_1 , \vec{p}_2 , \ldots , \vec{p}_n$ we wish to minimize the function

$$\sum_{i=1}^{n} \| \vec{x}_i - \vec{p}_i \|^2 \tag{4.1}$$

subject to the constraints

$$\|\vec{x}_i - \vec{p}\|^2 - r^2 = 0 \ , \tag{4.2}$$

$$(\vec{x}_i - \vec{p})^T \vec{a} = 0 \ , \tag{4.3}$$

$$\|\vec{a}\| - 1 = 0 \ , \tag{4.4}$$

and minimized with respect to the $3{\cdot}n + 7$ scalar parameters contained within $\vec{p}, \vec{a}, r, \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n$. The vector $\vec{p}$ defines the Cartesian position of the center of the circle, the vector $\vec{a}$ defines the unit vector normal to the plane in which the circle lies, r is the radius of the circle, and the vectors $\vec{x}_i$ for $i = 1, 2, \ldots, n$ define the Cartesian position of n points on the circle. This constrained minimization problem can be transformed into an equivalent unconstrained minimization problem using LaGrange's technique. The solution is then given by the solution to a set of $5{\cdot}n + 8$ simultaneous nonlinear algebraic equations in the same number of unknowns. The additional $2{\cdot}n + 1$ unknowns are LaGrange multipliers. In practice, the minimization of (4.1) becomes an extremely difficult problem to solve when the number of measurements, as in our implementation, is on the order of 50 to 100. In Sections 4.3.2.1 and 4.3.2.2, the problem of identifying the plane-of-rotation and center-of-rotation is separated into two independent problems. The combination of the two solutions is both realistically and intuitively pleasing.

4.3.2.1. Plane-of-Rotation Estimation

We present two methods for fitting the measured target positions of a revolute joint to a plane. Our objective is to minimize the sum of the perpendicular distances between the measurements and the estimated plane. The solution to this minimization problem yields the coefficients which define an estimated plane-of-rotation. The first method [1], is based on an error measure divided by its average gradient and uses an eigenvalue solution. The second method approximates the solution of the desired minimization problem by the repeated application of a linear least-squares regression. In formulating the latter, we rely heavily upon our a priori knowledge of and control over the generation of the target positions. We have incorporated both algorithms into our S-Model identification algorithm. In Chapter 6 we demonstrate experimentally that for measurements obtained using an ultrasonic sensor system, the performance of these two algorithms is identical (i.e., the same feature parameters are obtained). For sensor systems with different noise properties and/or physical constraints upon the measurable target locations, the two algorithms may yield differences in performance. The linear least-squares solution is easier to implement in software. The eigenvalue solution is analytically exact.

While measuring the position of the i th target, joints *1* through *i-1* are required to be in their corresponding signature configuration positions. Joints *i+1* to *n*, on the other hand, can be positioned arbitrarily. Through independent control of the manipulators joints, joint i is then sequentially indexed to m different positions, $\mathbf{q}_{i,j}$ for $j = 1, \ldots, m$. These positions should be uniformly spaced about the physical range of motion of joint i. To later maintain a correct sense of rotation, it is assumed that the positions $\mathbf{q}_{i,j}$ are ordered such that $\mathbf{q}_{i,j} < \mathbf{q}_{i,j+1}$, as illustrated in Figure 4-1. Thus, the interval between successive joint positions

$$\Delta \mathbf{q}_i = \frac{\mathbf{q}_i^{max} - \mathbf{q}_i^{min}}{m-1} , \tag{4.5}$$

where $\mathbf{q}_i^{min}$ and $\mathbf{q}_i^{max}$ are the corresponding minimum and maximum limits for joint i, respectively. At each successive configuration, the Cartesian position of the i th target, denoted by the column vector $\vec{\mathbf{p}}_j = [x_j, y_j, z_j]^T$, is measured with respect to the sensor frame. The correspondence between the manipulator configurations and the target position measurements is listed in Table 4-1.

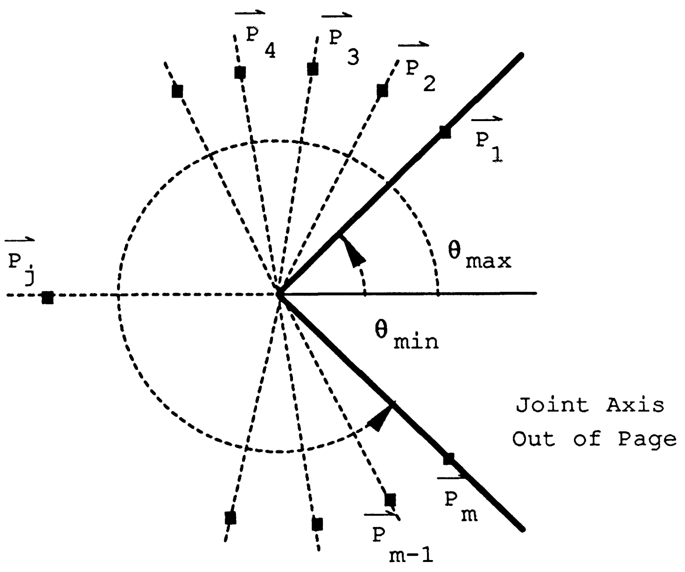


Figure 4-1: Indexing Joint *i* to Generate a Plane-of-Rotation and a Center-of-Rotation

The normal form for the equation of plane is

$$\mathbf{A}x + \mathbf{B}y + \mathbf{C}z + \mathbf{D} = 0 \ , \tag{4.6}$$

where the coefficients **A**, **B**, **C**, and **D** are the parameters to be identified. We chose as the measure of error between the measured target position and the plane the function

$$\xi(x,y,z) = \mathbf{A}x + \mathbf{B}y + \mathbf{C}z + \mathbf{D}$$
$$\xi(x,y,z) = \phi^T\Theta = \Theta^T\phi \ , \tag{4.7}$$

Table 4-1: Target Point Correspondence

Manipulator Configuration	Measured Target Position
$\vec{q}_1 = [q_1^s, q_2^s, \cdots, q_{i-1}^s, q_{i,1}, q_{i+1}^a, \cdots, q_n^a]^T$	$\vec{p}_1 = [x_1 \ y_1 \ z_1]^T$
$\vec{q}_2 = [q_1^s, q_2^s, \cdots, q_{i-1}^s, q_{i,2}, q_{i+1}^a, \cdots, q_n^a]^T$	$\vec{p}_2 = [x_2 \ y_2 \ z_2]^T$
•	•
•	•
•	•
$\vec{q}_j = [q_1^s, q_2^s, \cdots, q_{i-1}^s, q_{i,j}, q_{i+1}^a, \cdots, q_n^a]^T$	$\vec{p}_j = [x_j \ y_j \ z_j]^T$
•	•
•	•
•	•
$\vec{q}_m = [q_1^s, q_2^s, \cdots, q_{i-1}^s, q_{i,m}, q_{i+1}^a, \cdots, q_n^a]^T$	$\vec{p}_n = [x_n \ y_n \ z_n]^T$

where

$$\phi = [x \ y \ z \ 1]^T \ , \tag{4.8}$$

and

$$\Theta = [\mathrm{A} \ \mathrm{B} \ \mathrm{C} \ \mathrm{D}]^T \tag{4.9}$$

are the augmented information vector and parameter vector, respectively. At a point on the plane, the error function $\xi(x, y, z)$ is zero. It is also observed that the magnitude of the error function increases at points farther away from the plane. For points close to the plane, the error function is proportional to the perpendicular distance from the point to the plane. The constant of proportionality is the reciprocal of the magnitude of the gradient of $\xi(x, y, z)$. The magnitude of the error is

$$\xi^2(x,y,z) = \Theta^T \phi \phi^T \Theta \ . \tag{4.10}$$

Using the m measured target positions, $\vec{p}_{i,j}$, the aggregate error is

$$\Xi \equiv \sum_{j=1}^{m} \xi_j^2 = \sum_{j=1}^{m} (\Theta^T \phi_j \phi_j^T \Theta) = \Theta^T \Phi \Theta \ , \tag{4.11}$$

where

$$\Phi = \sum_{j=1}^{m} (\phi_j \phi_j^T) \ . \tag{4.12}$$

Minimizing the sum of the squares of the perpendicular distances between the measured target points and the plane is equivalent to minimizing Ξ under the constraint that the average magnitude of the gradient of the error function at the measured target points, $\vec{p}_{i,j}$, for $j=1, \ldots ,m$, is unity. The squared magnitude of the gradient of the error is

$$(\nabla\xi)^2 \equiv (\frac{\partial\xi}{\partial x})^2 + (\frac{\partial\xi}{\partial y})^2 + (\frac{\partial\xi}{\partial z})^2 = \Theta^T \Psi \Theta \tag{4.13}$$

where Ψ is the (4x4) diagonal matrix Diag [1 1 1 0]. Since Ψ is constant, the constraint requiring that the mean-squared gradient of ξ equal unity is equivalent to setting (4.13) equal to one,

$$\Theta^T \Psi \Theta = 1 \ . \tag{4.14}$$

We thus seek the parameter vector Θ that minimizes (4.11) subject to the constraint (4.14). Using LaGrange multipliers, the solution to this constrained minimization is given by the solution to the generalized eigenvalue problem [24]

$$\Phi\Theta = \lambda \Psi \Theta \ , \tag{4.15}$$

where λ is a scalar. Using (4.11) and (4.15), the aggregate error $\Xi = \lambda m$.

Therefore, the desired set of parameters are given by the eigenvector corresponding to the smallest eigenvalue. The singularity of Ψ precludes the direct application of typical eigenvalue methods to solve (4.15). To solve (4.15), we first partition it into the form

$$\left[\begin{array}{ccc|c} \Sigma x_j^2 & \Sigma x_jy_j & \Sigma x_jz_j & \Sigma x_j \\ \Sigma y_jx_j & \Sigma y_j^2 & \Sigma y_jz_j & \Sigma y_j \\ \Sigma z_jx_j & \Sigma z_jy_j & \Sigma z_j^2 & \Sigma z_j \\ \hline \Sigma x_j & \Sigma y_j & \Sigma z_j & m \end{array}\right] \left[\begin{array}{c} A \\ B \\ C \\ \hline D \end{array}\right] = \lambda \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 \end{array}\right] \left[\begin{array}{c} A \\ B \\ C \\ \hline D \end{array}\right] \tag{4.16}$$

where summations from $j = 1$ to m are indicated by the symbol Σ. The scalar equation defined by the bottom row in (4.16) yields the solution for the parameter D, namely,

$$\begin{aligned} D &= -\frac{A\sum_{j=1}^{m} x_j + B\sum_{j=1}^{m} y_j + C\sum_{j=1}^{m} z_j}{m} \\ &= -\left(A\bar{x} + B\bar{y} + C\bar{z}\right) , \end{aligned} \tag{4.17}$$

where $\bar{x}$, $\bar{y}$, and $\bar{z}$ are the sample means of the x, y, and z coordinates, respectively. Substituting solution (4.17) into the upper 3x3 matrix equation in (4.16) yields the well defined standard eigenvalue problem

$$\begin{bmatrix} \Sigma\, x_j x_j - m\,\overline{x}^2 & \Sigma\, x_j y_j - m\,\overline{y}\,\overline{x} & \Sigma\, x_j z_j - m\,\overline{z}\,\overline{x} \\ \Sigma\, y_j x_j - m\,\overline{x}\,\overline{y} & \Sigma\, y_j y_j - m\,\overline{y}^2 & \Sigma\, y_j z_j - m\,\overline{z}\,\overline{y} \\ \Sigma\, z_j x_j - m\,\overline{x}\,\overline{z} & \Sigma\, z_j y_j - m\,\overline{y}\,\overline{z} & \Sigma\, z_j z_j - m\,\overline{z}^2 \end{bmatrix} \begin{bmatrix} A \\ B \\ C \end{bmatrix} = \lambda \begin{bmatrix} A \\ B \\ C \end{bmatrix} \qquad (4.18)$$

which can be solved using any number of methods, such as the Q-R algorithm [10]. The estimated plane-of-rotation defined by the parameter vector $\boldsymbol{\Theta}$ is given by the eigenvector corresponding to the smallest eigenvalue λ_k which comprises a solution to (4.18) and **D** obtained from (4.17).

Algorithms for solving eigenvalue problems are relatively complex and may be costly to implement in software. As an alternative method of solving the desired minimization problem, we have developed an algorithm which uses repeated application of a linear least-squares regression. Formally, this algorithm provides an approximate solution rather than the exact solution to (4.18) and (4.17). Our experiments (in Chapter 6), however, have demonstrated that for our system the two solutions are essentially identical. The formulation and performance of this algorithm is based upon the following assumptions:

- The standard deviation in the measurements of the target's Cartesian position is several orders of magnitude (i.e., typically > 3) less than the nominal distance between the target and the axis-of-rotation (i.e., the nominal target radius) (B-1).
- The measured target positions correspond to joint positions uniformly distributed between the upper and lower limits of the joints travel (B-2).
- The manipulators revolute joints have at least 180 degrees of travel (B-3).

Ensuring that assumptions (B-1) and (B-2) are satisfied is a simple matter and nearly all industrial robotic manipulators satisfy assumption (B-3) in order to have sufficient dexterity and reach.

The general equation for a plane (4.6) can be rewritten as

$$z = \mathrm{E}x + \mathrm{F}y + \mathrm{G} = \phi^T \Theta \ , \tag{4.19}$$

where

$$\phi = [x \ y \ 1]^T \ , \tag{4.20}$$

and

$$\Theta = [\mathrm{E} \ \ \mathrm{F} \ \ \mathrm{G}]^T \tag{4.21}$$

are defined to be the information vector and the parameter vector, respectively. The Z coordinate is defined to be the output of (4.19). A simple regression of z on x, y, and 1 corresponds to a minimization of the the sum of the squared errors in the z coordinate, namely

$$\Xi \equiv \sum_{j=1}^{m} \xi_j = \sum_{j=1}^{m} (z-z_j)^2 = \sum_{j=1}^{m} (\phi^T \Theta - z_j)^2 \ . \tag{4.22}$$

The well known closed form solution [24] for the minimization of (4.22) is

$$\Theta = [\Phi^T \Phi]^{-1} \Phi \mathrm{Z} \ , \tag{4.23}$$

where

$$\Phi = [\phi_1 \ \phi_2 \ , \dots , \ \phi_m]^T \ , \tag{4.24}$$

$$\mathrm{Z} = [z_1 \ z_2 \ , \dots , \ z_m]^T \ . \tag{4.25}$$

The (3x3) correlation matrix $\Phi^T \Phi$ is composed of sums of products of x, y, and 1. The application of (4.22) assumes that the X and Y coordinates are independent variables and measured without error. Consequently, the closer the plane-of-rotation is to being parallel with the Z axis the farther the solution (4.23) is from the true plane-of-rotation. We avoid this problem by transforming the data to a new coordinate frame in which the measured target positions lie essentially in the X-Y plane. Assumption (B-1) insures that the measured data closely fits a plane. Then, by applying the linear least-squares solution (4.23), we are effectively minimizing the sum of the perpendicular errors to the plane-of-rotation. The required transformation can be computed using three of the *m* measured target positions. These three positions uniquely define an initial approximation to the plane-of-rotation. The initial approximation is formed using the three measured positions which are mutually most distant from one another. We denote them by $\vec{p}_k$, $\vec{p}_l$, and $\vec{p}_m$. Further, we require that the corresponding joint positions satisfy $q_{i,k} < q_{i,l} < q_{i,m}$ in order to preserve the sense of rotation. Figure 4-2 illustrates the coordinate frame C_0 formed by these points.

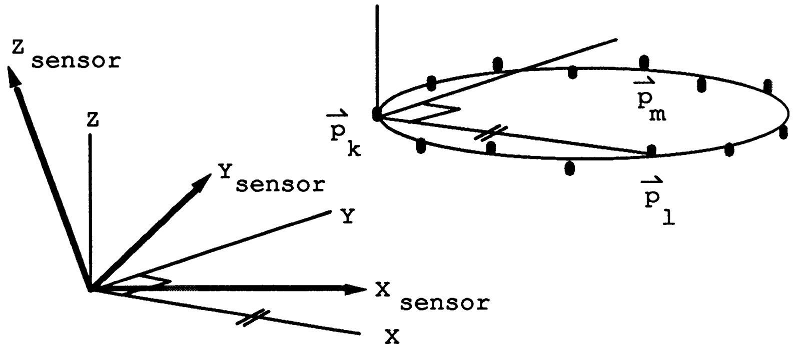


Figure 4-2: Initial Approximation to the Plane-of-Rotation Using Three Mutually Distant Target Measurements

The X axis is parallel to the line joining $\vec{p}_k$ and $\vec{p}_l$; the Z axis is perpendicular to this line and the line joining $\vec{p}_k$; and $\vec{p}_m$, and the Y axis completes the orthogonal system. The origin of C_0 is coincident with the origin of the sensor frame. Thus, the homogeneous transformation matrix defining the transformation between the sensor frame and C_0 is

$$\mathbf{R} = \begin{bmatrix} \vec{n}_i & \vec{o}_i & \vec{a}_i & \vec{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.26}$$

where

$$\vec{n} = \frac{\vec{p}_l - \vec{p}_k}{|\vec{p}_l - \vec{p}_k|} , \tag{4.27}$$

$$\vec{a} = \frac{(\vec{p}_l - \vec{p}_k) x (\vec{p}_m - \vec{p}_k)}{|(\vec{p}_l - \vec{p}_k) x (\vec{p}_m - \vec{p}_k)|} , \tag{4.28}$$

and

$$\vec{o} = \vec{a} x \vec{n} . \tag{4.29}$$

The transformed measurements are computed according to

$$\vec{p}_j^{\,0} = \mathbf{R}\vec{p}_j \qquad \text{for } j=1, \dots ,m \tag{4.30}$$

and are then used in (4.23) to obtain the first estimate, $\Theta = \Theta_1^0$, for the plane-of-rotation with respect to coordinate frame C_0[5]. The standard coefficients A^0, B^0, C^0, and D^0 can then be computed. The coefficients are transformed back into the sensor frame via $[\mathrm{A\ B\ C\ D}]^T = \mathbf{R}^T [\mathrm{A}^0\ \mathrm{B}^0\ \mathrm{C}^0\ \mathrm{D}^0]^T$, since $\mathbf{R}$ represents a pure rotation. The n_r unit normal vectors to the estimated plane-of-rotations which will be used in the next section are

[5]The superscript "0" indicates that the associated parameters are measured with respect to coordinate frame C_0.

$$[Aw \ \ Bw \ \ Cw]^T \tag{4.31}$$

where

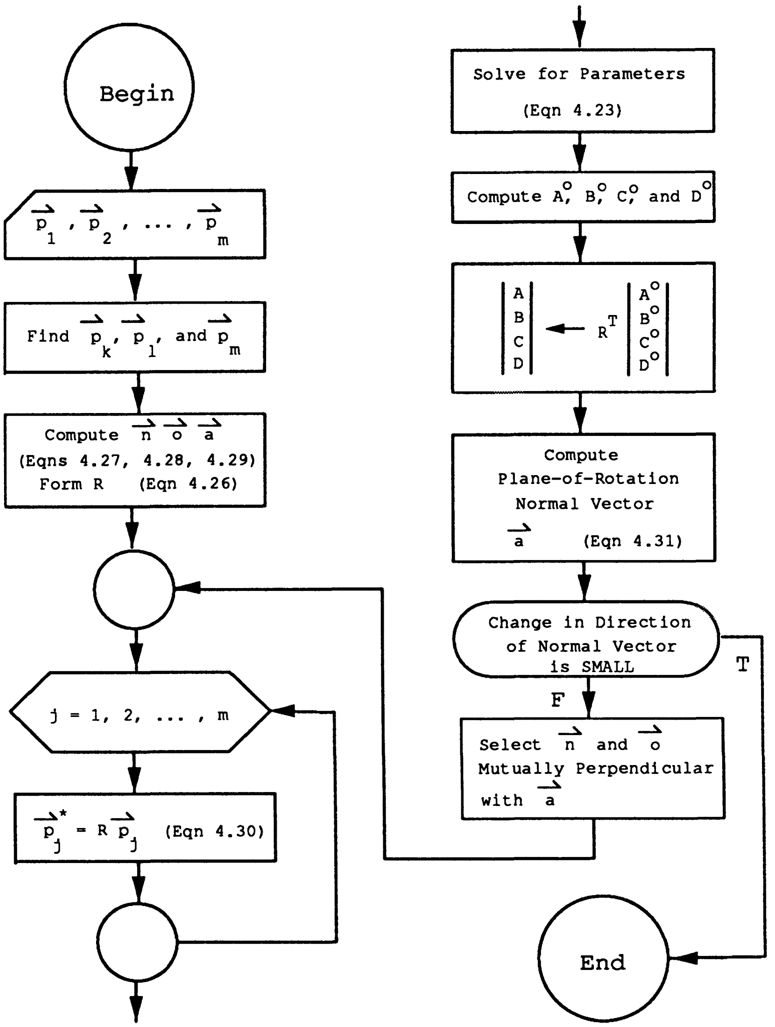
$$w = \sqrt{A^2+B^2+C^2} \ . \tag{4.32}$$

In general, since the estimated plane-of-rotation will not be exactly parallel to the X-Y plane of C_0 , the desired minimization problem may not be adequately solved. To improve upon the estimate Θ_1 we repeatedly transform the measured target positions and apply the least-squares solution (4.23). At the i th iteration, the transformation matrix $\mathbf{R}=\mathbf{R}_i$ is computed using the estimated normal to the plane-of-rotation (4.31) obtained from the previous iteration. In (4.26), the Z axis unit direction vector $\vec{a}$ is then given by the i th normal estimate (4.31) while the X and Y axis unit direction vectors $\vec{n}$ and $\vec{o}$ are chosen arbitrarily. When the difference between consecutive estimates of the plane-of-rotation becomes negligible, the algorithm terminates. Figure 4-3 is a flowchart of the repeated linear least-squares estimation algorithm. In practice, when assumptions (B-1), (B-2), and (B-3) are satisfied, only a few iterations e.g., 2 - 4, are required.

4.3.2.2. Center-of-Rotation Estimation

The center-of-rotation feature physically corresponds to the position of an arbitrary point which lies on the joint axis of rotation. In this section, we present an algorithm for estimating the Cartesian position of one such point. We then apply the estimate, in Section 4.3.3, to formulate a kinematic model.

Ideally, the target positions $\vec{p}_{i,j}$ (in Section 4.3.2.1), in addition to lying on a plane, should also lie on a circle in space. The point defined by the center of this circle naturally satisfies the criteria for a center-of-rotation feature even though it may be impossible to determine to which physical point on the joint axis and/or on the manipulator this point corresponds. An



Begin
$\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_m$
Find $\vec{p}_k$, $\vec{p}_l$, and $\vec{p}_m$
Compute $\vec{n}$ $\vec{o}$ $\vec{a}$
(Eqns 4.27, 4.28, 4.29)
Form R (Eqn 4.26)
j = 1, 2, ... , m
$\vec{p}_j^{\,*} = R\,\vec{p}_j$ (Eqn 4.30)
Solve for Parameters
(Eqn 4.23)
Compute A^o, B^o, C^o and D^o
$\begin{vmatrix} A \\ B \\ C \\ D \end{vmatrix} \leftarrow R^T \begin{vmatrix} A^o \\ B^o \\ C^o \\ D^o \end{vmatrix}$
Compute Plane-of-Rotation Normal Vector
$\vec{a}$ (Eqn 4.31)
Change in Direction of Normal Vector is SMALL
T
F
Select $\vec{n}$ and $\vec{o}$ Mutually Perpendicular with $\vec{a}$
End

Figure 4-3: Flow Chart of the Repeated Linear Least-Squares Algorithm for Estimating the Plane-of-Rotation

estimate for the Cartesian coordinates of the center-of-rotation, with respect to the sensor frame, are obtained by fitting the measured target positions $\vec{p}_{i,j}$, for $j = 1, \ldots, m$, to the equation of a circle. Our objective in this estimation problem is to minimize the sum of the perpendicular distances between the measured and the estimated circle. Unfortunately, a method analogous to the average gradient constraint algorithm used in Section 4.3.2.1 cannot be applied to solve this minimization problem. Therefore, we take the same approach as in Section 4.3.2.1 to develop an algorithm based upon repeated application of a linear least-squares regression. In other words, it is assumed that (B-1), (B-2), and (B-3) are satisfied.

The standard form for the equation of circle is

$$(x - \mathbf{g})^2 + (y - \mathbf{h})^2 = \mathbf{r}^2 , \tag{4.33}$$

where **g** and **h** are the X and Y coordinates of the center, respectively, and **r** is the radius. While a circle can lie in three dimensions, it only spans two dimensions. In the form (4.33), the Z coordinate of the center is thus implicitly zero. To apply (4.33) or a variation thereof, we must project the measured target positions which will, in general, span three dimensions to an appropriate two-dimensional subspace. From our previous discussions in Section 4.2, the required subspace is defined by the corresponding estimated plane-of-rotation. Furthermore, since (4.33) is represented to lie in the X-Y plane, a transformation of coordinates is also necessary. If the appropriate coordinate frame transformation is applied to the data first, the subsequent projection onto the subspace becomes trivial.

The new coordinate frame is denoted by $\mathcal{C}$. The transformation between the sensor frame and $\mathcal{C}$ is defined by the homogeneous transformation matrix

$$\mathbf{R} = \begin{bmatrix} \vec{n}_i & \vec{o}_i & \vec{a}_i & \vec{0} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.34}$$

where the vector $\vec{a}$ is the unit normal vector to the estimated plane of rotation (4.31) and the parameters **A**, **B**, **C**, and **D** are obtained using either of the methods in Section 4.3.2.1. The unit direction vectors $\vec{n}$ and $\vec{o}$ can be chosen arbitrarily so long as $\vec{n}$, $\vec{o}$, and $\vec{a}$ form an orthogonal right-handed coordinate system. The target positions are transformed according to

$$\vec{p}_j^{\,\times} = \mathbf{R}\vec{p}_j \qquad \text{for } j=1,\ldots,m. \tag{4.35}$$

The Z coordinate of the center-of-rotation in frame C is

$$z_c^{\times} = \frac{\mathbf{D}}{(\mathbf{A}^2+\mathbf{B}^2+\mathbf{C}^2)^{1/2}} . \tag{4.36}$$

Since the X-Y plane of C is parallel to the estimated plane-of-rotation, the X and Y coordinates of $\vec{p}_j^{\,\times}$ constitute the projection of the target positions.

The equation of a circle (4.33) can be rewritten as

$$w \equiv x^2 + y^2 = \mathbf{A}x + \mathbf{B}y + \mathbf{C} = \phi^T\Theta , \tag{4.37}$$

where

$$\phi = [x \ y \ 1]^T \tag{4.38}$$

and

$$\Theta = [\mathbf{A} \ \mathbf{B} \ \mathbf{C}]^T \tag{4.39}$$

are the information vector and the parameter vector, respectively. The output of (4.37) is the squared distance between a point on the circle and the origin. A simple regression of w on x, y, and 1 corresponds to the minimization of

$$\Xi \equiv \sum_{j=1}^{m} \xi_j = \sum_{j=1}^{m} (w-w_j)^2 = \sum_{j=1}^{m} (\phi^T\Theta - w_j)^2 \ , \tag{4.40}$$

the sum of the squared errors in the output, w. The solution is identical to (4.23) with **Z** replaced by the column vector **W** containing the elements $w_j = x_j^2 + y_j^2$. In general, minimizing (4.40) is not the same as minimizing the sum of the squared perpendicular distances between the measurements and the circle, unless the origin of the coordinate frame and the circle are coincident. Following the approach taken in the previous section, we repeatedly apply the linear solution (4.23) and a transformation of coordinates. At the *i* th iteration, we translate the X and Y components of the originally projected measurements $\vec{p}_j^{\times}$ by

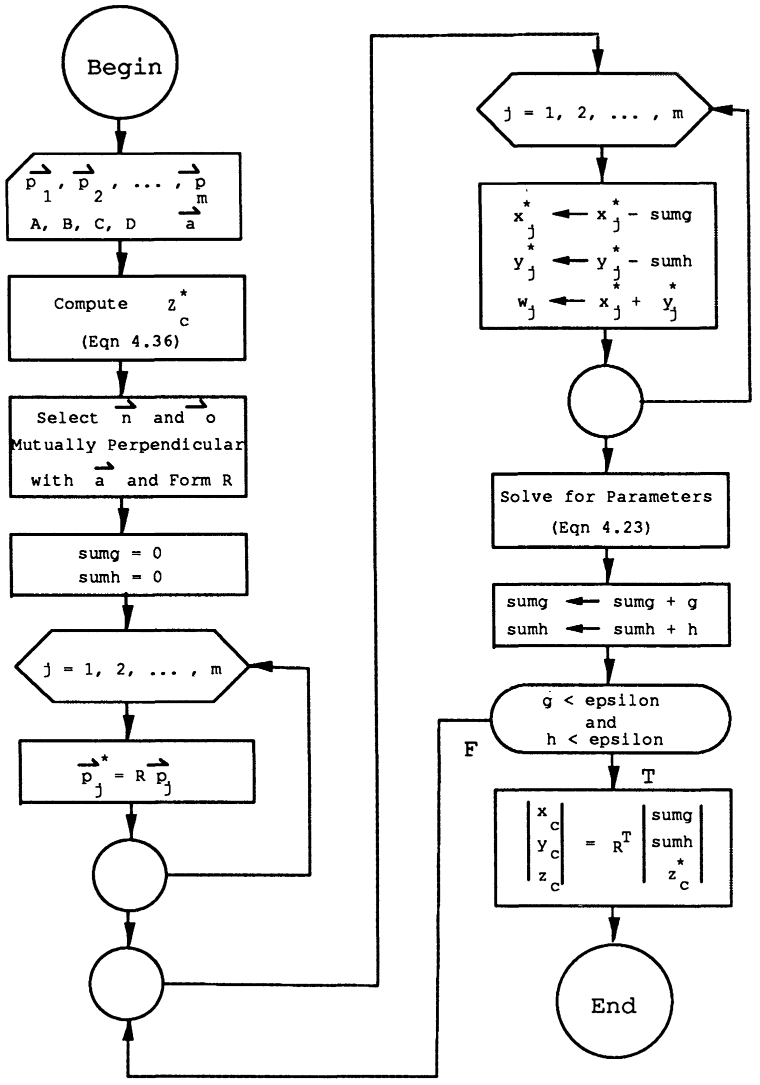
$$x_j^i = x_j^{\times} - \mathbf{g}_{i-1} \tag{4.41}$$

$$y_j^i = y_j^{\times} - \mathbf{h}_{i-1} \ , \tag{4.42}$$

where $\mathbf{g}_{i-1}$ and $\mathbf{h}_{i-1}$ are the estimated coordinates of the center-of-rotation computed during the *i-1* th iteration. By repeatedly translating the original data during each iteration, the origin of the circle in the translated frame approaches zero (i.e., approaches the origin of the frame). The solution (4.23) using x_j^i and y_j^i thus approaches the solution to the desired minimization problem. Figure 4-4 is a flow chart of the repeated linear least-squares algorithm for estimating the center-of-rotation. The vector defining the center-of-rotation for joint *i* in coordinate frame C is $[\mathbf{g}\ \ \mathbf{h}\ \ \mathbf{z}_c]^T$. The center-of-rotation in sensor coordinates is

$$\vec{p}_i^{\,c} = \mathbf{R}[\mathbf{g}\ \ \mathbf{h}\ \ \mathbf{z}_c]^T \ , \tag{4.43}$$

where **R** is defined in (4.34). The set of n_r center-of-rotation vectors are used in Section 4.3.3 to construct a kinematic model of the manipulator.



Begin
$\vec{p}_1, \vec{p}_2, \ldots, \vec{p}_m$
A, B, C, D $\vec{a}$
Compute z_c^* (Eqn 4.36)
Select $\vec{n}$ and $\vec{o}$ Mutually Perpendicular with $\vec{a}$ and Form R
sumg = 0
sumh = 0
j = 1, 2, ... , m
$\vec{p}_j^* = R\,\vec{p}_j$
j = 1, 2, ... , m
$x_j^* \leftarrow x_j^* - \text{sumg}$
$y_j^* \leftarrow y_j^* - \text{sumh}$
$w_j \leftarrow x_j^* + y_j^*$
Solve for Parameters (Eqn 4.23)
sumg ← sumg + g
sumh ← sumh + h
g < epsilon and h < epsilon
F
T
$\begin{vmatrix} x_c \\ y_c \\ z_c \end{vmatrix} = R^T \begin{vmatrix} \text{sumg} \\ \text{sumh} \\ z_c^* \end{vmatrix}$
End

Figure 4-4: Flow Chart of the Repeated Linear Least-Squares Algorithm for Estimating the Center-of-Rotation

4.3.2.3. Line-of-Translation Estimation

Prismatic joints are characterized by a single kinematic feature, their line-of-translation. This feature is defined by two components -- a unit direction vector indicating the orientation of the line and a vector indicating the position of an arbitrary point on the line.

The standard form for the equation of line in 3-dimensions is

$$\frac{x-x_o}{\mathbf{a}} = \frac{y-y_o}{\mathbf{b}} = \frac{z-z_o}{\mathbf{c}} , \tag{4.44}$$

where x_o, y_o, and z_o are the coordinates of a point on the line. Analogous to our approaches taken in Sections 4.3.2.1 and 4.3.2.2, we seek the line (4.44) which minimizes the sum of the squared perpendicular distances between it and the measured target positions. It is relatively simple to prove that the best fit line, in this sense, must pass through the mean of the measurements $\vec{\mathbf{p}}_j$ (i.e., the center of gravity). Hence, the parameter vector

$$\vec{u}_o = \frac{1}{m}\sum_{j=1}^{m} \vec{p}_j , \tag{4.45}$$

where $\vec{u}_o = [x_o \ y_o \ z_o]^T$. We then subtract the mean from the set of measurements to simplify our problem to that of determining the orientation of the line. The standardized measurements are denoted by $\vec{p}_j^{\times}$. To determine the best estimate for **a**, **b**, and **c**, we apply the principal axis method [3, 6]. The vector $\vec{v} = [\mathbf{a} \ \mathbf{b} \ \mathbf{c}]^T$ under the condition that $\vec{v}^T\vec{v} = 1$ represents the unit direction vector defining the orientation of the line (4.44). It also represents the unit normal vector to the plane $\mathbf{a}x + \mathbf{b}y + \mathbf{c}z = 0$. The perpendicular distance between such a plane and the point $\vec{p}_j^{\times}$ is $d_j = \vec{v}^T\vec{p}_j^{\times}$. Consider the plane which maximizes the sum of the squares of the distances d_j. The idea behind the principal axis method is that the line normal to the plane implicitly minimizes the sum of the squares of the distances between it and the points $\vec{p}_j^{\times}$. Hence, we seek the parameters $\vec{v}$ which maximize the error function

$$\Xi \equiv \sum_{j=1}^{m} d_j^2 = \sum_{j=1}^{m} (\vec{v}^T \vec{p}_j^{\times})^2 = \vec{v}^T \mathbf{M} \vec{v} \ , \tag{4.46}$$

where $\mathbf{M}$ is the covariance matrix of the standardized measurements. The vector $\vec{v}$ which maximizes (4.46) is the normalized eigenvector of $\mathbf{M}$ corresponding to the maximum eigenvalue of $\mathbf{M}$. This normalized eigenvector is not uniquely defined. The two possible solutions differ in sign (i.e., they point in opposite directions). In Section 4.3.3, we require that the vector $\vec{v}$ indicate the direction of positive joint translation. According to assumption (B-1), the dot product

$$\vec{v}^T \frac{(\vec{p}_m - \vec{p}_1)}{|\vec{p}_m - \vec{p}_1|} \tag{4.47}$$

between the vector $\vec{v}$ and unit direction vector pointing from the measurement $\vec{p}_1$ to $\vec{p}_m$ should have a magnitude of approximately 1. The sign of (4.47) indicates whether the $\vec{v}$ points in the positive or negative direction of travel. If (4.47) is negative, $\vec{v}$ is multiplied by -1.

We apply the n_p line-of-translation feature estimates, along with the n_r plane-of-rotation and center-of-rotation feature estimates, in the following section to specify the position and orientation of a set of S-Model link coordinate frames. From these link coordinate frames, we formulate a complete manipulator kinematic model and identify the Denavit-Hartenberg kinematic parameters.

4.3.3. Link Coordinate Frame Specification

In Step 2, we apply the identified features to specify the location of the S-Model link coordinate frames which satisfy the S-Model convention (in Section 3.2). We compute the *n+1* matrices

$$\overline{\mathbf{S}}_i = \mathbf{P}\cdot\mathbf{S}_i \qquad \text{for } i=0,\ldots,n \tag{4.48}$$

to define the positions and orientations of the S-Model link coordinate frames and an end-effector coordinate frame with respect to the sensor coordinate frame. In (4.48), **P** is a constant homogeneous transformation matrix representing the spatial transformation from the sensor coordinate frame to the manipulator base coordinate frame S_0. Placement of the sensor system relative to the manipulator is arbitrary, at least from the analytical point-of-view. The set of n constant matrices $\overline{\mathbf{S}}_i$ describes the kinematics of the manipulator in the *signature configuration*. In Section 4.3.5, we generalize this constant model to incorporate the entire joint space.

In Figure 4-5, we illustrate the construction of $\overline{\mathbf{S}}_i$ when joint *i+1* is revolute.The position and orientation of the i th link coordinate frame is specified by the identified joint *i+1* features. By definition

$$\overline{\mathrm{S}}_\mathrm{i} = \begin{bmatrix} \vec{n}_\mathrm{i} & \vec{o}_\mathrm{i} & \vec{a}_\mathrm{i} & \vec{p}_\mathrm{i} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.49}$$

The unit direction vector $\vec{a}_i$ is the unit normal vector to the estimated plane-of-rotation of the *i+1* th joint. The direction of either the X axis, defined by $\vec{n}_i$, or the Y axis, defined by $\vec{o}_i$, is arbitrary in the S-Model (in Section 3.2). Any method for selecting and orthogonalizing these vectors can be applied. We choose the unit direction vector $\vec{n}_i$ to be

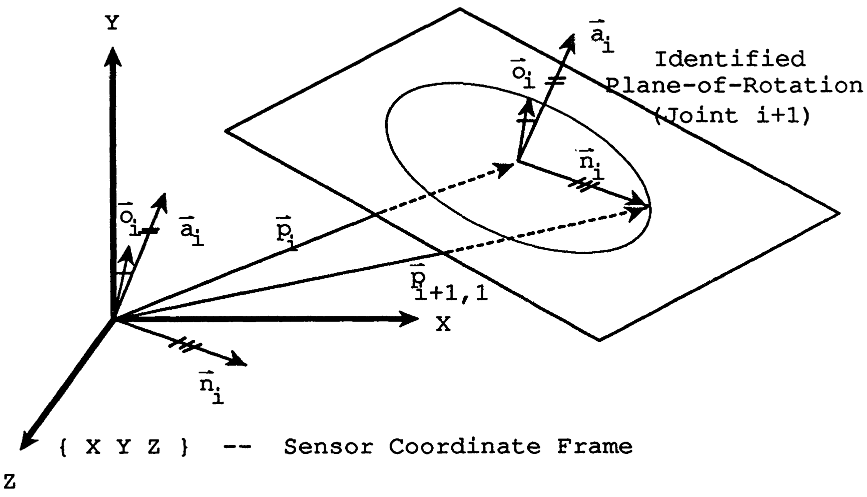


Figure 4-5: Coordinate Frame Construction for a Revolute Joint

$$\vec{n}_i = \frac{\vec{p}_{i+1,1} - \vec{p}_{i+1,c}}{|\vec{p}_{i+1,1} - \vec{p}_{i+1,c}|} , \tag{4.50}$$

where $\vec{p}_{i+1,1}$ is the location of the first target position for joint *i+1* projected onto the plane-of-rotation. For convenience, we have used the first target position. The Y axis direction vector $\vec{o}_i$ is the cross product of the Z and X axis unit direction vectors $\vec{a}_i$ and $\vec{n}_i$, respectively. The origin of the link *i* coordinate frame $\vec{p}_i$ is the center-of-rotation $\vec{p}_{i+1,c}$. When computing $\vec{a}_i$ from the plane-of-rotation, the positive sense of $\vec{a}_i$ must correspond with the positive sense of rotation of the *i+1* th joint.

In Figure 4-6, we illustrate the construction of $\overline{S}_i$ for a prismatic joint. The unit direction vector $\vec{a}_i$ in (4.49) is the unit direction vector to the es-

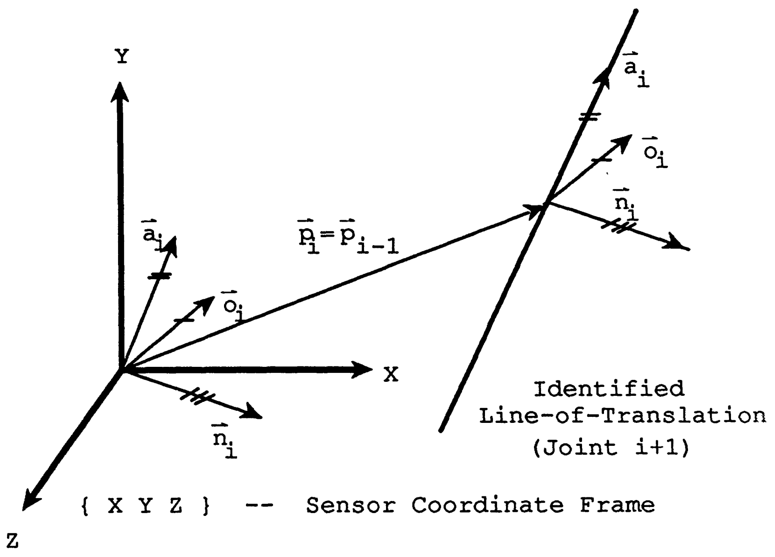

Y
X
Z
$\vec{a}_i$
$\vec{o}_i$
$\vec{n}_i$
$\vec{p}_i=\vec{p}_{i-1}$
Identified Line-of-Translation (Joint i+1)
{ X Y Z } -- Sensor Coordinate Frame

Figure 4-6: Coordinate Frame Construction for a Prismatic Joint

timated line-of-translation of the *i+1* th joint. Hence, $\vec{a}_i=\vec{v}$ where $\vec{v}$ is defined in Section 4.3.2.3. It is essential that the positive sense of $\vec{a}_i$ corresponds with the positive sense of translation of the *i+1* th joint. Again, the direction of either the X axis, defined by $\vec{n}_i$, or the Y axis, defined by $\vec{o}_i$, is arbitrary. To avoid additional computations, we choose $\vec{n}_i$ to be either of the two eigenvectors previously computed in Section 4.3.2.3 which correspond to the two smallest eigenvalues. The Y axis direction vector $\vec{o}_i$ is the cross product of the Z and X axis unit direction vectors $\vec{a}_i$ and $\vec{n}_i$, respectively. Unlike a revolute joint, a prismatic joint does not have a unique axis. In general then, the origin of the *i* th link coordinate frame is arbitrary. For convenience, however, we define $\vec{p}_i$ to be the mean of the measurements $\vec{u}$ defined in Section 4.3.2.3.

The matrix $\overline{\mathbf{S}}_n$ represents the position and orientation of the end-effector with respect to the end-effector coordinate frame. There are no features associated with this frame since the end-effector is assumed to have zero degrees of freedom. The end-effector in this sense refers to the face plate or mounting flange attached to the last joint. For convenience, the end-effector coordinate frame S_n is defined to be coincident with the *n-1* th link coordinate frame when the manipulator is the *signature configuration* (i.e., $\overline{\mathbf{S}}_n = \overline{\mathbf{S}}_{n-1}$).

These simple construction methods are applied to evaluate the elements of the *n+1* matrices $\overline{\mathbf{S}}_i$ which are then used to compute the S-Model parameters.

4.3.4. S-Model Parameters

In Step 3, we compute the transformation matrices $\mathbf{B}_i$ from the $\overline{\mathbf{S}}_i$ matrices according to (3.1) and (4.48)

$$\begin{aligned} \mathbf{B}_i &= \mathbf{S}_{i-1}^{-1}\ \mathbf{S}_i \\ \mathbf{B}_i &= \mathbf{S}_{i-1}^{-1}\ \mathbf{P}^{-1}\,\mathbf{P}\,\mathbf{S}_i \\ \mathbf{B}_i &= \overline{\mathbf{S}}_{i-1}^{-1}\ \overline{\mathbf{S}}_i \qquad\qquad \text{for } i=1,\ldots,n \ . \end{aligned} \tag{4.51}$$

We then apply Paul's backward multiplication technique [18] to compute the six constant transformation matrix parameters β_i, $\overline{\mathbf{d}}_i$, $\overline{\mathbf{a}}_i$, $\overline{\alpha}_i$, γ_i, and $\mathbf{b}_i$ from $\mathbf{B}_i$. The expressions for these parameters in terms of the elements of $\mathbf{B}_i$ were derived in Section 3.3 and are repeated here for clarity and completeness. They are

$$\beta_i = \begin{cases} 0 & \text{when } a_x = a_y = 0 \\ \operatorname{atan} \dfrac{-a_x}{a_y} & \text{otherwise} \end{cases} \tag{4.52}$$

$$\bar{\alpha}_i = \operatorname{atan} \frac{a_x \sin\beta_i - a_y \cos\beta_i}{a_z} \tag{4.53}$$

$$\gamma_i = \operatorname{atan} \frac{-o_x \cos\beta_i - o_y \sin\beta_i}{n_x \cos\beta_i + n_y \sin\beta_i} \tag{4.54}$$

$$\bar{\mathbf{a}}_i = p_x \cos\beta_i + p_y \sin\beta_i \tag{4.55}$$

$$\mathbf{b}_i = \begin{cases} \dfrac{-p_x \sin\beta_i + p_y \cos\beta_i}{\sin\bar{\alpha}_i} & \text{if } \sin\bar{\alpha}_i \neq 0 \\ 0 & \text{if } \sin\bar{\alpha}_i = 0 \end{cases} \tag{4.56}$$

$$\bar{\mathbf{d}}_i = p_z - \mathbf{b}_i \cos\bar{\alpha}_i \;, \tag{4.57}$$

where

$$\mathbf{B}_i = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.58}$$

4.3.5. Denavit-Hartenberg Parameters

In the fourth and final step of our identification algorithm, we extract the Denavit-Hartenberg parameters from the S-Model parameters to model the kinematics of the manipulator over the entire joint space. The physical position of a joint, as measured by the joint encoders, may not coincide with the value of the joint position as defined in the Denavit-Hartenberg model. The constant offset between these two measurements is denoted by $\mathbf{q}_i^{offset}$. In terms of the signature configuration, $\mathbf{q}_i^{offset}$ is

$$\mathbf{q}_i^{offset} = \mathbf{q}_i^s - \overline{\mathbf{q}}_i^s , \tag{4.59}$$

where $\overline{\mathbf{q}}_i^s$ is the encoder measured *signature configuration* position of joint i and $\mathbf{q}_i^s$ is the corresponding Denavit-Hartenberg position of joint i computed according to (3.12) for a revolute joint or (3.13) for a prismatic joint. The Denavit-Hartenberg model parameters, $\mathbf{q}_i = \theta_i$ for a revolute joint and $\mathbf{q}_i = \mathbf{d}_i$ for a prismatic joint, defined in (2.9) are thus

$$\mathbf{q}_i = \overline{\mathbf{q}}_i + \mathbf{q}_i^{offset} . \tag{4.60}$$

The parameters θ_i and $\mathbf{d}_i$ are functions of the controllable and measurable position $\overline{\mathbf{q}}_i$. With the exception of θ_1 or $\mathbf{d}_1$, the remaining $(3 \cdot n)-1$ constant Denavit-Hartenberg parameters are computed according to (3.12) - (3.15). For manipulators with prismatic joints, these are actually the modified Denavit-Hartenberg parameters (i.e, the parameter $\mathbf{a}_i$ for a prismatic joint is nonzero). In most applications, determination of the true Denavit-Hartenberg parameters is not necessary. However, if they are required, slight modifications must be made to the identified S-Model.

The Denavit-Hartenberg parameters of manipulators with prismatic joints can be obtained using the following method. For expository purposes, we will describe the method for a manipulator with *n-1* revolute joints and one prismatic joint. Let joint i denote the prismatic joint. Our objective is to implicitly redefine the location of the link coordinate frame $\mathbf{S}_{i-1}$ depicted in Figure 3-2. Altering the location of $\mathbf{S}_{i-1}$ will naturally affect the computation of the S-Model parameters for link *i-1*. The new location of the coordinate frame is chosen to insure that the parameter $\bar{\mathbf{a}}_i$ in the S-Model is zero. When we later apply (3.12) - (3.15) to determine the Denavit-Hartenberg parameters, $\mathbf{a}_i$ will be identically zero.

The first step is to determine the S-Model parameters of all the links except for links *i-1* and *i* using the relations (4.52) - (4.57). We then recognize that the coordinate frame S_{i-1} can always be relocated, while still satisfying the S-Model convention, such that the parameters $\mathbf{b}_{i-1}$, $\mathbf{b}_i$, and γ_{i-1} are redundant. For convenience, we let $\mathbf{b}_i = -\bar{\mathbf{d}}_{i+1}$ and set $\mathbf{b}_{i-1}$ and γ_{i-1} equal to zero. The parameter $\bar{\alpha}_i$ represents the angle between the joint *i* and *i+1* axes. The unit vectors describing the orientation of these axes, with respect to the coordinate frame S_{i-1} are $[0\ 0\ 1]^T$ and the vector $\vec{a}$ in $\mathbf{B}_i$, respectively. The solution for $\bar{\alpha}_i$, derived from the dot product and cross product of these two vectors, is

$$\bar{\alpha}_i = \operatorname{atan}\frac{a_x^2 + a_y^2}{a_z} , \tag{4.61}$$

where a_x , a_y , a_z are elements of $\mathbf{B}_i$. Our relocation of S_{i-1} includes a rotation of the X and Y axes. The new X axis is defined to point in the direction of the vector formed by the cross product of the joint *i* and *i+1* axes. Since the orientation of the Z axis of S_{i-1} must remain the same, the new Y axis merely completes the orthogonal set. Having rotated the axes, we can then solve for the parameter γ_i . The new X axis direction vector, measured with respect to the original coordinate frame S_{i-1}, is $[-a_y\ \ a_x\ \ 0]^T$. The parameter γ_i represents the angle between this new X axis and the X axis of coordinate frame S_i . Using the definition of the dot product and the cross product,

$$\gamma_i = \operatorname{atan}\frac{(a_x n_z)^2 + (a_y n_z)^2 + (a_y n_y + a_x n_x)^2}{-a_y n_x + a_x n_y} . \tag{4.62}$$

The new $\mathcal{S}_{i-1}$ coordinate frame is, in fact, the Denavit-Hartenberg link coordinate frame $\mathcal{T}_{i-1}$ in Figure 3-2 . Furthermore, the transformation between $\mathcal{S}_{i-2}$ and $\mathcal{S}_i$ is independent of the physical location of $\mathcal{S}_{i-1}$. This transformation is defined by the matrix product

$$\begin{aligned}\mathbf{B}_{i-1}\mathbf{B}_i = \; & Rot(z,\beta_{i-1})Trans(0,0,\overline{\mathbf{d}}_{i-1})Trans(\overline{\mathbf{a}}_{i-1},0,0) \\ & Rot(x,\overline{\alpha}_{i-1})Rot(z,\gamma_{i-1})Trans(0,0,\mathbf{b}_{i-1})Rot(z,\beta_i) \\ & Trans(0,0,\overline{\mathbf{d}}_i)Trans(\overline{\mathbf{a}}_i,0,0)Rot(x,\overline{\alpha}_i)Rot(z,\gamma_i) \\ & Trans(0,0,\mathbf{b}_i) \;, \end{aligned} \tag{4.63}$$

which is derived from (3.2). Setting $\mathbf{b}_i=-\overline{\mathbf{d}}_{i+1}$ and $\gamma_{i-1}=\mathbf{b}_{i-1}=\overline{\mathbf{a}}_i=0$ in (4.63) and rearranging yields

$$\begin{aligned}\mathbf{B}_{i-1}\mathbf{B}_i & Trans(0,0,\overline{\mathbf{d}}_{i+1})Rot(z,-\gamma_i)Rot(x,-\overline{\alpha}_i) = \\ & Rot(z,\beta_{i-1})Trans(0,0,\overline{\mathbf{d}}_{i-1})Trans(\overline{\mathbf{a}}_{i-1},0,0) \\ & Rot(x,\overline{\alpha}_{i-1})Rot(z,\beta_i)Trans(0,0,\overline{\mathbf{d}}_i) \;. \end{aligned} \tag{4.64}$$

The left-hand side of (4.64) can be evaluated using the parameters $\overline{\alpha}_i$ and γ_i obtained from (4.61), (4.62), and (4.54), and the known matrices $\mathbf{B}_{i-1}$ and $\mathbf{B}_i$. Conversely, the right-hand side of (4.64) is a function of the unknown parameters β_{i-1} , $\overline{\mathbf{d}}_{i-1}$, $\overline{\mathbf{a}}_{i-1}$, $\overline{\alpha}_{i-1}$, β_i , and $\overline{\mathbf{d}}_i$. Upon comparison, the right-hand side of (4.64) is functionally identical to that of the general transformation matrix in (3.2). Applying the appropriate substitutions to (4.52) - (4.57), the solutions for the six the remaining unknown S-Model parameters are:

$$\beta_{i-1} = \begin{cases} 0 & \text{when } a_x=a_y=0 \end{cases} \tag{4.65}$$

$$\quad \operatorname{atan} \frac{-a_x}{a_y} \qquad\qquad \text{otherwise}$$

$$\bar{\alpha}_{i-1} = \operatorname{atan} \frac{a_x \sin\beta_{i-1} - a_y \cos\beta_{i-1}}{a_z} , \tag{4.66}$$

$$\beta_i = \operatorname{atan} \frac{-o_x \cos\beta_{i-1} - o_y \sin\beta_{i-1}}{n_x \cos\beta_{i-1} + n_y \sin\beta_{i-1}} , \tag{4.67}$$

$$\bar{\mathbf{a}}_{i-1} = p_x \cos\beta_{i-1} + p_y \sin\beta_{i-1} , \tag{4.68}$$

$$\bar{\mathbf{d}}_i = \begin{cases} \dfrac{-p_x \sin\beta_{i-1} + p_y \cos\beta_{i-1}}{\sin\bar{\alpha}_{i-1}} & \text{if } \sin\bar{\alpha}_{i-1} \neq 0 \\ 0 & \text{if } \sin\bar{\alpha}_{i-1} = 0 \end{cases} , \tag{4.69}$$

$$\bar{\mathbf{d}}_{i-1} = p_z - \bar{\mathbf{d}}_i \cos\bar{\alpha}_{i-1} , \tag{4.70}$$

where n_x , n_y , n_z , o_x , o_y , a_z , a_x , a_y , and a_z are the elements of the matrix defined by the left-hand side of (4.64). When two or more consecutive joints are prismatic, the corresponding S-Model link coordinate frames should be relocated such that their origins coincide. Solutions for the nonzero S-Model parameters, analogous to those in (4.65) - (4.70), can then be derived.

The Denavit-Hartenberg parameters θ_1 and $\mathbf{d}_1$ are functions of the S-Model parameters γ_0 and $\mathbf{b}_0$, respectively. In principle, γ_0 and $\mathbf{b}_0$ can be computed from the elements of the matrix $\mathbf{B}_0$. From (4.51), however, $\mathbf{B}_0$ is undefined since neither $\bar{\mathbf{S}}_{-1}$ nor equivalently $\mathbf{S}_{-1}$ is defined in our identification algorithm. Fortunately, θ_1 and $\mathbf{d}_1$ are not essential for the kinematic modeling and control of manipulators. We circumvent the problem by modifying the model in (2.8). We replace the matrix $\mathbf{T}_n$ by $\bar{\mathbf{T}}_n$ to represent the position and orientation of the nth link coordinate frame in terms of the *identified* S-Model base coordinate frame S_0. To distinguish between the modified model and the original Denavit-Hartenberg model in (2.8), we express $\overline{\mathbf{T}}_n$ as

$$\begin{aligned} \overline{\mathbf{T}}_n &= \overline{\mathbf{A}}_1 \cdot \mathbf{A}_2 \cdot \ldots \cdot \mathbf{A}_n \\ &= [\,\mathrm{Rot}(z, -\gamma_0)\,\mathrm{Trans}(0,0,-\mathbf{b}_0)\,]\,\mathbf{T}_n \ , \end{aligned} \tag{4.71}$$

where the matrix $\overline{\mathbf{A}}_1 = \overline{\mathbf{A}}_1(\overline{\mathbf{q}}_1)$ has the functional form of (2.9), and $\mathbf{A}_i = \mathbf{A}_i(\overline{\mathbf{q}}_i)$ for $i = 2, \ldots, n$. We replace $\mathbf{q}_1$ by

$$\overline{\theta}_1 = \overline{q}_1 + (\beta_1 - \overline{q}_1^s) \tag{4.72}$$

for a *revolute* joint and

$$\mathbf{d}_1 = \overline{\mathbf{q}}_1 + (\overline{\mathbf{d}}_1 - \overline{\mathbf{q}}_1^s) \tag{4.73}$$

for a *prismatic* joint. We call the model in (4.71) the *pseudo* Denavit-Hartenberg model. Even though the parameters γ_n and $\mathbf{b}_n$ are not required to formulate the pseudo Denavit-Hartenberg model in (4.71), we compute *all* of the S-Model parameters in the third step of our identification procedure. The parameters γ_n and $\mathbf{b}_n$ may be required to invert (4.71).

The model (4.71) describes the spatial relationship between a coordinate frame attached to the base of the manipulator and a coordinate frame attached to the end-effector as a function of the joint positions as defined by the joint encoders. Robot kinematic models, such as (2.8), (3.1), and (4.71), only describe the internal kinematics of the device. The location of the link coordinate frames with respect to physical points or surfaces on the mechanical linkages, is not specified by these models. In particular, the physical location of the base frame and the end-effector frame are unknown. In practice, most robot manufacturers select the parameters of the design model so as to insure that the base frame and end-effector frame coincide with certain physical features of the base and end-effector (e.g., machined bosses or the surface of a mounting flange). This, in principle, allows the users to precisely locate the manipulator in a workcell and to program the robot motions in terms of absolute coordinates. This approach to robot programming and control has numerous disadvantages.

The preferred approach is to generate task descriptions and control programs which are independent of the manipulator. All motions of the end-effector are referenced to an arbitrary coordinate frame called the task coordinate frame. Prior to executing the task, the robot is taught the location of the task coordinate frame. Thus, in the practical implementation of robot tasks the absolute physical locations of the base frame and end-effector coordinate frames are not necessary.

The location of targets *1* and *n*, in the S-Model identification algorithm, determines where the base and end-effector coordinate frames lie. Their location and the location of the remaining *n*-2 targets has, conceptually, no affect upon identified internal kinematics of the manipulator. If, however, the application demands that the robot be positioned in a particular reference frame, the relationship between the base and end-effector frame must be specified with respect to the physical device. This is accomplished by mounting targets *1* and *n* at predefined locations which satisfy the necessary physical constraints.

4.4. Conclusions

The S-Model identification algorithm possesses the following features for accurate kinematic parameter estimation and robot modeling:

- Since the S-Model is an exact kinematic description, the estimated model approaches the ideal model as the measurement noise is reduced to zero. Simulation experiments have demonstrated this property of the model.
- The sensor system for measuring the position of the target point is *independent* of the robotic manipulator and eliminates the need for sophisticated fixturing and accurate placement of the manipulator. Acoustic and optical sensor systems for this application are commercially available.
- Placement of the target points on the robot is essentially arbitrary and requires only minor fixturing.
- Simple *linear* least-squares algorithms are applied to estimate the parameters. Solving a huge nonlinear minimization problem, as proposed by Whitney and Lozinski [30], is thus eliminated.
- One can calculate the standard Denavit-Hartenberg parameters from the S-Model parameters.
- The effects of measurement noise upon the estimated parameters can be reduced by increasing either the number of target locations measured per circle (line) or the nominal target radius (for revolute joints).
- The approach leads to a robust algorithm for identifying the kinematic structure of all n degree-of-freedom robotic manipulators.
- The systematic structure of the algorithm leads to efficient and practical implementations requiring little or no operator interaction (as demonstrated by our implementation described in Chapter 6).

Chapter 5
Inverse Kinematics

5.1. Overview

The inverse kinematic problem requires the solution of a set of *n* coupled nonlinear equations. Since arm signature models do not possess closed-form inverse kinematics, numerical methods must be applied to solve the nonlinear equations. In this chapter, we describe two algorithms for solving the inverse kinematic problem. The first algorithm, based upon Newton's method, was originally proposed by Khosla, et al. [12]. We have applied this algorithm in our hardware implementation and evaluation of arm signature identification. The second algorithm is the Jacobi iterative algorithm applied specifically to the inversion of signature models. In the context of this dissertation, we consider the implementation of the inverse kinematic equations to be synonymous with the robot's kinematic control algorithms. Used by themselves to position the end-effector, the inverse kinematic equations correspond to an *open-loop* control structure. Closed-loop or sensory feedback techniques are more appealing but the real-time measurement of the end-effector's position and orientation is, in general, impractical with current technologies. In this chapter, the term kinematic control will be used instead of the term inverse kinematics.

5.2. Newton-Raphson Algorithm

The kinematics of serially-connected robotic manipulators can be described by the multiplicative matrix model

$$\mathbf{T}_n = \mathbf{K}_1(\mathbf{q}_1)\cdot\mathbf{K}_2(\mathbf{q}_2)\cdot \ldots \cdot\mathbf{K}_n(\mathbf{q}_n) \;, \tag{5.1}$$

where the $\mathbf{K}_i$ matrix can be expressed as

$$\mathbf{K}_i(\mathbf{q}_i) = Rot(z,\mathbf{q}_i)\cdot\overline{\mathbf{K}}_i \qquad \text{if joint } i \text{ is revolute} \tag{5.2}$$

or

$$\mathbf{K}_i(\mathbf{q}_i) = Trans(0,0,\mathbf{q}_i)\cdot\overline{\mathbf{K}}_i \qquad \text{if joint } i \text{ is prismatic.} \tag{5.3}$$

The constant homogeneous transformation matrix $\overline{\mathbf{K}}_i$ defines the constant component of the transformation from the link *i-1* coordinate frame to the link *i* coordinate frame. The matrix $\mathbf{T}_n$ defines the position and orientation of the end-effector in terms of a coordinate frame fixed relative to the base of the manipulator.

For kinematic control, our objective is to determine the joint coordinates $\vec{\mathbf{q}}^*$ which correspond to the desired end-effector position and orientation $\mathbf{T}_n = \mathbf{T}_n^*$. When the nonlinear equations in (5.1) are solved iteratively by the Newton-Raphson algorithm [12], the solution at the (*k+1*) th iteration is

$$\vec{\mathbf{q}}^{k+1} = \vec{\mathbf{q}}^k - \mathrm{J}^{-1}(\vec{\mathbf{q}}^k)\cdot\vec{\delta}^k \;, \tag{5.4}$$

where $\mathrm{J}(\vec{\mathbf{q}}^k)$ is the manipulator Jacobian evaluated at $\vec{\mathbf{q}}^k$. The vector $\vec{\delta}^k$ represents the differential translation and rotation between the position and orientation of the desired end-effector location $\mathbf{T}_n^*$ and the predicted end-effector location corresponding to $\vec{\mathbf{q}}^k = [q_1^k q_2^k \ldots q_n^k]^T$ defined in (5.1). The differential translation and rotation vector $\vec{\delta}^k$ is defined in terms of the base coordinates in (5.1). Any set of rotation angles with a predefined sequence of rotations (e.g., Euler angles, roll-pitch-yaw, and Cartesian x-y-z) can be used to define the rotational components of $\vec{\delta}^k$. The choice only affects the formulation of the manipulator Jacobian. Expressions for the manipulator Jacobian in terms of the elements of the $\mathbf{K}_i$ matrices for different definitions of $\vec{\delta}^k$ are presented in [7, 18].

The Newton-Raphson algorithm (5.4) converges in theory when the difference, $\mathbf{T}_n(\vec{q}^{\,k+1}) - \mathbf{T}_n^*$, becomes zero. In practice, two approaches can be taken to assess convergence. In the first, the algorithm is assumed to converge when each element of the above difference has reached a user-prescribed threshold. The second approach assumes that the algorithm has converged when each component of the incremental change in the solution is less than the resolution of the corresponding joint encoder. The latter approach is easier to implement and takes into account the physical limitations of the device itself.

Application of numerical methods to solve the inverse kinematic problem gives rise to the following issues:

- Selection of an initial estimate
- Rate of convergence
- Convergence to the desired solution
- Presence of singularities

Selection of an initial estimate, in our application, is relatively simple. Since the robot design and identified signature models are nearly identical in terms of predicting end-effector position and orientation, we apply the design model to initialize $\vec{q}^{\,0}$. Our initial estimate $\vec{q}^{\,0}$ is then the inverse solution of the design model evaluated at the desired end-effector location $\mathbf{T}_n^*$. The number of iterations required for (5.4) to converge is directly related to the accuracy of the initial estimate. Since the initial estimate obtained from the design model is close to the true solution, we require only a few iterations. In our experimental evaluation of the Newton-Raphson algorithm we have found that the algorithm converges to the true solution within 2-3 iterations.

Fully articulated manipulators, such as the Puma 560, possess multiple inverse kinematic solutions. These multiple solutions differ by such physical characteristics as the right and left shoulder configurations of the Puma 560. For kinematic models with closed-form inverses, a parameter to specify the unique characteristics of a desired solution can be incorporated explicitly into the inverse algorithm. This approach, however, does not apply to kinematic models without closed-form inverse solutions. Consequently, the issue arises as to how to insure that the iterative solution converges to the desired solution. Since our initial estimate, which is derived from a closed-form inverse model, is close to the true solution, we select the initial estimate to be of the configuration of the desired solution.

The Newton-Raphson algorithm has two practical disadvantages. First, it requires the evaluation of the inverse Jacobian at each iteration. Inversion of the Jacobian matrix is a major contributor to its computational complexity. If the desired end-effector position $\mathbf{T}_n^*$ corresponds to a joint configuration which is close to being degenerate, the algorithm in (5.4) becomes numerically unstable (i.e., the Jacobian matrix becomes singular). Degenerate or near-degenerate configurations can almost always be avoided by modifying the robot trajectory planner. This approach often reduces manipulator dexterity and programmability, thereby limiting the range of applications to which the robot can be applied. Second, the Newton-

Raphson algorithm requires a significant number of multiplications, additions, and trigonometric function calls at each iteration (refer to Section 5.5). Computationally intensive operations include generation of the Jacobian elements, and evaluation of (5.1) and the elements of $\vec{\delta}^k$. The Newton-Raphson algorithm is more computationally complex than the closed-form inverse solution typically applied to kinematic control.

5.3. Jacobi Iterative Method

The extent to which arm signature identification and control is applied to improve robot positioning performance will, in part, be determined by the computational complexity of the signature control algorithm. In this section, we apply the *Jacobi iterative* algorithm to invert arm signature models as an alternative to the computationally complex Newton-Raphson algorithm. In contrast to the Newton-Raphson algorithm, the Jacobi iterative algorithm explicitly incorporates the knowledge of both the robot design and the signature models.

We introduce the following notation. Let the forward kinematics, according to the design model, be denoted by $\mathbf{T}_n^d = \mathbf{F}_d(\vec{q})$ and the inverse kinematic model by $\vec{q} = \mathbf{F}_d^{-1}(\mathbf{T}_n^d)$. Similarly, let the forward kinematics, according to the signature model, be denoted by $\mathbf{T}_n^s = \mathbf{F}_s(\vec{q})$ and the inverse kinematic model by $\vec{q} = \mathbf{F}_s^{-1}(\mathbf{T}_n^s)$. Finally, let

$$\mathbf{T}_n^* = \mathbf{F}_s(\vec{q}) \ . \tag{5.5}$$

Our objective is to determine the joint coordinates $\vec{q}^*$ so that $\mathbf{F}_s(\vec{q}^*)$ is equal to the desired end-effector position and orientation $\mathbf{T}_n^*$ (i.e., to compute $\vec{q}^* = \mathbf{F}_s^{-1}(\mathbf{T}_n^*)$). We premultiply both sides of (5.5) by $[\mathbf{F}_s(\vec{q})]^{-1}$ and the result by the design model $\mathbf{F}_d(\vec{q})$ to obtain

$$\mathbf{F}_d(\vec{q}) = \mathbf{F}_d(\vec{q}) \cdot [\mathbf{F}_s(\vec{q})]^{-1} \cdot \mathbf{T}_n^* \ . \tag{5.6}$$

By applying the inverse design kinematic model relationships $\mathbf{F}_d^{-1}(\vec{q})$ to both sides of (5.6), we obtain the *fixed point* equation

$$\vec{q} = \mathbf{F}_d^{-1}[\mathbf{F}_d(\vec{q}) \cdot [\mathbf{F}_s(\vec{q})]^{-1} \cdot \mathbf{T}_n^*] \ . \tag{5.7}$$

We thus seek the joint coordinates $\vec{q}^*$ which are the *fixed point* of the right-hand side of (5.7) [9]. According to the Jacobi iterative algorithm, we can obtain a solution by iterating the right-hand side of (5.7). The Jacobi iterative algorithm is

$$\vec{q}^{k+1} = \mathbf{F}_d^{-1}[\mathbf{F}_d(\vec{q}^k) \cdot [\mathbf{F}_s(\vec{q}^k)]^{-1} \cdot \mathbf{T}_n^*] \ , \tag{5.8}$$

which is initialized by setting $\vec{q}^0 = \mathbf{F}_d^{-1}(\mathbf{T}_n^*)$. We select the initial estimate to be of the configuration of the desired solution as we did for the Newton-Raphson algorithm.

The algorithm (5.8) has a relatively simple physical interpretation. In Figure 5-1, we represent the end-effector positions and orientations defined by the matrices $\mathbf{F}_s(\vec{q}^k)$, $\mathbf{F}_d(\vec{q}^k)$, and $\mathbf{T}_n^*$ by three generic vectors in six dimensions. The vector denoted by Δ^k in Figure 5-1 represents the difference between the solution at the k th iteration and the desired solution, measured in Cartesian space. Analytically, Δ^k is the product

$$\Delta^k = [\mathbf{F}_s(\vec{q}^k)]^{-1} \cdot \mathbf{T}_n^* \tag{5.9}$$

contained in (5.8) and is the error term which drives the algorithm towards the solution. The new Cartesian position and orientation

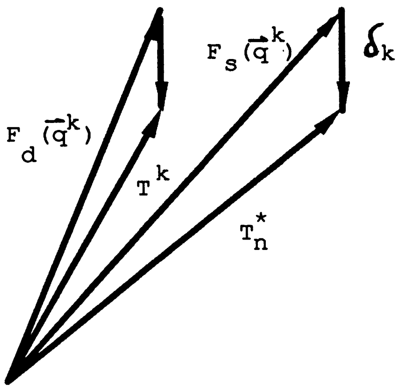


Figure 5-1: Physical Interpretation of the Jacobi Iterative Algorithm

$$T^k = F_d(\vec{q})\cdot\Delta \tag{5.10}$$

in (5.8) is then input to the inverse design kinematic model to compute the new solution $\vec{q}^{k+1}$. At each iteration, the location T^k is translated and rotated by the same amount as the error between the desired and signature model end-effector locations. Implicitly, the algorithm searches for the end-effector location T^k corresponding to T_n^* such that

$$F_d^{-1}(T^k) = F_s^{-1}(T_n^*) \ . \tag{5.11}$$

In the Jacobi algorithm, the design model $F_d(\vec{q}^k)$ and the signature model $F_s(\vec{q}^k)$ must both be expressed in terms of the same physical base coordinate frame as illustrated by the common origin in Figure 5-1.

We have applied this algorithm in simulation to invert signature models. Our simulations demonstrate that the algorithm in (5.8) converges to the true solution $\vec{q}^*$ if the product

$$\mathbf{P}(\vec{q}) = \mathbf{F}_d(\vec{q}^k)\cdot[\mathbf{F}_s(\vec{q}^k)]^{-1} \tag{5.12}$$

is close to the identity matrix. The matrix $\mathbf{P}(\vec{q})$ is the transformation between the end-effector position and orientation predicted by the design and the signature models. Since robot manufacturing errors are small in comparison with the physical dimensions of the robot, the mappings between the joint and Cartesian spaces defined by the two models will, in practice, be similar. Hence, for joint configurations $\vec{q}^k$ in the vicinity of $\vec{q}^*$, the model mismatch (5.12) will vary only slightly in comparison to the workspace of the manipulator (e.g., for the Puma 560, typical values for the model mismatch are on the order of 0.5 mm and 0.1 degrees).

Henrici [9] has shown that the Jacobi iterative algorithm is guaranteed to converge if the function on the right-hand side of (5.7) is a continuous contraction mapping. While we cannot prove that this function is a continuous contraction mapping due to the analytic complexity of the design and signature models and the inverse design model, the simulations illustrate the fact that the Jacobi iterative algorithm has a linear rate of convergence and that the Newton-Raphson algorithm has a quadratic rate of convergence. We assume that the algorithm has converged when all the components of the incremental change in the solution are less than the resolution of their corresponding joint encoder. In all of our simulations, the Jacobi iterative algorithm converged to the solution (within the limits of the Puma 560 joint encoders) within at most 9 iterations while the Newton-Raphson algorithm required 3-4 iterations to converge to an equally accurate solution.

The algorithm in (5.8) requires matrix multiplication, evaluation of the *closed-form* inverse of a homogeneous transformation matrix, and the evaluation of the inverse design model. Since homogeneous transformation matrices are orthogonal, their closed-form inverses are written by inspection. Even though the Newton-Raphson algorithm requires fewer iterations than the Jacobi iterative algorithm to converge, our simulations indicate that the reduced number of iterations may not offset their additional complexity. Computationally, the Newton-Raphson and Jacobi Iterative algorithm are both significantly more complex than the closed-form inverse model for a Puma 560. In Sections 5.4 and 5.5, we evaluate the comparative performance and real-time applicability of the Newton-Raphson and Jacobi Iterative algorithms. The computational complexity, the rate of convergence, and the effects of reduced precision computation upon the performance of these algorithms will be among the features to be compared.

5.4. Performance Evaluation

Our evaluation of the performance of the Newton-Raphson algorithm (5.4) and the Jacobi Iterative algorithm (5.8) consists of comparing their rates of convergence and the effects of reduced precision computation. The convergence of the control algorithms depends upon the overall mismatch between the design and signature models and the desired end-effector position and orientation. In simulation, we applied both algorithms to numerically invert identified arm signature models of a perfectly manufactured Puma 560 robot (i.e., the manufacturing errors are assumed zero). The identified arm signature models of the Puma 560 were obtained from a simulator (described in Chapter 7). With this second simulator, we identified arm signatures of varying accuracy.

Each control algorithm was used to invert three different signature models for n different desired end-effector locations (i.e., the input to the control algorithm). The three signature models span a wide range in accuracy as determined by the magnitude of simulated measurement noise. The n end-effector locations are chosen to span the full range of the Puma 560's workspace. At each iteration of each algorithm, we compute the error Δ^k with respect to the base coordinates. The orientational component of the error is expressed in terms of Euler angles. We also compute the radial position error as the Euclidean norm of the X, Y, and Z axis translational errors. The four translational errors and three rotational errors were computed during each iteration of the two algorithms for the three signature models and the various end-effector locations.

For all cases tested, the behavior of the two algorithms was qualitatively equivalent. As the accuracy of the identified signature model increased the number of iterations required by both algorithms to converge decreased. In all but a few cases, the three orientational and four translational measures of the error term Δ^k converged asymptotically. The difference between the solutions obtained using single precision arithmetic versus double precision arithmetic was insignificant[6]. Tables 5-1 and 5-2 list the components of the error between the desired end-effector location and the end-effector location computed using the solution at each iteration for the Newton-Raphson algorithm and the Jacobi Iterative algorithm, respectively.

Table 5-1 illustrates the *quadratic* convergence of the Newton-Raphson algorithm as compared to the *linear* convergence of the Jacobi algorithm in Table 5-2. In this example, the Jacobi algorithm required nine iterations to converge while the Newton-Raphson algorithm required only four. The errors listed in Tables 5-1 and 5-2 were computed using the same identified

[6]Our simulations were conducted on a Digital Equipment Corporation VAX 11/780 and implemented in the C Programming Language.

Table 5-1: Typical Convergence Behavior of the Newton-Raphson Algorithm

Iteration	X Axis Error (cm)	Y Axis Error (cm)	Z Axis Error (cm)	Radial Error (cm)
0	4.6465327	4.8498771	1.1024610	6.8063937
1	0.0271668	0.0072422	-0.0710502	0.0764108
2	0.0000035	0.0000184	-0.0000147	0.0000238
3	0.0000096	-0.0000160	-0.0000000	0.0000187

Table 5-2: Typical Convergence Behavior of the Jacobi Iterative Algorithm

Iteration	X Axis Error (cm)	Y Axis Error (cm)	Z Axis Error (cm)	Radial Error (cm)
0	4.6465327	4.8498771	1.1024610	6.8063937
1	-0.2821745	0.2077792	-0.0052285	0.3504596
2	0.0264061	-0.0120712	-0.0386135	0.0483115
3	0.0107435	-0.0024874	-0.0036998	0.0116318
4	-0.0011183	0.0000441	0.0011644	0.0016151
5	-0.0000703	0.0000454	-0.0000066	0.0000839
6	0.0000188	-0.0000212	-0.0000116	0.0000306
7	0.0000123	-0.0000165	-0.0000007	0.0000206
8	0.0000092	-0.0000160	0.0000004	0.0000184
9	0.0000096	-0.0000160	-0.0000000	0.0000186

signature model and same desired end-effector location. For expository purposes, the signature model was grossly inaccurate. The error computed at the *0* th iteration is in fact a measure of this inaccuracy expressed in Cartesian space. Recall that the initial solution is obtained from the closed-form inverse design model. The radial component to the error is 6.806 centimeters. In practice, the radial error for a Puma 560 robot will typically be less than 5.0 millimeters. In this example, the Newton-Raphson algorithm requires approximately 2 iterations to converge while the Jacobi algorithm requires approximately 3-5 iterations. Based upon the results of numerous simulations, we conclude that both algorithms are numerically stable and extremely robust.

In the next section, we analyze the computational complexity of the Newton-Raphson and Jacobi Iterative algorithm and assess their real-time applicability.

5.5. Comparative Computational Complexity

The computations required by the closed-form inverse design model (in Appendix B) for a Puma 560 are outlined in Table 5-3.

Table 5-3: Computational Complexity of the Closed-Form Inverse Kinematic Model of the Puma 560 Robot

Variable	atan2	sin	cos	+	×	$\sqrt{\ }$
θ_1	2	0	0	3	2	1
θ_3	1	1	1	5	5	1
θ_{23}	1	1	1	4	6	0
θ_2	0	0	0	1	0	0
θ_4	1	0	1	3	7	0
θ_5	1	0	1	2	5	0
θ_6	1	0	1	7	15	0
Total	7	2	5	25	40	2

The first column indicates the joint position or sum of joint positions to which the operations correspond. Similarly, the computations required by one iteration of the Newton-Raphson algorithm and one iteration of the Jacobi Iterative algorithm for a Puma 560 are outlined in Tables 5-4 and 5-5, respectively.

Table 5-4: Computational Complexity of One Iteration of the Newton-Raphson Algorithm

Step	sin	cos	+	×	/
1	6	6	172	266	0
2	0	0	12	0	0
3	0	0	6	9	0
4	0	0	27	36	0
5	0	0	0	0	0
6	0	0	45	45	15
7	0	0	6	0	0
8	0	0	0	0	0
Total	6	6	268	356	15

The Newton-Raphson algorithm and the Jacobi algorithm are described in terms of a sequence of steps. The eight steps of the Newton-Raphson algorithm and the six steps of the Jacobi algorithm are defined in Appendix C. The total number of operations for one iteration of each algorithm are also listed in the tables. The total complexity of each algorithm is obtained by multiplying these totals by an upper bound for the number of iterations required for convergence. The most computationally intensive step in the Newton-Raphson algorithm is the evaluation of the inverse jacobian. The majority of the computations involved in this task actually pertain to evaluating the forward signature model.

Table 5-5: Computational Complexity of One Iteration of the Jacobi Iterative Algorithm

Step	sin	cos	atan2	+	×	$\sqrt{\ }$
1	6	6	0	0	0	0
2	0	0	0	154	236	0
3	0	0	0	22	50	0
4	0	0	0	6	9	0
5	0	0	0	54	72	0
6	2	5	7	25	40	2
Total	8	11	7	261	407	2

The execution times of the Motorola 68020 microprocessor hardware instructions with a 68881 floating point co-processor are listed in Table 5-6 [16, 17].

Table 5-6: Execution Times of the Motorola 68020 with a 68881 Floating Point Co-Processor (12-MHz Clock)

Function	Time (μs)
Multiply	9.42
Add	7.75
Divide	12.08
Square Root	12.25
Sine	35.92
Cosine	35.92
Arc Tangent	36.92

The times listed in Table 5-6 are based upon a 12 megahertz clock frequency.

These floating point operation times, including the times required to load and store the operand, are used to estimate the time required to compute the inverse kinematic solution. The execution times of the 68020/68881 are representative of the computational capabilities of microprocessors currently being applied to robot control. The estimated computation time for the closed-form inverse model is 1.1 ms. In contrast, the Newton-Raphson algorithm requires 17.77 ms (assuming three iterations to converge) and the Jacobi algorithm requires 33.99 ms (assuming five iterations to converge).

The *cycle* time for the Puma 560 is 28.0 ms[7] corresponding to a sample rate of 39 Hz. During each cycle, the trajectory planner inputs an end-effector location to the kinematic controller which must then compute the inverse solution and output the result to the various joint servo controllers. The estimates indicate that the Newton-Raphson is implementable in real-time using commercially-available hardware whereas the Jacobi Iterative algorithm is not. The Newton-Raphson algorithm is computationally more efficient than the Jacobi Iterative algorithm in terms of a direct single processor implementation. However, the Jacobi algorithm does have a potential advantage over the Newton-Raphson algorithm. The evaluation of the forward signature model in Step 1 of the Jacobi algorithm involves numerous operations which can be performed in parallel. With the proper hardware, significant reductions in the computation time can be realized. A cursory examination suggests that the time required to perform the operations contained in steps 1 through 5 of the Jacobi Iterative algorithm can be reduced by a factor four yielding a total computation time of approximately 12.6 ms. This is not the case for the Newton-Raphson algorithm where the evaluation of the forward signature model must be performed sequentially to provide

[7]A recent update to the Puma 560 controller includes an 8 ms cycle time.

the intermediate results necessary for computing the elements of the inverse Jacobian matrix.

5.6. Conclusions

In this chapter, two numerical algorithms for solving the inverse kinematic problem have been presented, namely, the Newton-Raphson algorithm and Jacobi Iterative algorithm. The results of numerous simulations, indicate that both algorithms are numerically stable and extremely robust. In our experiments, we have found that the Newton-Raphson algorithm converges to the true solution within 2-3 iterations while the Jacobi Iterative algorithms typically converges within 7-9 iterations.

Computationally, the Newton-Raphson and Jacobi Iterative algorithm are both significantly more complex than the closed-form inverse model. In fact, the closed-form inverse solution is applied in both cases to provide the initial estimate with which to begin. Based upon a serially programmed single processor architecture, the Jacobi Iterative algorithm requires significantly more computation time than does the Newton-Raphson algorithm. Based upon the execution times of the Motorola 68020 microprocessor with a 68881 floating point co-processor the Newton-Raphson solution takes approximately 17.77 ms to compute and the Jacobi Iterative solution takes approximately 33.99 ms. Since the cycle time of a Puma 560 robot is 28.0 ms, the Newton-Raphson algorithm is implementable in real-time using commercially-available hardware.

The Jacobi Iterative algorithm has two distinct advantages over the Newton-Raphson algorithm. First, it is easier to program. The Newton-Raphson algorithm, for instance, requires separate algorithms just to invert the 6x6 manipulator jacobian. Second, the Jacobi Iterative algorithm involves numerous operations which can be performed in parallel which, with the proper processor architecture, could significantly reduce the overall computation time. A rough estimate suggests a total computation time of approximately 12.6 ms which is less than the Newton-Raphson algorithm. The recent development of customized processors for robotic applications is expected to further reduce the computation time.

In the next chapter, we demonstrate both the feasibility and practicality of arm signature identification and control by applying the S-Model identification and Newton-Raphson algorithms in proof-of-concept hardware experiments to improve the performance of seven Puma 560 robots.

Chapter 6

Prototype System and Performance Evaluation

6.1. Overview

In order to demonstrate the feasibility and evaluate the performance of the identification and control algorithms, we have developed a complete prototype system. This prototype arm signature system has been applied to identify and control seven Westinghouse/Unimation Puma 560 robots.[8]. The Puma 560 is a six degree of freedom robot with revolute joints. Evaluation of the arm signature-based control showed consistent improvement of accuracy in several standard tasks.

In this chapter, we describe the hardware/software implementation of the identification and control algorithms and describe the evaluation of kinematic performance.

[8]Puma is a trademark of the Westinghouse/Unimation Corporation

6.2. System Overview

Figure 6-1 illustrates the configuration of the hardware components in the implementation. Figure 6-2 is photograph of the system with the target sparker for Joint 6 mounted in the bar on the end-effector. The sensor array, described in Section 6.3.1, is a rigid fixture which supports four ultrasonic microphones. In the figure, the microphones are labeled as A, B, C, and D. The sensor array also supports to calibration rods indicated by the two sets of parallel lines which intercept with microphones B and D. Microphones A, B, and C form an approximate right triangle with vertex at microphone B. The sensor array is mounted on a wall such that the active volume of the sensor array and the workspace of manipulator which is mounted on a table coincide. The sensor array in combination with the GP-8-3D sonic digitizer are used to measure the straight line distances between the microphones and the target points attached to the robot. These measurements, called *slant ranges*, are then used to determine the Cartesian position of the target points based upon triangulation. The ultrasonic digitizer is interfaced to the Puma Controller using an RS-232 serial line which is itself interfaced to a VAX 11/780 using another RS-232 serial line.

The software which controls the Puma 560 during the collection of the raw target slant ranges is implemented on the Puma's Val II Controller and is written in the Val II programming language [27, 28]. This program is initially downloaded from the VAX 11/780 which maintains current versions of all the system software and provides permanent storage for all experimental data. The software which implements the slant range compensation and target triangulation algorithms (Appendix E.2) is written in the C programming language to run under the Unix[9] operating system on the Digital

[9]Unix is a trademark of AT&T Bell Laboratories, Murray Hill, NJ.

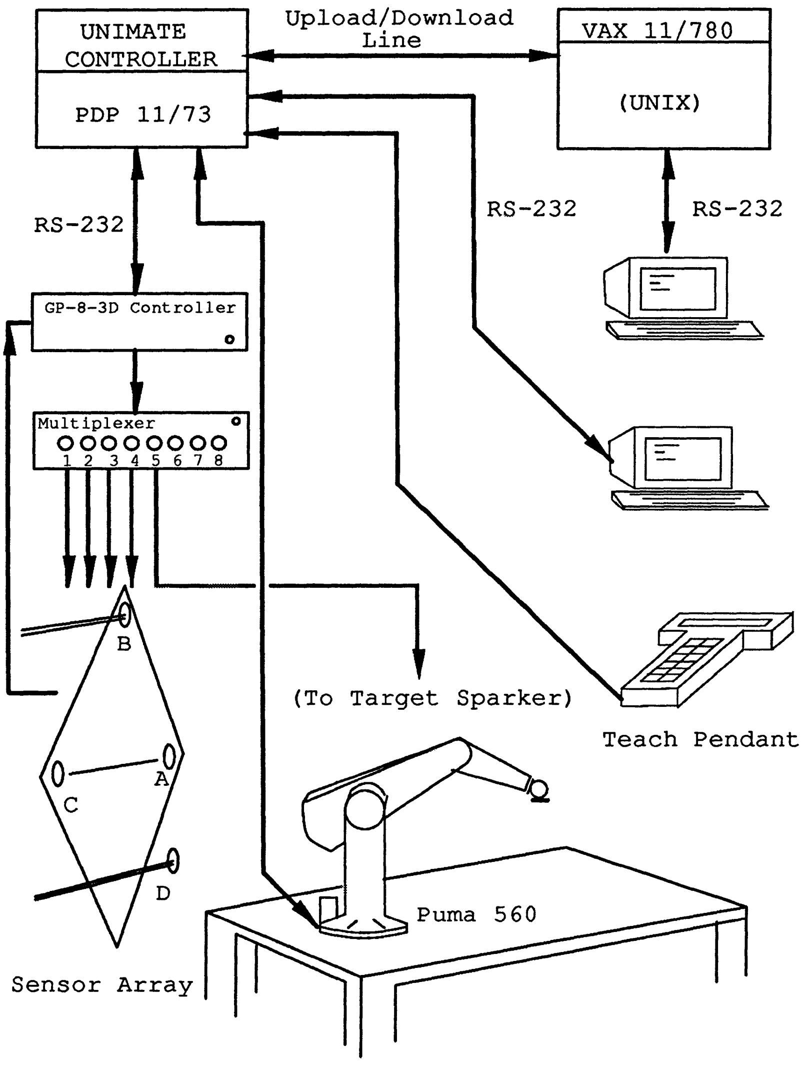


Figure 6-1: Hardware Configuration

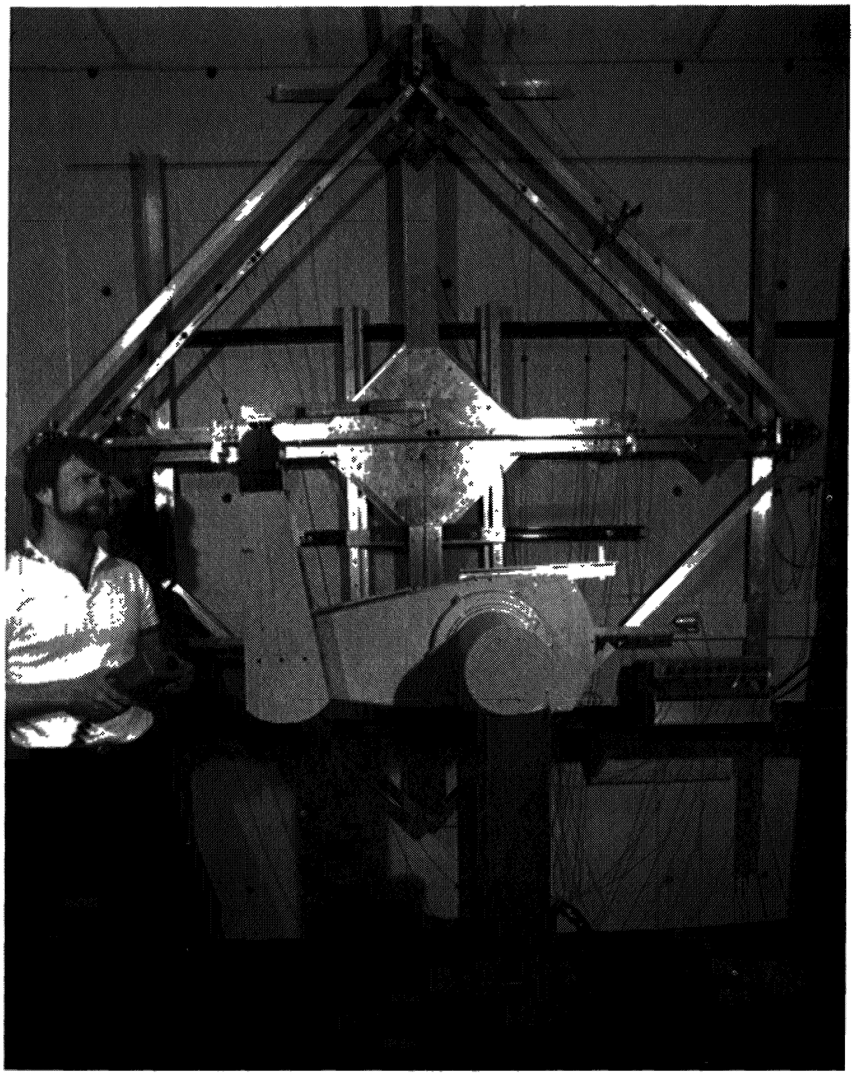


Figure 6-2: Arm Signature Identification System and Puma 560 Robot.

Equipment Corporation VAX 11/780 [11]. The slant range measurements are thus collected by the VAL II controller and then uploaded to the VAX 11/780. The software which implements the feature parameter estimation and computes the signature parameters is also written in the C programming language and runs on the VAX 11/780. Once identified, the arm signature is stored for future use to control the manipulator.

The closed-form inverse kinematics of the Puma 560 (refer to Appendix B.2) and the Newton-Raphson algorithm (in Section 5.2) are implemented in the C programming language on the VAX 11/780. We apply both the closed-form inverse kinematic model and Newton-Raphson algorithm to solve for the joint configurations corresponding to desired Cartesian locations of the end-effector and to control the robot in a variety of tasks. The tasks used to analyze manipulator kinematic performance and the instrumentation required to obtain the pertinent measurements are described in Section 6.5.

6.3. Sensor System

The sensor system measures the three-dimensional positions of targets attached to the robot arm. We use an ultrasonic source (sparker) for each target and an array of four ultrasonic range detectors to estimate target positions. Each detector measures time-of-flight independently from the target and computes an estimated range. Knowing the geometry of the detector configuration the three-dimensional coordinates of the point may be estimated.

6.3.1. Description

The GP-8-3D Sonic Digitizer (manufactured by the Science Accessories Corporation) is an ultrasonic range sensing system which computes range measurements based upon the time of flight of an ultrasonic wave emitted from a source to a set of receiving microphones [22]. The primary components of the GP-8-3D Sonic Digitizer are

- A controller
- A multiplexer
- Four ultrasonic piezo-electric transducers (microphones)
- Eight emitters (sparkers)

The interconnection of these components is illustrated in Figure 6-3.

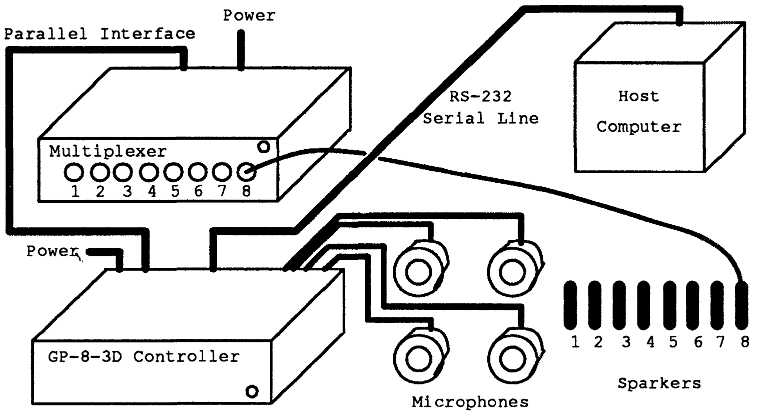


Figure 6-3: GP-8-3D Sonic Digitizer Components

The controller, equipped with an RS-232 serial port, can be interfaced to a host computer to allow for remotely-controlled digitizing and data acquisition. In our application, the host computer is the PDP 11/73 in the Puma 560 Controller. Using the multiplexer, the range measurements to any one of eight different sparkers can be obtained. The GP-8-3D controller has four input channels, one for each microphone. When requested to digitize, the GP-8-3D initiates a control signal which simultaneously triggers the desired sparker to spark and resets four counters.

When the sound wave emitted by the sparker reaches a microphone a level detector is triggered, and the time of flight is recorded. The velocity of sound at ambient temperature is used to compute the estimated range from the time-of-flight and the computed slant ranges are then transmitted to the Puma 560 Controller via the RS-232 serial line.

Our sensor system utilizes five sparkers and four microphones. For identification purposes, the sparkers are numbered 1 through 5 and the microphones are labeled A through D. Sparker 5 represents the target sparker which is attached to each of the links of the manipulator during the identification process. The remaining sparkers and the microphones are mounted at fixed locations relative to one another as illustrated in Figure 6-4. The locations of microphones A, B, and C define the *sensor plane*. The sensor system is mounted on a wall such that the sensor plane is perpendicular to the ground and the line joining microphones A and C is approximately horizontal. In our implementation, the distances between microphones A and B, B and C, and A and C are nominally 143.0 cm, 143.0 cm, and 203.0 cm, respectively. The active volume of the sensor array depicted in Figure 6-4 is a cube approximately 143.0 cm on a side with vertices at microphones A, B, and C.

The Cartesian coordinates of the target sparker, Sparker 5, obtained

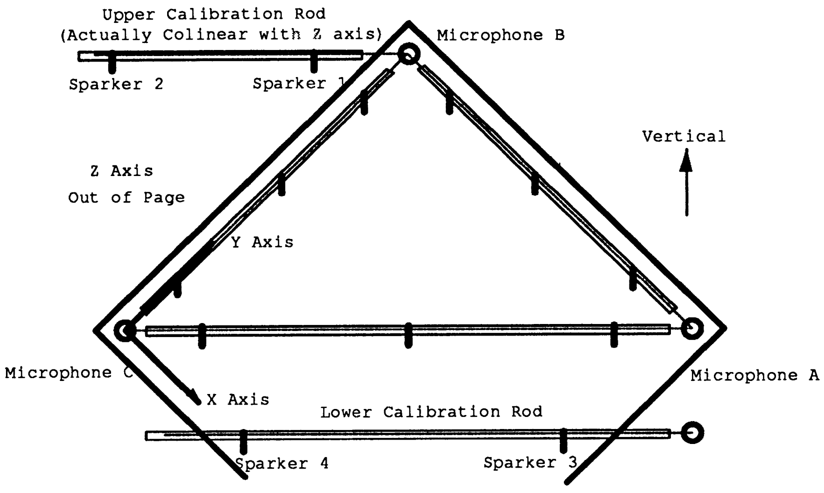


Figure 6-4: Sensor Configuration and Orientation

through triangulation, are referenced to a Cartesian coordinate frame implicitly defined by the location of Microphones A, B, and C. This reference frame is the *Sensor Coordinate Frame* depicted in Figure 6-4. Its origin coincides with the acoustic zero point of Microphone C. The Y axis is directed along the line joining the acoustic zero points of Microphones C and B. The Z axis is perpendicular to and out of the plane defined by the acoustic zero points of Microphones A, B, and C. The X axis, formed by the vector cross product of the Y and Z axis unit direction vectors, completes the orthogonal set.

The GP-8-3D can measure slant ranges between 30 cm and 250 cm with a resolution of .01 cm. The accuracy of the computed slant ranges, however, varies significantly, depending upon the environmental operating conditions. The speed at which sound travels in air is primarily a function of temperature, humidity, altitude, frequency, barometric pressure, and air flow. The speed of sound is particularly sensitive to variations in temperature (1281.4.4 cm/sec/°F at 70.0 °F). For instance, if the actual temperature of the air between a microphone and the sparker is 69.0 °F instead of the assumed 70.0 °F, the GP-8-3D would measure a 150.0 cm slant range as 155.79 cm. In general, differences between the actual and nominal operating conditions produce a difference between the actual and nominal speed of sound and hence, reduce the accuracy of the computed slant ranges. The use of multiple sparkers and a special sensor array design enable us to compensate the slant range measurements taken by the GP-8-3D for variations in the speed of sound (refer to Appendix E). This greatly increases the Cartesian accuracy of the overall sensor system.

Following the compensation procedure described in Appendix E, the final estimates for the three primary slant ranges s_1 , s_2 , and s_3 representing the distances between the target sparker and Microphones A, B, and C, respectively, are applied to determine the Cartesian coordinates of the target. Upon triangulating, the X, Y, and Z coordinates of the target are computed according to

$$x = \frac{r\cos(\theta) + r\tan^2(\theta)\cos(\theta) - y}{\tan\theta} \tag{6.1}$$

$$y = \frac{(w_{B,C})^2 + (s_3)^2 - (s_2)^2}{2w_{B,C}} \tag{6.2}$$

$$z = \sqrt{(s_3)^2 - x^2 - y^2}\ , \tag{6.3}$$

where

$$r = \frac{(w_{A,C})^2 + (s_3)^2 - (s_1)^2}{2w_{A,C}} \tag{6.4}$$

$$\cos(\theta) = \frac{(w_{A,C})^2 + (w_{B,C})^2 - (w_{A,B})^2}{2 w_{A,C} w_{B,C}} \tag{6.5}$$

$$\tan(\theta) = \frac{(1 - \cos^2(\theta))^2}{\cos(\theta)} \tag{6.6}$$

and $w_{A,B}$, $w_{B,C}$, and $w_{A,C}$ are the distances between Microphones A and B, B and C, and A and C, respectively. The coordinate relations (6.1) - (6.3) can be simplified if the triangle formed by Microphones A, B, and C is a perfect right triangle.

6.4. Generating Features

The S-Model Identification algorithm leads to an efficient and well-defined method for collecting measurements. We sequentially attach a sparker to each link of the Puma robot to implement the target points described in Section 4.3.2. The fixtures used to attach the sparker to a link are simple both in design and construction. For example, consider the fixture illustrated in Figure 6-5 which supports the sparker to Link 6 implementing Target Point 6. The two requirements for this fixture are that it be rigid and that the distance between the sparker and the Joint 6 axis be approximately equal to R_6, where R_6 depends upon the active volume of the sensor system. We call R_i the nominal radius of target point i. The nominal radii for the six target points in our implementation are listed in Table 6-1. In general, increasing the nominal radii increases the accuracy of the identified planes-of-rotation.

We begin the measurement process by positioning the manipulator in the desired signature configuration using a teach pendant. Conceptually, the signature configuration can be any arbitrary configuration. However, in practice, the signature configuration may depend upon the characteristics of

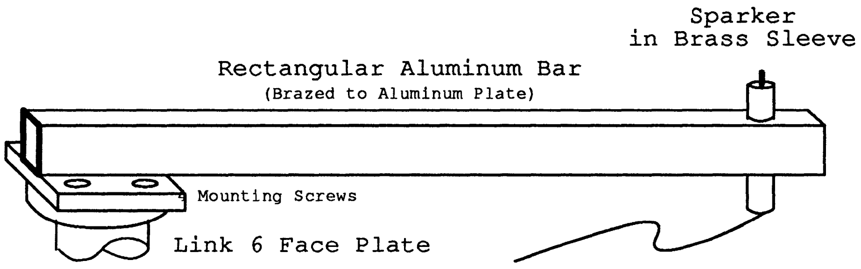


Figure 6-5: Fixture for Attaching Target Point 6 to Link 6

Table 6-1: Puma 560 Target Point Nominal Radii

Target Point	Nominal Radius (cm)
1	40.87
2	41.62
3	43.71
4	33.30
5	45.13
6	33.30

the sensor system. In our implementation, we must insure that the lines of sight between the target sparker and the three Microphones A, B, and C remain unobstructed. Requirements such as this may also affect the design of the mounting fixtures and the selection of the target point's positions with respect to the links. The signature configuration we have selected is illustrated in Figure 6-6.The angular positions of the joints which define the signature configuration are measured and recorded. We denote the signature configuration by the vector $\vec{q}^{\,s} = [q_1^s, q_2^s, \ldots, q_6^s]^T$. In Figure 6-6, $\vec{q}^{\,s}$ is nominally $[155.0\ 179.0\ 178.0\ 0.0\ 0.0\ 0.0]^T$.

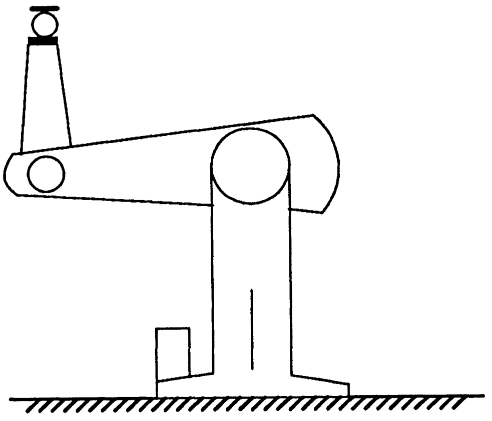


Figure 6-6: Approximate Signature Configuration for the Puma 560

We independently and incrementally vary the position of each joint of the manipulator such that the six target points generate circular trajectories in space. At each fixed configuration of the manipulator, we measure and record the set of seven slant ranges required to determine the Cartesian position of the target sparker. To further improve the estimates of measured target positions, we measure and record several sets (typically 10-20) of seven slant ranges at each configuration. Random fluctuations in the measured slant ranges are then reduced through averaging.

The measured target positions represent sample points on the generated circular trajectories. Six sets of points are obtained, each set corresponding to the trajectory of a distinct target point. The measured target positions will be used to identify 12 kinematic features, six planes-of-rotation and six centers-of-rotation, which are then applied to construct the set of

S-Model link coordinate frame transformation matrices $\mathbf{S}_i$ for $i=1, \ldots, 6$. To identify the manipulator's kinematic parameters, the matrices $\mathbf{S}_i$ must represent the position and orientation of a set of S-Model link coordinate frames when the manipulator is in the specified signature configuration. Consequently, while taking measurements of the i th target's position, joints 1 - $i\text{-}1$ must remain in their respective signature configuration positions (i.e., $q_1 = q_1^s, q_2 = q_2^s, \ldots, q_{i-1} = q_{i-1}^s$). The positions of joints $i+1$ - 6 have no effect upon the position of the i th target point since it is assumed to be rigidly attached to the i th link. Joints $i+1$ - 6 are positioned so as not to limit the motion of joint i or obstruct the lines of sight between the target and the sensor array.

We have established a systematic method for indexing the configuration of the manipulator to insure that the previous conditions are satisfied. Figure 6-7 is a flow chart indicating the sequence of operations in this method. Measurements of the position of Target Point *6* are first obtained, followed successively by those of Target Points *5, 4, 3, 2,* and *1*. In Figure 6-7, N_i denotes the number of distinct positions of joint i at which the position of target i is measured and N_{avg} denotes the number of slant range measurements averaged at each target position. Increasing N_i typically increases the accuracy of the estimated feature parameters. In Section 6.5 we explore the relationship between N_i and the arm signature accuracy. We have identified signatures using as few as five measurements per circle and as many as 100. Each identified arm signature is based upon

$$\sum_{i=1}^{6} 7 \cdot N_{avg} \cdot N_i \tag{6.7}$$

raw slant range measurements. For instance, if $N_i = 100$ for $i=1, \ldots, 6$ and $N_{avg} = 10$, then 42,000 slant range measurements will be required. We are able to obtain these measurements using the hardware in Figure 6-1 in ap-

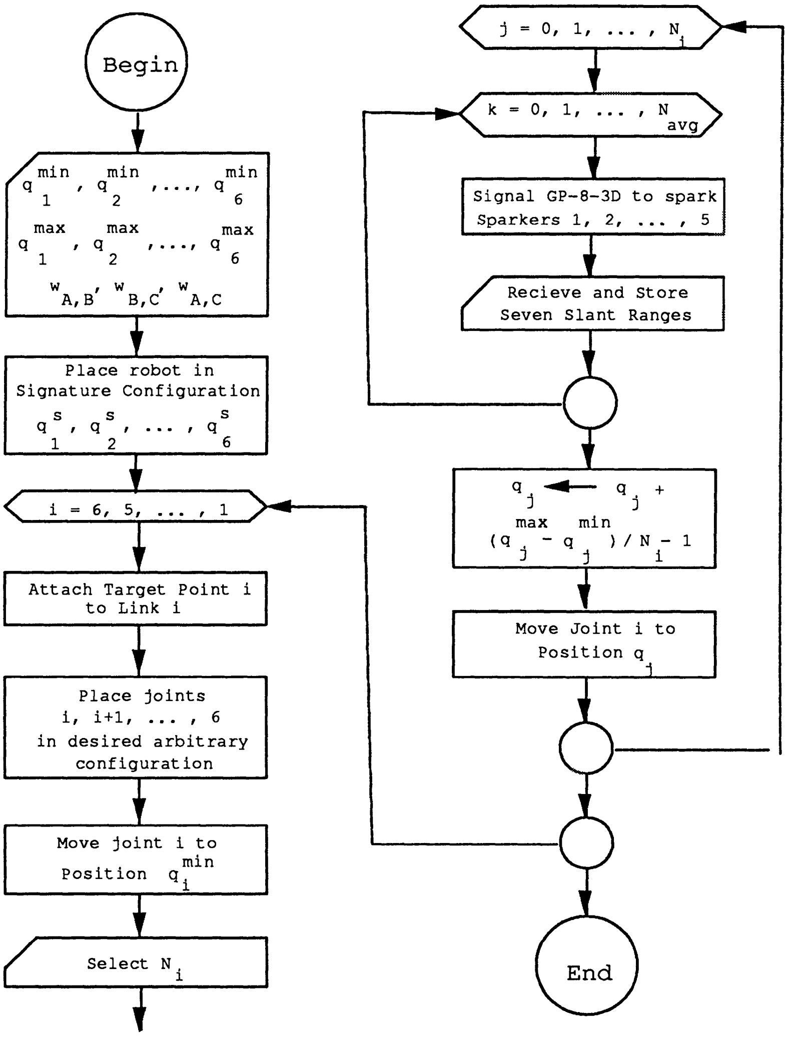

Begin
$q_1^{min}, q_2^{min}, \dots, q_6^{min}$
$q_1^{max}, q_2^{max}, \dots, q_6^{max}$
$w_{A,B}, w_{B,C}, w_{A,C}$
Place robot in Signature Configuration $q_1^s, q_2^s, \dots, q_6^s$
i = 6, 5, ... , 1
Attach Target Point i to Link i
Place joints i, i+1, ... , 6 in desired arbitrary configuration
Move joint i to Position q_i^{min}
Select N_i
$j = 0, 1, \dots, N_i$
$k = 0, 1, \dots, N_{avg}$
Signal GP-8-3D to spark Sparkers 1, 2, ... , 5
Recieve and Store Seven Slant Ranges
$q_j \leftarrow q_j + (q_j^{max} - q_j^{min}) / N_i - 1$
Move Joint i to Position q_j
End

Figure 6-7: Flow Chart of the Algorithm for Collecting Target Point Measurements

proximately 1 1/2 hours. This includes the time required to manually switch the target sparker from one target point fixture to the next. The algorithm described in Figure 6-7 is implemented in the VAL II programming language.

6.5. Measuring Performance

We have devised three methods for measuring manipulator kinematic performance and have applied these methods to evaluate the performance of the seven Puma 560 robots. Each method has an associated task which is performed by the robot. These are

- The *one*-dimensional grid touching task
- The *two*-dimensional grid touching task
- The *three*-dimensional grid touching task

During these tasks, the manipulator is programmed to sequentially position its end-effector at the vertices of a one-, two-, or three-dimensional grid defined with respect to the workspace coordinate frame. At each of the grid points, the actual relative position of the end-effector with respect to the workspace coordinate frame is measured. Measurements of the end-effector's position, with respect to the robot's base coordinate frame cannot be obtained since the physical location of the base frame has not been determined in the signature analysis. The Cartesian positioning accuracy of a robot with respect to a workspace coordinate frame is most useful in applications.

The Puma is initially taught three arbitrary points to define a workspace coordinate frame. The corresponding joint configurations are denoted by the vectors $\vec{q}_1$, $\vec{q}_2$, and $\vec{q}_3$. These configurations are recorded by the Puma Controller. The position and orientation of the end-effector, with respect to the base coordinate frame for each of these configurations, are computed using the appropriate forward kinematic model (i.e., either the signature model or the design model). The origin of the workspace coordinate frame is defined to be coincident with the position of the end-effector at Point 1. The X axis is parallel to the line joining the end-effector's position at Points 1 and 2. The Z axis is normal to the plane defined by the position of the end-effector at Points 1, 2, and 3. Finally, the Y axis completes the orthogonal set. Using this procedure, it is relatively easy to align the workspace coordinate frame with a desired physical coordinate frame such as that defined by the surface of a table.

Our objective is to measure positioning errors resulting from the mismatch between the actual and identified kinematics and the actual and design kinematics. The positioning errors which occur while a robot is in motion are both a function of the kinematic errors and the dynamic control system errors. Consequently, measurements of the end-effector's positions are taken only when the manipulator is stationary in specified fixed configuration. The Puma's steady-state joint positioning errors are typically within ± 1 encoder count ($\pm$ .005 degrees) of the desired positions.

6.5.1. One-Dimensional Grid

The objective in the one-dimensional grid touching task is to determine the accuracy with which the robot can displace its end-effector. The robot is programmed to touch two points in space. The location of these points is specified in terms of Cartesian coordinates and defined with respect to a previously taught workspace coordinate frame. Conceptually, these points represent the endpoints of a traced line. We attach a pushpin to the end-effector and measure the actual distance between the tip of the pin at the two points with a cathetometer. Figure 6-8 illustrates the measurement process.

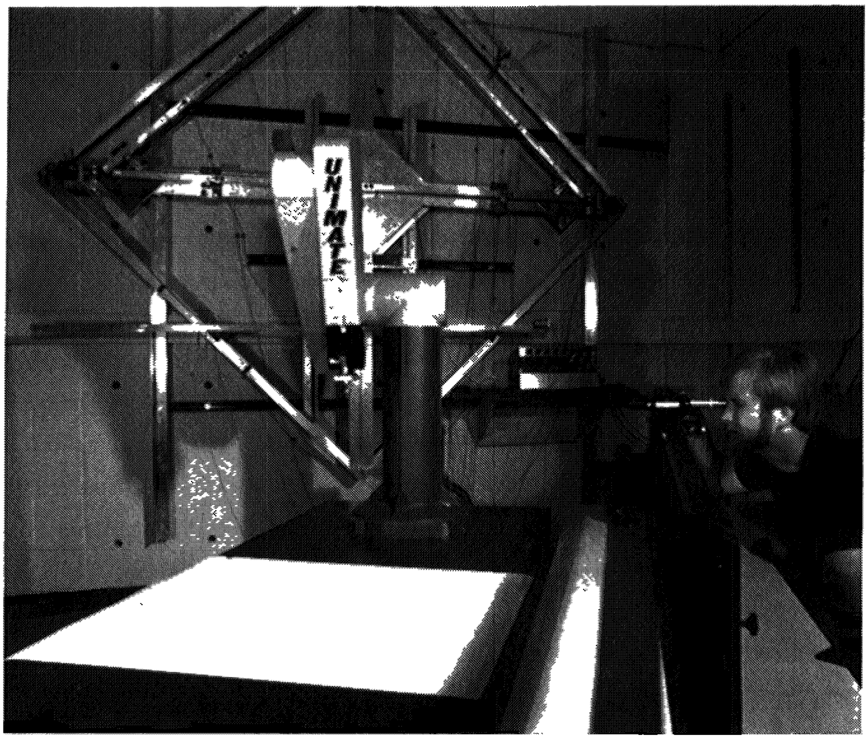


Figure 6-8: Measuring the Performance of a Puma 560 with a Catheto-meter

A cathetometer is an instrument used to measure the differences of levels and changes from a horizontal or vertical line. It consists of a telescope mounted on the carriage of a linear translational stage. The axis of the telescope and the axis of the translational stage are perpendicular. The carriage can be positioned and clamped in an approximate position and then finely adjusted using a micrometer-screw. A vernier scale, which can be read through a separate viewing microscope, measures the relative displacement of the carriage. A small leveling instrument is also mounted on the carriage. The entire stage and carriage assembly can be rotated about the axis of the stage and clamped at any desired position. The base which supports the assembly has adjustable feet for leveling the telescope. Our cathetometer has a range of 1.2 m and a resolution of 0.05 mm (.002 in.).

The axis of the cathetometer is brought into parallel with the line joining the two endpoints through a series of iterative adjustments in the position and orientation of the cathetometer. During this procedure, the robot is programmed to move back and forth between the endpoint locations. Following each move, a machinist's rule and square are used to determine the approximate normal distance between the endpoint locations and the cathetometer axis. Small misalignments of the cathetometer have a negligible effect upon the accuracy of our measurements. Once aligned, the actual measurement is taken and recorded.

In an experiment, we perform the grid touching task twice, once with the ideal parameters and once with the arm signature parameters. In both cases, we use the same physical workspace coordinate frame and endpoint locations. The procedure for specifying the workspace coordinate frame and the endpoint locations must be performed twice since the physical location of the base coordinate frame for the two models will, in general, differ.

Relative end-effector positioning errors are a function of the magnitude of the displacement and the location of the line within the workspace of the robot. These errors tend to increase as the robot approaches full extension and as the magnitude of the displacement increases. In the experiments conducted, we utilize large displacements between the endpoints to increase the observability of the errors. Furthermore, to evaluate the robots performance over its entire workspace, pairs of endpoint locations are selected which span the majority of the robot's workspace. For convenience, the lines formed by the endpoints are approximately aligned with the axes of the ideal model's base coordinate frame.

6.5.2. Two-Dimensional Grid

We evaluated the performance of six of the Puma 560 robots with a two-dimensional grid touching task. During this task, the manipulator was programmed to sequentially position its end-effector at the vertices of a two-dimensional grid defined with respect to a workspace coordinate frame. The workspace coordinate frame is taught using the same procedure as in the one-dimensional case. The grid plane is nominally aligned with the surface of the table upon which the robot is mounted and a custom-designed, spring-loaded pen is attached to the robot's end-effector. The pen assembly, illustrated in Figure 6-9, is adjusted to insure that the pen tip just hits the surface of the table when the end-effector reaches a vertex. The positions of the end-effector (i.e., the pen point) are recorded on a piece of mylar affixed to the surface of the table. In practice, a set of approach points are also defined to insure that the pen generates points on the mylar and not lines. The approach locations are defined to be 3.0 cm above (relative to the table's surface) the vertex locations. The robot is programmed to first position the end-effector at an approach point. At a reduced speed, the end-effector then moves to the corresponding vertex point, generating a spot on the mylar,

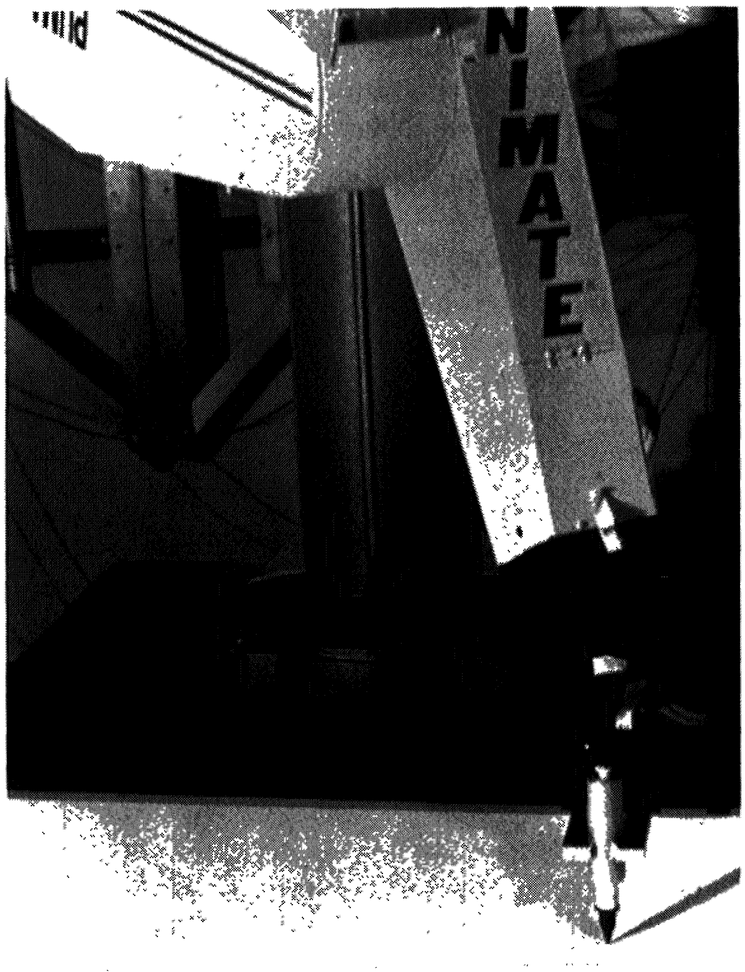


Figure 6-9: Spring-Loaded Pen Assembly Attached to the Puma 560's End-Effector

followed by a return move to the approach point. The speed is temporarily increased during motions between successive approach points.

A machinist's rule is applied to measure the relative positional deviations in the grid points. Features of interest are the perpendicularity of the lines defined by the vertices, line straightness, and line length. The advantage of this approach is that the positional errors in two dimensions can be analyzed. Qualitatively, it is then easy to determine whether the arm signature model provides improved kinematic performance as compared to the ideal model. The disadvantage is that the physical size of the pen tip and the resolution of the rule severely limits the accuracy with which the errors can be measured. Only an approximate quantitative description of the robot's performance was obtained.

6.5.3. Three-Dimensional Grid

A challenging task for analyzing the kinematic performance of a manipulator is the *three*-dimensional grid touching task. The Puma is initially taught three arbitrary points to define a workspace coordinate frame. During this task, the manipulator is programmed to sequentially position its end-effector at the vertices of a three-dimensional grid defined with respect to the workspace coordinate frame. Again, the workspace coordinate frame is taught using the same procedure as in the one-dimensional case. At each of the grid points, the actual position and orientation of the end-effector are measured. To obtain these measurements, we constructed a measurement system using a Ranky sensor jig [21] and a Bridgeport milling machine. The Ranky sensor jig, illustrated in Figure 6-10, consists of three mutually orthogonal plates of steel oriented to form the inner corner of a box. Three precision dial gauges are mounted in each plate. The axes formed by the intersections of the three plates define the jig's Cartesian coordinate frame as

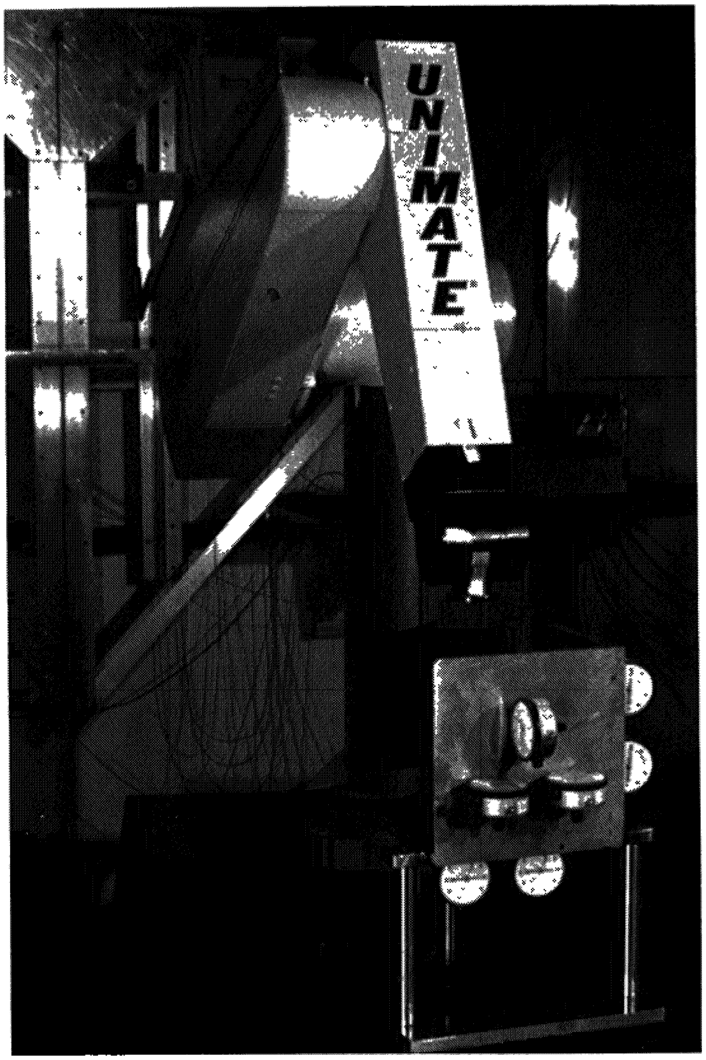


Figure 6-10: Ranky Sensor Jig for Measuring End-Effector Position and Orientation

Used with permission from the Robotics Institute at Carnegie-Mellon University, copyright 1987. Photo taken by Mary Jo Dowling.

indicated in Figure 6-10. A similar box corner, made of an aluminum alloy, is affixed to the end-effector. It too has an associated coordinate frame implicit to its shape. The objective is to place the cube on the end-effector into the jig so as to deflect the nine dial gauges. From the measured deflections and the known placement of the gauges, one can determine the position and orientation of the end-effector relative to the sensor jig.

We mounted the sensor jig on the bed of a three-axis milling machine equipped with linear optical encoders. The jig can thus be displaced accurately over a large volume. A set of approach points are defined to insure that the test cube does not damage the dial gauge stems by hitting them sideways. In addition, the retraction of the test cube from the jig permits indexing of the machine bed. The robot is programmed to first position the end-effector at an approach point. At a reduced speed, the end-effector then moves to the corresponding vertex point, deflecting the nine dial gauges. The end-effector remains at the vertex point while the dial gauges are read and the measurements are recorded. The end-effector then returns to the approach point and the sensor jig is translated to the next vertex point. The speed is temporarily increased during motions between successive approach points. The dial gauges have a resolution of 0.001 in. The accuracy of our sensor jig is 0.001 in. in position and 0.01 degree in orientation. In the experimental results which follow, the grid size was 25 cm x 25 cm x 60 cm. The measured set of dial gauge deflections are uploaded to the VAX 11/780 and stored. Separate software, residing on the VAX 11/780, processes the grid touching measurement data and tabulates indices of kinematic performance. Grid line perpendicularity, line straightness, and line length are the primary features of interest.

The three-dimensional grid touching experiments were conducted in the Civil Engineering Department Machine Shop at Carnegie-Mellon University. Each experiment requires several hours of setup time and approximately 3 hours to collect the measurements. The limited availability of the six robots from Unimation meant that only two Puma 560s could be tested using this technique, the results of which are presented in Section 6.6.3.

6.6. Kinematic Performance Evaluation

We illustrate the performance of the identification algorithm and the subsequent increase in manipulator kinematic performance for seven Puma 560 robots. The Denavit-Hartenberg parameters of the ideal kinematic model of a Puma 560 ***prior to manufacture*** are listed in Table 6-2 [27].

Table 6-2: Unimate Puma 560 Denavit-Hartenberg Kinematic Parameters

Link	Variable	d (cm)	a (cm)	α (deg)
1	θ_1	0.0	0.0	-90.0
2	θ_2	14.909	43.18	0.0
3	θ_3	0.0	-2.032	90.0
4	θ_4	43.307	0.0	-90.0
5	θ_5	0.0	0.0	90.0
6	θ_6	0.0	0.0	0.0

In Table 6-3, we list the pseudo Denavit-Hartenberg parameters identified by the S-Model identification algorithm for Robots 1 and 2. The identified pseudo Denavit-Hartenberg parameters of the five remaining robots are listed in Table D-1 in Appendix D.

The estimated values in Table 6-3 illustrate the sensitivity of the Denavit-Hartenberg parameters to typical mechanical errors in the physical

Table 6-3: Identified Arm Signature Parameters

Name	Link	Variable	θ_{offset} (deg)	d (cm)	a (cm)	α (deg)
Robot 1	1	θ_1	110.308	-20.678	-0.005	-90.066
	2	θ_2	79.977	11,210.213	7.607	-0.217
	3	θ_3	-79.483	-11,194.978	-2.039	90.519
	4	θ_4	-0.296	43.278	-0.001	-90.007
	5	θ_5	-0.497	-0.018	0.002	89.990
	6	θ_6	-89.081	10.300	0.000	0.000
Robot 2	1	θ_1	110.849	-20.428	-0.023	-90.013
	2	θ_2	63.467	2,783.580	19.084	-0.803
	3	θ_3	-63.276	-2,768.574	-2.028	90.134
	4	θ_4	0.328	43.299	-0.009	-89.991
	5	θ_5	-0.126	-0.054	0.007	89.969
	6	θ_6	-89.517	10.273	0.000	0.000

manipulator. For instance, the values of $\mathbf{d}_2$ and $\mathbf{d}_3$ in Table 6-3 indicate that the joint 2 and joint 3 axes are slightly out of parallel. Customarily, this is referred to as the *convergence angle* problem. When these axes are slightly out of parallel, their common normal lies extremely far from the physical manipulator (refer to Figure 2-2). Since the origin of the Denavit-Hartenberg Link 2 coordinate frame is defined by the intersection of the joint 2 axis and this common normal, it too lies extremely far from the physical manipulator. The parameters $\mathbf{d}_2$ and $\mathbf{d}_3$, which are opposite in sign and nearly equal in magnitude, represent the translation out to and back from the link 2 coordinate frame, respectively. The dramatic differences between the identified and the ideal parameters arise from small manufacturing errors.

The error committed in positioning and orienting the robot end-effector is a measure of the cumulative accuracy of the kinematic parameters upon which the control algorithms are based. For the parameters identified in Table 6-3, the cumulative accuracy can also be measured by determining the mean square errors between the Cartesian target positions and the identified features. Intuitively, as the overall accuracy of the identified signature increases these errors decrease.

In Section 4.3.2.1, our estimate of a plane-of-rotation feature minimizes the sum of the squares of the perpendicular errors between the target positions and the plane. Similarly, in Section 4.3.2.2, our estimate of a circle of rotation minimizes the sum of the squares of the perpendicular errors between the projected target positions and the circle. The algorithms presented in Sections 4.3.2.1 and 4.3.2.2, for minimizing these functions, are implicitly based upon the assumption that the measurement errors (i.e., the residual errors) are random and normally distributed. In Figure 6-11, we have plotted the residual errors in fitting a plane and a circle to the measured Cartesian positions of Target Point 6 corresponding to the identified signature of Robot 1 listed in Table 6-3. The perpendicular errors between the target's positions and the identified plane (i.e., the normal residuals) are indicated by asterisks. The circles represent the perpendicular errors between the *projected* target positions and the identified circle (i.e, the radial residuals). These errors are plotted as a function of the measurement number. (The measurement number corresponds to the index i in Figure 4-1.) The behavior of the residuals in Figure 6-11, as a function of the measurement number, is typical of the behavior of the residuals which correspond to the remaining identified signatures in Table D-1. The lack of any significant systematic trends in the normal and radial residuals leads us to assume that the target position errors are random. Furthermore, we have observed that the distributions of the residual errors are closely approximated by the normal distribution. To illustrate, Figures 6-12 and 6-13 contain the histograms of the twelve sets of residual errors corresponding to the identified arm

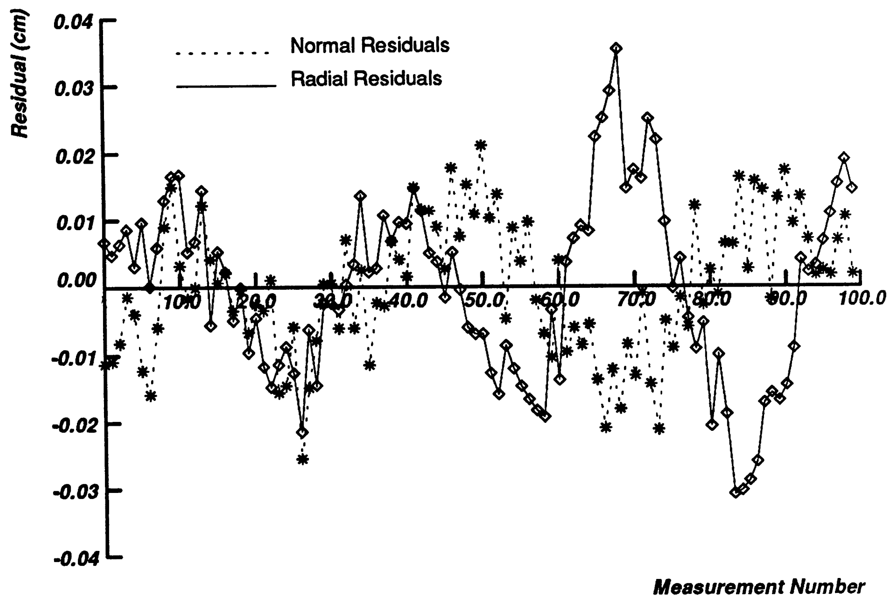


Figure 6-11: Residuals: Normal to Plane and Radial (Joint 6)

signature of Robot 1. The sample variances and standard deviations of these residuals are listed in Table 6-4.

The essential aspect of a plane-of-rotation is its angular orientation. The accuracy of the angular orientation of an identified plane-of-rotation is a function of the accuracy of the target measurements, the number of measurements N_i, and the nominal radius of the target point. In Section 4.3.2.1, we applied simple linear regression techniques to estimate the parameters E, F, and G in (4.19) which define the identified plane-of-rotation. The physical interpretation of these parameters is illustrated in Figure 6-14. It is noted that

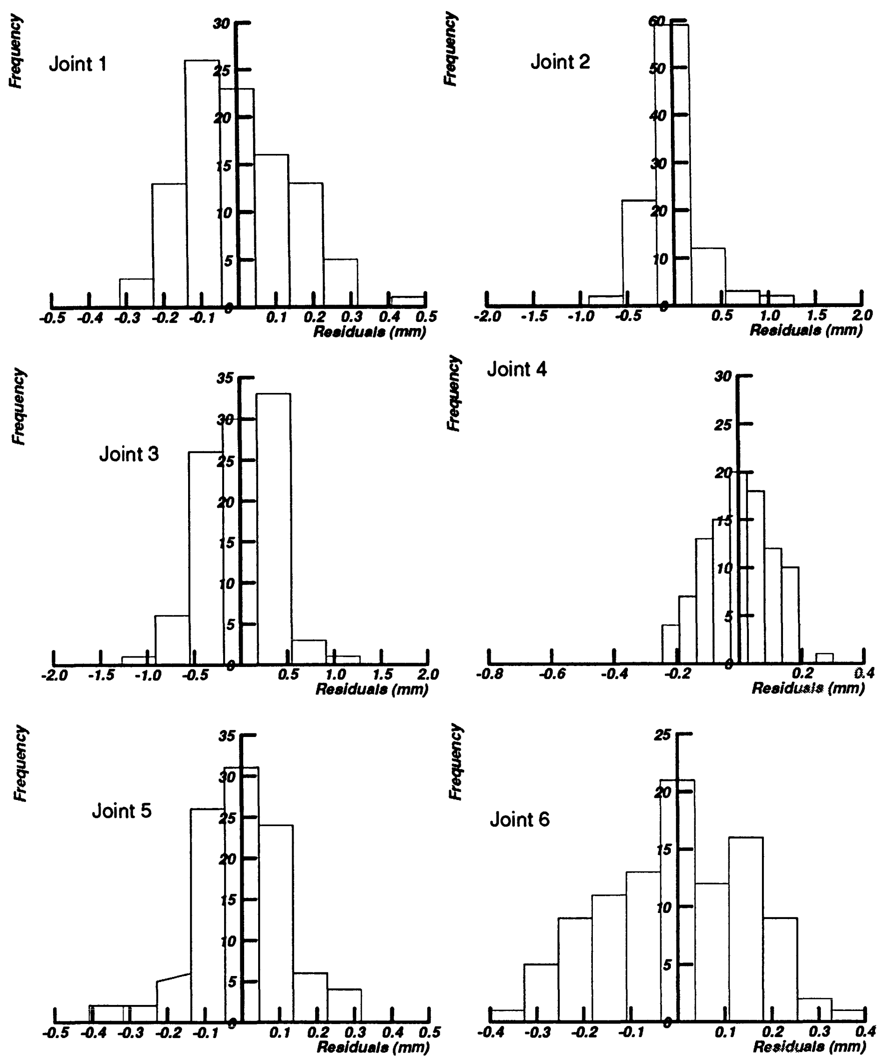


Figure 6-12: Histogram Plots of the Residuals for the Plane-of-Rotation Estimates for Joints 1 through 6

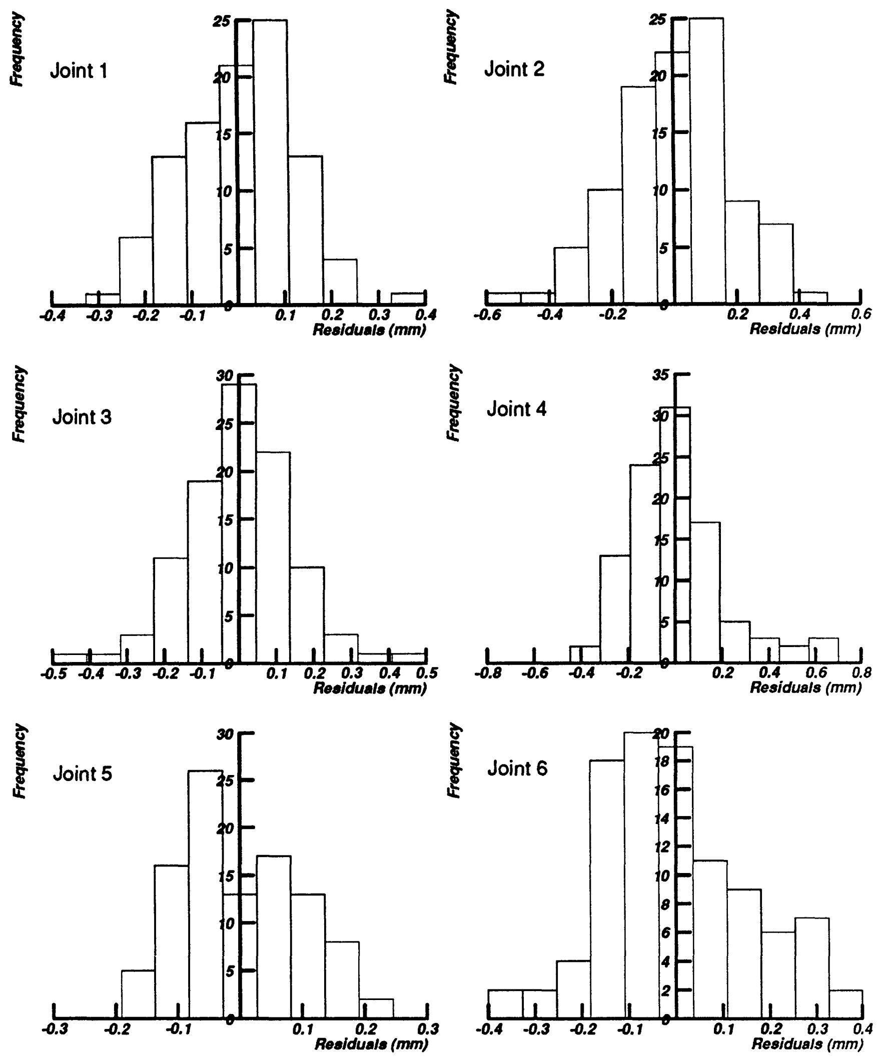


Figure 6-13: Histogram Plots of the Residuals for the Circle-of-Rotation Estimates for Joints 1 through 6

Table 6-4: Sample Variances and Standard Deviations of the Normal and Radial Residuals

Joint	Normal Residuals		Radial Residuals	
	Variance (cm^2)	Stdev (cm)	Variance (cm^2)	Stdev (cm)
1	0.000187	0.0137	0.000133	0.0115
2	0.000848	0.0291	0.000329	0.0181
3	0.001437	0.0379	0.000202	0.0142
4	0.000114	0.0107	0.000416	0.0204
5	0.000142	0.0119	0.000090	0.0095
6	0.000230	0.0152	0.000230	0.0152

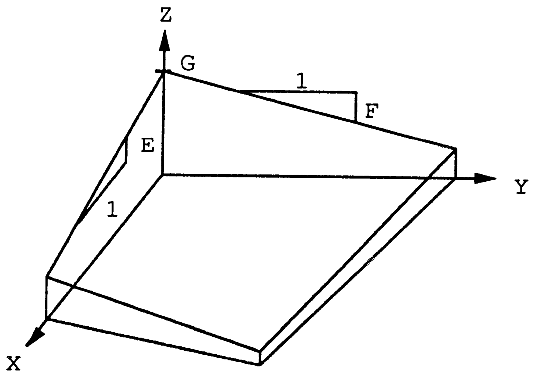


Figure 6-14: An Identified Plane-of-Rotation

during the final iteration of the repeated least-squares algorithm, the estimated parameters E and F, in the model (4.19), will be approximately zero. If the residual errors ξ_j , in (4.22), are independent with zero mean and variance σ_n^2, then the estimate Θ (4.23) will be unbiased [29]. Furthermore, the variance of Θ will be

$$VAR(\Theta) \equiv \sigma_\Theta^2 = \sigma_n^2[\Phi^T\Phi]^{-1} \ , \tag{6.8}$$

where Φ is defined in (4.24). Expanding (6.8) yields

$$VAR(\mathrm{E}) \equiv \sigma_\mathrm{E}^2 = \frac{\sigma_n^2\sum_{j=1}^{n}\hat{y}_j^2}{\sum_{j=1}^{n}\hat{x}_j^2\sum_{j=1}^{n}\hat{y}_j^2 - [\sum_{j=1}^{n}\hat{x}\hat{y}]^2} \tag{6.9}$$

and

$$VAR(\mathrm{F}) \equiv \sigma_\mathrm{F}^2 = \frac{\sigma_n^2\sum_{j=1}^{n}\hat{x}_j^2}{\sum_{j=1}^{n}\hat{x}_j^2\sum_{j=1}^{n}\hat{y}_j^2 - [\sum_{j=1}^{n}\hat{x}\hat{y}]^2} \ , \tag{6.10}$$

where $\hat{x}_j$ and $\hat{y}_j$ represent the original x_j and y_j minus their respective means (i.e., $\hat{x}_j=x_j-\bar{x}$ and $\hat{y}_j=y_j-\bar{y}$). The *estimated* variances of E and F are computed according to (6.9) and (6.10) with σ_n^2 replaced by the sample variance of the normal residuals.

A variation in the parameter E corresponds to an angular variation in the slope of the plane parallel with the X axis. Similarly, a variation in the parameter F corresponds to an angular variation in the slope of the plane parallel with the Y axis. When the parameters E and F are small (i.e., much less than 1.0) and their variances are also small, the variances (6.9) and (6.10) represent the variances of the angular orientation of the plane with respect to the X and Y axes, expressed in units of rad^2. The standard deviations of the estimates E and F, σ_E and σ_F , provide a physically meaningful measure of the orientational accuracy of the estimated plane-of-rotation. These standard deviations translate into confidence intervals which can be compared with manufacturing tolerances to determine whether an identified signature is likely to be more or less accurate than the robot's design model. If the standard deviation in the orientational accuracy of the planes-of-rotation are significantly greater than the orientational accuracy with which the robot can be manufactured, then it is highly probable that the identified signature will actually degrade the performance of the robot.

For many robots, the expressions (6.9) and (6.10) can be simplified due to the symmetry of the target measurements in the X-Y plane. For instance, consider a joint which has 360 degrees of rotation. The X and Y coordinates of the target's positions, with respect to the coordinate frame indicated in Figure 6-14, lie essentially on a circle of radius R_i . According to the methodology outlined in Section 4.3.2.1, these points are equally spaced along the circle's circumference. Consequently, the covariance of the X and Y coordinates of the measurements is zero. Therefore, the term $\sum_{j=1}^{n} \hat{x}_j \hat{y}_j$ in (6.9) and (6.10), equals zero. Furthermore, it can be readily shown that the sums $\sum_{j=1}^{n} \hat{x}_j^2$ and $\sum_{j=1}^{n} \hat{y}_j^2$ are equal to $nR^2{}_i/2.0$. The variance in the angular orientation of the plane is, in this case, the same in all directions and equal to

$$\sigma_\delta^2 \equiv \sigma_E^2 = \sigma_F^2 = \frac{2.0\sigma_n^2}{R_i^2 n} \qquad (\text{in } rad^2). \qquad (6.11)$$

We are thus 95 percent confident that the true angular orientation of the plane lies within $\pm 2\sigma_\delta$ radians of the estimated orientation. The expression (6.11) for the variance is intuitively pleasing. As the accuracy of the measurements decrease the accuracy of the estimate decreases; as the nominal radius of the increases the accuracy of the estimate increases; and as the number of measurements increases the accuracy of the estimate increases. Most importantly, The expression (6.11) provides a simple mechanism for determining how to select the identification parameters n and

R_i given the accuracy of a proposed sensor system and the manufacturing tolerances of the robot.

In general, for a joint with less than 360 degrees of rotation, the variance in the angular orientation of the estimated plane-of-rotation varies with the direction within the plane. The maximum occurs in the direction in which the projection of the measurements onto the plane has the least variance. If, for instance, the minimum and maximum limits on the motion of a joint are 0 degrees and 270 degrees, respectively, then the maximum variance in angular orientation of the estimated plane-of-rotation will occur in the direction of 315 degrees. The exact value for the maximum variance can be computed using (6.9). However, for joints with at least 180 degrees of rotation, we have determined experimentally that the maximum variance is closely approximated by

$$\sigma_\delta^2 \approx \frac{\sigma_n^2}{R_i^2 n(.00244X - .344)}, \tag{6.12}$$

where X represents the joint range of motion in degrees. The expression (6.12) is a conservative approximation.

Table 6-5 lists the variances and standard deviations ofthe angular orientation of the identified planes-of-rotation corresponding to the identified arm signature of Robot 1, listed in Table 6-3. For Joint 6, the orientation of the identified plane-of-rotation is accurate to within $\pm .0074\,deg$. Similar results are obtained for the remaining signatures listed in Table D-1. It is apparent that the orientational accuracies of the identified planes-of-rotation are significantly greater than the machining and assembly accuracies with which commercially available robots are manufactured. Such accuracies may be on the order of hundredths of a degree or more.

Table 6-5: Orientational and Positional Accuracy of the Identified Planes-of-Rotation and Centers-of-Rotation Corresponding to the Arm Signature of Robot 1

Joint	Feature Estimate Accuracy	
	Plane-of-Rotation σ_δ (deg)	Center-of-Rotation σ_c (cm)
1	.002937	.00176
2	.007397	.00335
3	.008514	.00250
4	.003204	.00356
5	.004149	.00184
6	.003699	.00176

The essential aspect of a center-of-rotation is its position relative to a joint axis. The accuracy of the position of an identified center-of-rotation is a function of the accuracy of the target positions measured relative the joint axis. Since the origin of an S-Model link coordinate frame is permitted to lie anywhere on the joint axis, errors in the identified position of a center-of-rotation in the direction of the joint axis, have no effect upon the accuracy of the identified kinematic parameters. In Section 4.3.2.2, we applied simple linear regression techniques to estimate the parameters **A**, **B**, and **C** which define the identified circle-of-rotation. The parameters **A** and **B** are linearly related to the X and Y coordinates of the center-of-rotation. Unfortunately, since the errors in measuring w (4.37), are dependent upon x and y, a derivation of the variances of the estimates of the parameters **A** and **B** analogous to that for the parameters **E** and **F** is impossible.

We have applied Monte-Carlo simulation techniques to empirically determine the relationship between the variance in the estimated coordinates of the center-of-rotation, the variance in the measurements, the range of motion of the joint, and the number of measurements. In these simulations, we have assumed that the errors in the measurements of the target point coordinates, x_j and y_j, are independent and normally distributed with zero mean and variance $\sigma^2 \mathbf{I}_{(2x2)}$. Furthermore, it is assumed that the target radius is much larger than the measurement noise, as would be the case in practice. The simulation experiments demonstrate that the radial residuals (i.e., the errors in w) are essentially characterized by a normal distribution. For joints with 360 degrees of rotation, the variances in the estimated coordinates of the center-of-rotation are equal while for joints with less than 360 degrees of rotation, these variances will differ. For our purposes, we characterize the accuracy of the coordinates of the estimated center-of-rotation by the maximum variance, denoted by σ_c^2. The relationship between the variance σ_c^2, the number of measurements, and the range of motion of the joint is found to be

$$\sigma_c^2 \approx \frac{\sigma_r^2}{n(.00229X - .313)}, \tag{6.13}$$

where σ_r^2 is the variance of the radial residuals and X is the range of motion of the joint measured in degrees. The accuracy of an estimated center-of-rotation can be increased by either increasing the accuracy of the sensor system or increasing the number of measurements.

Table 6-5 lists the positional accuracies of the identified centers-of-rotation for the arm signature of Robot 1. The positional accuracy of the identified Joint 6 center-of-rotation is ±0.00176 cm. Again, this exceeds the machining tolerances with which commercially available robots are manufactured. The accuracies of the identified features of Robots 2 through 7 are essentially the same as those for Robot 1. In the following sections, we demonstrate that the identified arm signatures of Robots 1 through 7 are indeed more accurate than the design model by applying them to improve the kinematic performance of all seven robots.

6.6.1. One-Dimensional Performance Evaluation

We have applied the one-dimensional grid touching task to evaluate the kinematic performance of Robot 1, both with and without arm signature identification. Table 6-6 lists a sampling of results obtained in this task.

Table 6-6: Line Length Errors

Line	Desired Length (cm)	Design Model		Signature Model	
		Length (cm)	Error (cm)	Length (cm)	Error (cm)
1	60.0	59.470	-0.530	59.959	-0.050
2	50.0	50.350	0.350	49.875	-0.125
3	60.0	59.710	-0.290	59.940	-0.060
4	50.0	60.325	0.325	49.895	-0.105
5	60.0	59.375	-0.625	59.960	-0.040
6	60.0	59.725	-0.275	59.955	-0.045

Lines 1 and 4 are nominally aligned with the robot's X axis; Lines 2 and 5 are nominally aligned with the robot's Y axis; and Lines 3 and 6 are nominally aligned with the robot's Z axis. The locations of Lines 1 through 6, relative to the base of the robot, are depicted in Figure 6-15. The approximate coordinates of the Lines 1 through 6 are listed in Table 6-7. (Note that the X-Y plane of the base coordinate frame indicated in Figure 6-15 is coincident with the surface of the table to which the robot is mounted. Formally, the X-Y plane of the base coordinate frame is parallel with the Joint 2 axis.)

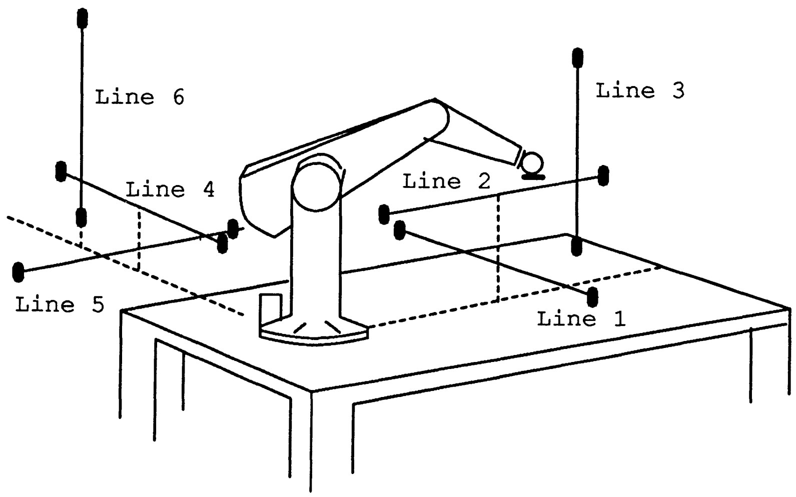


Figure 6-15: Approximate Location of Lines 1 through 6

Table 6-7: Line Coordinates Measured Relative to Base Coordinate Frame

Line	X Coordinate (cm)	Y Coordinate (cm)	Z Coordinate (cm)
1	-	58.0	3.0
2	8.0	-	38.0
3	8.0	73.0	-
4	-	8.0	38.0
5	-58.0	-	3.0
6	-73.0	8.0	-

The results listed in Table 6-6 indicate that the identified arm signature model significantly improves the Cartesian displacement accuracy of the Puma 560. Similar results have been obtained throughout the robot's workspace. From these results, we can further illustrate that the arm signature model is cumulatively more accurate than the design model.

If we assume that the design model accurately represents the kinematic structure of Robot 1, we can apply it to predict the displacement error generated by using the signature model for control. For instance, consider Line 1. According to the signature model, the desired Cartesian position and orientation of the end-effector at the beginning and end of the line correspond to the two joint configurations

$$\vec{q}_1 = [-48.062 \; -200.056 \; 56.300 \; 0.241 \; -36.208 \; 41.812]^T$$

and

$$\vec{q}_2 = [-102.706 \; -199.448 \; 54.660 \; 1.880 \; -35.392 \; -14.207]^T \;,$$

respectively. Using the design model, configurations $\vec{q}_1$ and $\vec{q}_2$ correspond to the two Cartesian end-effector positions (-32.047, 57.957, -56.562) and (28.519, 59.118, -56.555), respectively. The predicted displacement of the end-effector is thus 60.577 cm and the predicted displacement error is 0.577 cm. This compares with the actual measured displacement error of -0.050 cm. On the other hand, if we assume that the signature model accurately represents the kinematic structure of Robot 1, we can apply it to predict the displacement error generated by using the design model for control. According to the design model, the desired Cartesian position and orientation of the end-effector at the beginning and end of the line corresponds to the two joint configurations

$$\vec{q}_1 = [-48.106 \; -200.029 \; 56.287 \; 0.0 \; -36.258 \; 41.894]^T$$

and

$$\vec{q}_2 = [-102.385 \; -197.951 \; 50.486 \; 0.0 \; -32.534 \; -12.385]^T \;,$$

respectively. Using the signature model, configurations $\vec{q}_1$ and $\vec{q}_2$ correspond to the two Cartesian end-effector positions (-31.019, 57.813, -57.300) and (28.394, 57.065, -57.245), respectively. The predicted displacement of the end-effector is thus 59.418 cm and the predicted displacement error is -0.582 cm. This error compares with the actual measured displacement error of -0.530 cm. In this case and others, the signature model is clearly a better predictor of displacement errors than the design model. Therefore, the signature model is cumulatively more accurate than the design model.

6.6.2. Two-Dimensional Performance Evaluation

We have applied the two-dimensional grid touching task to evaluate the kinematic performance of Robots 2 through 7. Figure 6-16 illustrates the position and orientation of the grid relative to the robot's base as viewed from above the table. The X and Y axes of the grid were nominally aligned with the X and Y axes of the robot's base coordinate frame. The grid spans 60 cm in the X direction and 15 cm in the Y direction. The grid contains 62 uniformly spaced points (i.e., vertices). The desired X and Y axis spacings were both 5.0 cm.

The features used to measure the performance of a robot are line length, line straightness and grid perpendicularity. Ideally, the end-effector's position would have coincided with the intersection points of Lines $X1, X2, X3, X4$ and Lines $Y1, Y2, \ldots, Y13$, indicated in Figure 6-16. In reality, the grid lines are neither parallel, perpendicular, nor straight. We denote the actual grid lines by $X1^*, X2^*, X3^*, X4^*$ and $Y1^*, Y2^*, \ldots, Y13$. The line Xi^* is defined to be the line which passes through the actual recorded points (1,i) and (13,i). Similarly, the line Yj^* is defined to be the line which passes through the actual recorded points (j,1) and (j,4). The actual length of line Xi^* is the distance between points (1,i) and (13,i). The actual length of line Yj^* is

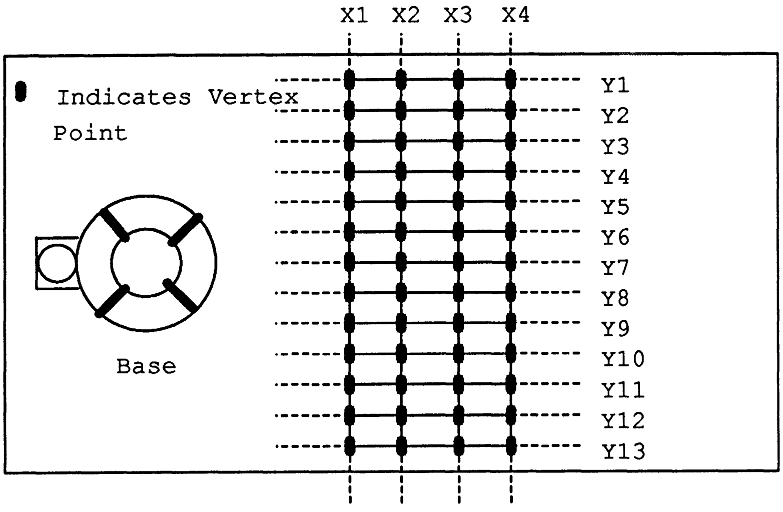


Figure 6-16: Relative Location of Two-Dimensional Grid

the distance between points (j,1) and (j,4). For each line, we also measure the normal deviation of the intermediate points. The maximum normal deviation provides a measure of the straightness (i.e., the curvature) of the grid line. The perpendicularity of the grid is determined by measuring the length of the grid diagonals (i.e., the distances between actual points (1,1) and (13, 4), and points (13,1) and (1,4)).

Tables 6-8 and 6-9 are a summary of the results obtained in the two-dimensional grid touching task. Table 6-8 pertains to the case where the design model is applied to control the position and orientation of the end-effector and Table 6-9 pertains to the case where the signature model is applied to the control the end-effector. The columns labeled *X Line Length Error* and *Y Line Length Error* represent the maximum deviation in line length

Table 6-8: Two-Dimensional Grid Touching Task Performance Summary Using the Design Model

Robot	X Line Length Error (cm)	Y Line Length Error (cm)	Maximum Normal Deviation (cm)	Diagonal 1 Length Error (cm)	Diagonal 2 Length Error (cm)
1	0.100	-0.575	0.150	-0.495	-0.320
2	0.075	-0.150	0.050	-0.170	-0.070
3	0.150	0.300	0.075	0.480	0.105
4	0.050	-0.750	0.025	-0.095	0.005
5	-0.075	0.350	0.075	-0.695	0.230
6	-0.125	0.950	0.125	0.755	0.705

Table 6-9: Two-Dimensional Grid Touching Task Performance Summary Using the Signature Model

Robot	X Line Length Error (cm)	Y Line Length Error (cm)	Maximum Normal Deviation (cm)	Diagonal 1 Length Error (cm)	Diagonal 2 Length Error (cm)
1	-0.050	-0.050	0.025	-0.020	0.005
2	0.050	-0.050	-	0.030	-0.095
3	0.075	0.075	0.025	0.105	0.030
4	0.075	-0.025	-	-0.080	-0.095
5	0.050	-0.100	-	-0.045	-0.070
6	-0.025	0.050	0.025	0.030	0.105

among the sets of lines $X1^*, X2^*, X3^*, X4^*$ and $Y1^*, Y2^*, \ldots, Y13^*$, respectively. The column labeled *Maximum Normal Deviation* represents the maximum of the set of 4 maximum normal deviations for lines $X1^*, X2^*, X3^*, X4^*$. (Since the lines $Y1^*, Y2^*, \ldots, Y13^*$ are only 15 cm long, the observed normal deviations are negligible.)

A comparison of the results in Table 6-8 with the results in Table 6-9 again indicates that use of the identified signature model to control the end-effector's Cartesian position and orientation leads to a general improvement in the kinematic performance of all 6 robots. Unfortunately, due to the limited accuracy with which these specific measurements were obtained, it is difficult to state with any accuracy by what factor the performance has increased.

6.6.3. Three-Dimensional Performance Evaluation

We have applied the three-dimensional grid touching task to evaluate the kinematic performance of Robots 3 and 5. The X and Y axes of the grid were nominally aligned with the X, Y, and Z axes of the robot's base coordinate frame. The grid spans 60 cm in the X direction, 25 cm in the Y direction, and 25 cm in the Z direction. The range in the Y and Z directions was limited by the throw of the corresponding two axes of the milling machine. The grid contains 63 uniformly spaced points (i.e., vertices). The desired X, Y, and Z axis spacings were 10.0 cm, 12.5 cm, and 12.5 cm, respectively.

The features used to measure the performance of a robot are line length, line straightness, grid line perpendicularity, radial deviation, and orientational deviation. In analogy to the two-dimensional grid, the three-dimensional grid contains 9 lines nominally parallel with the X axis, 21 lines nominally parallel with the Y axis, and 21 lines nominally parallel with the Z axis. The actual lines are defined by the corresponding pairs of points along the grid's border. In Tables 6-10 and 6-11, we summarize thirteen measured indices which describe the relative improvements in the kinematic performance of Robots 3 and 5, respectively. The last column in each of these tables lists the performance improvement factor which is the ratio of the error

Table 6-10: Three-Dimensional Grid Touching Task Performance Summary for Robot 3

Description	Design Model	Signature Model	Improvement Factor
Maximum Normal Deviation from a Line Parallel with the X Axis	0.071	0.023	3.09
Maximum Normal Deviation from a Line Parallel with the Y Axis	0.023	0.006	3.83
Maximum Normal Deviation from a Line Parallel with the Z Axis	0.019	0.007	2.71
Magnitude of Line Length Error for Line Parallel with the X Axis	0.055	0.077	0.71
Magnitude of Line Length Error for Line Parallel with the Y Axis	0.096	0.016	6.00
Magnitude of Line Length Error for Line Parallel with the Z Axis	0.027	0.013	2.08
Deviation from Perpendicular between X Axis and Y Axis Grid Lines	-1.040	-0.096	10.83
Deviation from Perpendicular between Y Axis and Z Axis Grid Lines	-0.278	0.229	1.21
Deviation from Perpendicular between Z Axis and X Axis Grid Lines	-0.505	-0.238	2.12
Relative Orientational Deviation about X Axis at Vertex 4	0.153	-0.038	4.00
Relative Orientational Deviation about Y Axis at Vertex 4	-0.157	0.038	4.14
Relative Orientational Deviation about Z Axis at Vertex 4	-0.067	-0.027	2.50

committed when the arm signature model is used for control to the error

Table 6-11: Three-Dimensional Grid Touching Task Performance Summary for Robot 5

Description	Design Model	Signature Model	Improvement Factor
Maximum Normal Deviation from a Line Parallel with the X Axis	0.066	0.023	2.87
Maximum Normal Deviation from a Line Parallel with the Y Axis	0.022	0.006	3.67
Maximum Normal Deviation from a Line Parallel with the Z Axis	0.038	0.009	3.98
Magnitude of Line Length Error for Line Parallel with the X Axis	0.139	0.182	0.76
Magnitude of Line Length Error for Line Parallel with the Y Axis	0.110	0.016	6.88
Magnitude of Line Length Error for Line Parallel with the Z Axis	0.079	0.026	3.04
Deviation from Perpendicular between X Axis and Y Axis Grid Lines	-0.974	-0.077	12.65
Deviation from Perpendicular between Y Axis and Z Axis Grid Lines	0.365	0.226	1.62
Deviation from Perpendicular between Z Axis and X Axis Grid Lines	0.271	-0.160	1.69
Relative Orientational Deviation about X Axis	0.172	0.000	-
Relative Orientational Deviation about Y Axis	-0.086	-0.005	16.0
Relative Orientational Deviation about Z Axis	-0.041	-0.014	3.01

committed when the design model is used for control. The improvement factors range between 0.71 and 16.0.

6.7. Conclusions

In this chapter, we have described our hardware/software implementation of a complete prototype arm signature identification system and have applied this system to improve the positioning accuracy of seven Puma 560 robots. This system incorporates an ultrasonic range sensor to measure the three-dimensional Cartesian positions of target sparkers placed on the robot's links. The S-Model identification algorithm uses the target position measurements to first identify a set of robot kinematic features and then combines the information contained therein to identify the robot's Denavit-Hartenberg kinematic parameters. This algorithm leads to an efficient method for collecting measurements. Using our prototype system, thousands of measurements can be obtained in less than one hour. The use of our prototype system to increase the performance of actual robots demonstrates that the S-Model identification algorithm is both feasible and practical.

A key issue in the design of an arm signature identification system is the selection and/or design of a sensor system. Our sensor system incorporates a commercially available ultrasonic range sensor to measure the three-dimensional Cartesian positions of target sparkers placed on the robot's links. A unique sensor array design and specialized slant range compensation algorithms are used to obtain ultrasonic range measurements which are significantly more accurate (by a factor of 10-50) than accuracies which can be obtained using the commercially available system itself.

For arm signature identification, the advantages of using our ultrasonic sensor system design are:

- The sensor array can be easily calibrated.

- The fixtures required to attach the target sparkers to the robot are extremely simple to manufacture and there placement with respect to the links is essentially arbitrary. This avoids the need for elaborate and costly fixturing.
- The sensor system can be easily interfaced to a host computer (typically the robot controller) which can coordinate the digitizing operation while simultaneously controlling the robots motions.
- The ultrasonic emission characteristics of the sparkers are such that they can be modeled as infinitely small point sources. The sparkers are thus ideally suited for implementing the target points used in the S-Model identification algorithm.
- The sparkers and the microphones remain stationary during the measurement process. This is, for instance, in contrast to optical devices which would customarily require an elaborate positioning system to enable the detector to track and locate the position of a target. Avoiding the tracking problem leads to a simple, efficient, and relatively inexpensive sensor system.
- The sensor system has a large dynamic range with high resolution. The active volume is a cube approximately 1.5 meters in length and the Cartesian positioning accuracy is on the order of $\pm$ 0.02 cm.
- The speed at which range measurements can be acquired and the Cartesian positions of the target can be determined enables high sampling rates (on the order of 4 measurements/sec) to be achieved.
- The configuration of the sensor array and the two calibration rods are fixed and independent of the target sparker. Thus, the measurements for identifying the arm signatures of a variety of robots can be obtained without reconfiguring or recalibrating the sensor array.
- Remote range sensing eliminates the need to position detectors within close proximity to the target and avoids the problems associated with contact sensing.
- The ability to increase the target radii provides a mechanism for increasing arm signature accuracy.

- The operation of the sensor system can be completely automated thus eliminating the need for a highly skilled operator.

We have devised three methods for evaluating manipulator kinematic performance in which the Puma is programmed to position its end-effector at the vertices of a one-, two-, or three-dimensional grid. At each of the grid vertices, the actual relative position of the end-effector is measured.

In the one-dimensional grid touching task, the relative position of the end-effector is measured with a cathetometer. The tip of a pin, attached to the end-effector is sighted through the telescope of the cathetometer. The task is performed twice, once using signature based control and once using the design kinematic model for control. The relative improvement in performance is then determined. With signature-based control, the end-effector positioning accuracy of one Puma 560 robot was increased by as much as a factor of 10.0. The lowest increase in performance for this same robot was a factor 2.8.

In the two-dimensional grid touching task, the workspace coordinate frame is aligned with the surface of a table and a custom-designed, spring-loaded pen is attached to the robot's end-effector. The positions of the end-effector are recorded when the pen touches the mylar surface. A machinist's rule is applied to measure the relative positional deviations in the grid points. Features of interest are the perpendicularity of the lines defined by the vertices, line straightness, and line length. The measured performance of six Puma 560 robots using design model based control and signature-based control are listed in Tables 6-8 and 6-9, respectively. These tables describe the general improvement in the performance of all six robots.

In the three-dimensional grid touching task, a jig is mounted on the table of a three-axis milling machine. As the robot approaches a desired target point, the precision test cube attached to its end-effector deflects the nine dial gauges within the jig. The relative Cartesian position and orientation of the end-effector is computed based upon the measured deflections and the known geometry of the gauge array. The three instrumented axes of the milling machine are used to accurately position the jig in three dimensions. The robot is programmed to position its end-effector at 63 vertex points within a rectangular grid coinciding with the position of the jig. This technique has been applied to measure and evaluate the performance of two Puma 560 robots, the results of which are listed in Tables 6-10 and 6-11. As in the previous two methods, the performance of these robots was significantly increased by using the identified kinematic parameters for control. The performance improvement factors listed in Tables 6-10 and 6-11 range between 1.21 and 16.0, with but two exceptions.

The improvements are not as large as one might expect based upon the feature estimate accuracy predictions listed in Table 6-5. The disparity is believed to be caused by the presence of small biases in the individual range measurements, and this appears to be the limiting factor in the performance of our prototype system. In order to obtain additional improvements in manipulator kinematic performance, alternative sensor systems may be necessary.

Chapter 7
Performance Evaluation Based Upon Simulation

7.1. Overview

In this chapter, we consider the statistical performance of robot position control methods, and discuss the origin of robot positioning errors. In conventional *design-model robot control,* the robot design model is used as a basis for kinematic control. In this case, manufacturing errors contribute most to robot positioning errors. In *arm signature-based robot control* (S-Model), the correct arm signature model eliminates kinematic errors due to manufacturing. In this case, robot performance is limited by sensor errors which contribute to inaccuracy of the identified arm signature model.

Experimental studies of D-Model and S-Model control were described in the previous chapter. In those studies, the S-Model control showed consistent performance improvement, and demonstrated accuracy approaching limits predicted by the joint encoder resolution. In this chapter, we analyze the origin of this improvement, and quantify the requirements of an arm signature identification system in terms of the underlying sensor performance.

D-Model control performance depends upon the manufacturing process which determines the actual geometry of the robot. Since manufacturing error vary randomly from one robot to another, the performance of an arbitrarily chosen robot has a random component. The statistical parameters which characterize a robot's positioning accuracy are related to the statistical parameters which characterize the manufacturing error probability distribution functions. Unfortunately, the complexity of robot kinematics prevents us from analytically deriving this relationship.

In this chapter, we apply Monte-Carlo simulation techniques to gain insight into the relationship between manufacturing error and the performance of a design model controller. A kinematic model of a Puma 560 robot which directly incorporates manufacturing errors as parameters has been developed for our analysis and is described in Section 7.2.2. The results of our analysis are presented in Section 7.4.1 and 7.4.2, and bring to light some of the major disadvantages of the design model control approach.

The performance of S-Model control is dictated by the presence of errors in the arm signature identification process and not by the presence of manufacturing errors. Errors are introduced into the identification process by the sensor system and must be analyzed to determine if the performance of the robot is improved relative to the performance obtained by Controller D. In this chapter, we analyze the relationship between target measurement errors and Controller S performance using simulation methods.

This chapter is organized as follows. In Section 7.2, we describe the structure of our Monte-Carlo simulator and test its validity in Section 7.3. Then, in Sections 7.4.1 and 7.4.2, we explore the relationship between the performance of the D-Model controller and the two types of manufacturing errors referred to as *encoder calibration errors* and *machining and assembly errors*. Sections 7.4.3 - 7.4.5 discuss the relationships between the performance of the S-Model controller and the target measurement accuracy, the number of measurements, and the length of the target radii, respectively.

Finally, in Section 7.5, the issue as to whether or not S-Model control improves robot end-effector positioning accuracy is resolved.

7.2. A Monte-Carlo Simulator

In this section, we describe a Monte-Carlo simulation of a robot under either D-control or S-control and analyze the effects of the various error sources on resulting performance.

The performance evaluation of these two controllers is complicated by the nonlinear robot kinematics, the nonlinearities of the S-Model identification algorithm, and the presence of error sources. Random errors are introduced into both the manufacturing and identification processes. This reality forces us to develop a Monte-Carlo simulator to conduct a statistical evaluation. Because of the variability of robot designs, development of a general-purpose Monte-Carlo simulator was impractical. Thus, to complement our hardware experimentation, we have evaluated the kinematic performance of the Puma 560 robot with both design model based control and signature-based control. In doing so, we have established a methodology to evaluate the kinematic performance of all robots.

7.2.1. Evaluating Kinematic Performance

The three-dimensional grid touching task is used in simulation to evaluate the kinematic performance of the Puma 560 with alternative controller designs. The grid contains twelve vertices, labeled 1 through 12, whose approximate positions, relative to the robot, are indicated in Figure 7-1. The desired positions of the vertices, measured with respect to the base coordinates of an ideal Puma 560 are listed in Table 7-1. The desired orientation of the end-effector is the same for all twelve vertices. The orientational

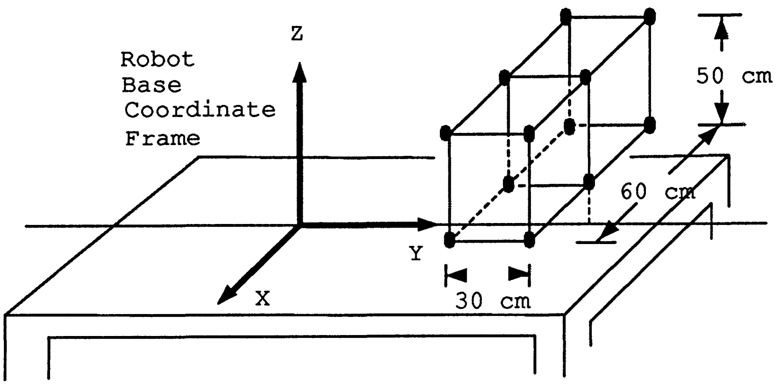


Figure 7-1: Definition and Location of Points in the Simulated Three-Dimensional Grid Touching Task

relationship between the robot's base coordinate frame and the end-effector's coordinate frame is specified by the transformation $Rot(y, 180.0)Rot(z, 90.0)$. Vertices 1 through 8 define four lines parallel to the X axis (Lines 1, 4, 5, and 8) each 60.0 cm long, four lines parallel to the Y axis (Lines 2, 3, 6, and 7) each 30.0 cm long, and four lines parallel to the Z axis (Lines 9, 10, 11, and 12) each 50.0 cm long. The four vertices 9 through 12 are the midpoints of lines 1, 4, 5, and 8, respectively.

Six types of positioning and orienting errors are used to measure and evaluate robot performance.

1. *Linear Displacement Errors* -- These errors are defined as the difference between the actual and desired lengths of various grid lines. Sixteen linear displacement errors are considered, twelve

Table 7-1: Positions of Three-Dimensional Grid Vertices

Point	Cartesian Position		
	X (cm)	Y (cm)	Z (cm)
1	-30.0	40.0	30.0
2	30.0	40.0	30.0
3	-30.0	70.0	30.0
4	30.0	70.0	30.0
5	-30.0	40.0	80.0
6	30.0	40.0	80.0
7	-30.0	70.0	80.0
8	30.0	70.0	80.0
9	0.0	40.0	30.0
10	0.0	70.0	30.0
11	0.0	40.0	80.0
12	0.0	70.0	80.0

corresponding to the errors between the actual and desired lengths of Lines 1 through 12 and the four corresponding to the errors between the actual and desired lengths of the diagonal lines defined by Vertices, 1 and 4, 2 and 3, 5 and 8, and 6 and 7.

2. *Normal Deviation Errors* -- These errors are defined as the normal distance between the end-effector's position at one of the mid-point vertices and the line joining the two actual positions of the end-effector at the lines' endpoints. These errors represent a measure of line straightness. For each of the Lines, 1, 4, 5, and 8, a *normal deviation error* is computed.

3. *Radial Displacement Errors* -- These errors are defined as the radial distances between the actual and desired positions of the end-effector. Errors at Vertices 1, 4, 5, and 8 are computed.

4. *Absolute Orientational Errors* -- These errors are defined as the angular deviation between the actual and desired orientations of the end-effector. Absolute orientational errors are computed at Vertices 1, 4, 5, 8.

5. *Relative Orientational Errors* -- Each of these errors are defined as the angular deviation between the actual orientation of the end-

effector at a specified vertex and the orientation of the end-effector at the grid origin (i.e., Vertex 1).

6. *Line Perpendicularity Errors* -- The *X-Y Line Perpendicularity Error* is the angular deviation between Line 1 and Line 3, minus 90 degrees. Similarly, the angular deviations between Lines 3 and 8 and Lines 8 and 1 are the *Y-Z* and *Z-X Line Perpendicularity Errors*, respectively.

A total of thirty-four positioning and orienting errors are computed during each experiment (i.e., simulated event) within a simulation run in order to measure the robot's kinematic performance. Each error represents an index of performance and each simulation consists of 500 experiments. Following the completion of each simulation, we compute the mean, variance, and standard deviation of each performance index. Different sets of input parameters are specified in each simulation. To analyze the effects of manufacturing errors upon the performance of the D-Model controller, the input parameters are the manufacturing error variances while to analyze the performance of the S-Model controller, the input parameters are the nominal target radii $\mathbf{R}_i$, number of points per circle $\mathbf{N}_i$, and etc.

7.2.2. Design Model Control

The performance of the D-Model controller is a direct reflection of the manufacturing errors. To simulate the performance, we have developed a kinematic simulator model of the Puma 560. The model

$$\mathbf{T}_6^* = \mathbf{K}_0 \cdot \mathbf{K}_1 \cdot \mathbf{K}_2 \cdot \mathbf{K}_3 \cdot \mathbf{K}_4 \cdot \mathbf{K}_5 \cdot \mathbf{K}_6 \tag{7.1}$$

where

$$\begin{aligned}\mathbf{K}_0 = {} & \text{Trans}(\varepsilon_1, \varepsilon_2, \varepsilon_3)\,\text{Rot}(x, \varepsilon_4)\,\text{Rot}(y, \varepsilon_5)\,\text{Trans}(0.0, 0.0, x_1) \\ & \text{Rot}(x, \varepsilon_6)\,\text{Rot}(y, \varepsilon_7)\,\text{Rot}(z, \varepsilon_8)\end{aligned} \tag{7.2}$$

$$\begin{aligned}\mathbf{K}_1 = {} & \mathrm{Rot}(z,\theta_1)\,\mathrm{Trans}(0.0,0.0,\varepsilon_9)\,\mathrm{Rot}(x,\varepsilon_{10})\,\mathrm{Rot}(y,\varepsilon_{11}) \\ & \mathrm{Trans}(\varepsilon_{12},0.0,x_2)\,\mathrm{Rot}(x,\varepsilon_{13}+x_3)\,\mathrm{Trans}(0.0,0.0,x_4) \\ & \mathrm{Rot}(z,\varepsilon_{14})\end{aligned} \tag{7.3}$$

$$\begin{aligned}\mathbf{K}_2 = {} & \mathrm{Rot}(z,\theta_2)\,\mathrm{Trans}(0.0,0.0,\varepsilon_{15}+x_5)\,\mathrm{Rot}(x,\varepsilon_{16}) \\ & \mathrm{Rot}(y,\varepsilon_{17})\,\mathrm{Trans}(\varepsilon_{18}+x_6,0.0,0.0)\,\mathrm{Rot}(x,\varepsilon_{19}) \\ & \mathrm{Rot}(y,\varepsilon_{20})\,\mathrm{Rot}(z,\varepsilon_{21})\end{aligned} \tag{7.4}$$

$$\begin{aligned}\mathbf{K}_3 = {} & \mathrm{Rot}(z,\theta_3)\,\mathrm{Trans}(0.0,0.0,\varepsilon_{22})\,\mathrm{Rot}(x,\varepsilon_{23})\,\mathrm{Rot}(y,\varepsilon_{24}) \\ & \mathrm{Trans}(x_7+\varepsilon_{25},x_8+\varepsilon_{26},x_9+\varepsilon_{27})\,\mathrm{Rot}(x,x_{10}+\varepsilon_{28}) \\ & \mathrm{Rot}(y,\varepsilon_{29})\,\mathrm{Rot}(z,\varepsilon_{30})\end{aligned} \tag{7.5}$$

$$\begin{aligned}\mathbf{K}_4 = {} & \mathrm{Rot}(z,\theta_4)\,\mathrm{Trans}(\varepsilon_{31},0.0,x_{11}+\varepsilon_{32})\,\mathrm{Rot}(x,x_{12}+\varepsilon_{33}) \\ & \mathrm{Rot}(z,\varepsilon_{34})\end{aligned} \tag{7.6}$$

$$\begin{aligned}\mathbf{K}_5 = {} & \mathrm{Rot}(z,\theta_5)\,\mathrm{Trans}(0.0,0.0,\varepsilon_{35})\,\mathrm{Rot}(x,x_{13}+\varepsilon_{36}) \\ & \mathrm{Trans}(\varepsilon_{37},0.0,x_{14})\,\mathrm{Rot}(z,\varepsilon_{38})\end{aligned} \tag{7.7}$$

$$\mathbf{K}_6 = \mathrm{Rot}(z,\theta_6)\,\mathrm{Trans}(0.0,0.0,x_{15}+\varepsilon_{39}) \tag{7.8}$$

incorporates both the nominal mechanical design specifications and the manufacturing errors as parameters. The homogeneous transformation matrix $\mathbf{T}_6^*$ defines the position and orientation of a coordinate frame fixed relative to the end-effector with respect to a coordinate frame fixed relative to the base. The parameters, $x_1, x_2, \ldots, x_{15}$, represent the nominal mechanical design specifications while the parameters, $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{39}$, represent the manufacturing errors. The angular positions of joints 1 through 6, as measured by the respective joint encoders, are denoted by θ_1 through θ_6, respectively. The transformations comprising the matrix $\mathbf{K}_0$ describe the actual physical structure of the base link while the matrices $\mathbf{K}_1, \mathbf{K}_2, \ldots, \mathbf{K}_6$ describe the actual physical structure of Links 1, 2, ..., 6, respectively, for a particular robot. A derivation of the model (7.1) is presented in Appendix F.1.

The values of the nominal design parameters for the Puma 560, x_1 - x_{15}, are listed in Table 7-2.

Table 7-2: Nominal Design Kinematic Parameters for a Puma 560

Parameter	Value	
	(cm)	(deg)
x_1	58.500	-
x_2	8.900	-
x_3	-	-90.0
x_4	17.750	-
x_5	2.500	-
x_6	43.180	-
x_7	-2.032	-
x_8	-33.320	-
x_9	-5.341	-
x_{10}	-	90.0
x_{11}	10.050	-
x_{12}	-	-90.0
x_{13}	-	90.0
x_{14}	4.445	-
x_{15}	1.270	-

When the manufacturing errors are zero, the model (7.1) is nearly equivalent to the Denavit-Hartenberg model (2.8) where the ideal Denavit-Hartenberg parameters are listed in Table 6-2. The difference is caused by the fact that, in (7.1), we consider the base frame to be located at the lower end of Link 0 (i.e., coincident with the surface to which the robot would normally be mounted) as indicated in Figure 7-2. To compare (7.1) with (2.8) the ideal Denavit-Hartenberg parameter $\mathbf{d}_1$ is 67.4 cm - - the desired distance between the bottom surface of the base and the Joint 2 axis.

The values listed in Table 7-2 where obtained from a cursory examination of the mechanical structure of a Puma 560 and not from an exhaustive study of the actual manufacturing process. Such a study is beyond the scope

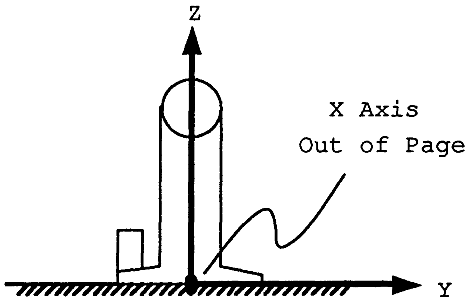


Figure 7-2: Location of Physical Base Coordinate Frame of the Puma 560 as Defined by the Kinematic Simulator Model without Manufacturing Errors

of this dissertation. It is believed that the model (7.1) is sufficiently complex and realistic enough to provide insight into the expected performance of a Puma 560 whose kinematics are affected by random manufacturing errors.

In simulation, the manufacturing errors are generated using standard random number generators. Gaussian distributions are used to model the variations in manufacturing errors from one robot to another. It is assumed that all manufacturing errors have zero mean.

The manufacturing errors $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_{39}$ are divided into three categories; *positional errors, orientational errors,* and *encoder calibration errors.* For convenience, it is assumed that errors within a category have the same variance. The variance of the errors in the *positional error* category is measured in cm^2 and is denoted by σ_p^2. The variances of the errors in the remaining two categories are measured in deg^2 and are denoted by σ_o^2 and σ_e^2, respectively. Our experience with Puma 560 robots suggests that the orientational errors introduced during the calibration of the joint encoders are significantly greater than the orientational errors committed during machining and assembly thus the need for two separate orientational error variances.

7.2.3. Signature-Based Control

Simulating the performance of the S-Model controller is complicated by the need to simulate the identification process. Arm signature identification in combination with signature-based control proceeds through a sequence of five steps. Table 7-3 lists the inputs, the outputs, and the potential sources of errors for each of these steps. The table indicates how the input errors propagate through the identification algorithm to produce end-effector positioning and orienting errors. The outputs of a step are affected by both the errors of that step and the errors of previous steps. Previous errors are thus input errors to the current step. The error sources listed in Steps 1 and 2 are specific to our sensor system. In practice, by interchanging simulator modules, alternative sensor systems can be analyzed.

Since it would be impractical to simulate all of the errors listed in Table 7-3 and interpret the results, our primary objective has been to evaluate on the effect of sensor errors (i.e., the errors which affect the accuracy of the target measurements) on the performance of the S-Model Controller. In our simulation of the Puma 560, we have considered only the sensor errors which most strongly affect the measurement of the target loci. These include slant range errors and sensor system calibration errors.

The S-Model identification algorithm is applied, in simulation, to identify the kinematic parameters of a perfectly manufactured Puma 560 robot in the presense of slant range errors and sensor system calibration errors. The assumption that the actual robot has no manufacturing errors merely

Table 7-3: Propagation of Identification and Control Errors

Step	Description	Inputs	Outputs	Sources of Error
1	Sensor Model	Actual Target Range	Filtered Target Range	Temperature, Acoustic Noise, Numerical
2	Sensor System Model	Filtered Target Ranges	Cartesian Target Position	Sensor Misalignment, Numerical
3	Feature Model	Cartesian Target Positions	Feature Parameter Vector	Joint Wobble, Link Compliance, Gear Backlash, Transmission Compliance
4	S-Model	Feature Vectors	S-Model Parameters	Numerical
5	D-H Model	S-Model Parameters	Denavit-Hartenberg Model Parameters	Numerical
6	Control Model	Denavit-Hartenberg Model Parameters Desired End-Effector Position	Actual End-Effector Position	Steady-State Joint Position Control Errors, Encoder Resolution, Link Compliance, Backlash Numerical

simplified the simulator design task. Since the S-Model identification algorithm is a general method and does not require a priori knowledge as to the nominal kinematic structure of the robot, the statistical performance of a S-Model Controller for a perfectly manufactured Puma 560 will be identical to the performance of an S-Model Controller for a Puma 560 with an arbitrary set of manufacturing errors. When the slant range errors are zero and the sensor system calibration errors are zero, the S-Model identification algorithm identifies the true kinematic parameters and the S-Model Controller provides infinite positioning accuracy (i.e., the positional and orientational deviation between the desired and actual location of the end-effector is zero for all points within the workspace of the Puma 560).

In order to initiate the simulator, a variety of input parameters must be specified such as the true locations of the target points with respect to the links, and the spatial transformation between the sensor coordinate frame and the robot base coordinate frame. A list of the input parameters and the values used in our simulation experiments are included in Appendix F.2. These values coincide with our hardware implementation of the S-Model identification algorithm and our prototype system.

We highlight the sequence of computational tasks performed by the simulator. First, the Puma 560 design model is used to compute the actual locations of the target points with respect to the sensor coordinate frame. Then, the actual slant ranges are computed based upon the known positions of the microphones. Next, zero mean gaussian noise is added to each of the slant ranges. The standard deviation of this noise depends upon the range. The linear approximation

$$\sigma = 9.6432e{-}5X + 0.003 \tag{7.9}$$

models the relationship between the noise standard deviation, σ, and the range, X. In (7.9), σ and X are measured in centimeters. The corrupted slant ranges are also quantized to two decimal places to simulate the limited resolution of the GP-8-3D. Using these corrupted slant ranges, the S-Model identification algorithm is applied to determine the robot's arm signature. This procedure is repeated 500 times producing 500 arm signatures. The Newton-Raphson algorithm is then applied to control the actual Puma 560 robot in performing the grid touching task described in Section 7.2.1 based upon each of the identified signatures. Finally, the means, variances, and standard deviations of the thirty four performance indices are computed and tabulated for later analysis.

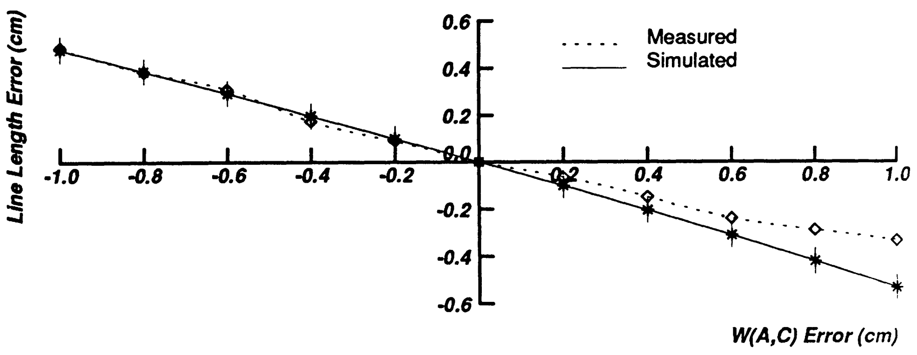
7.3. Simulator Verification

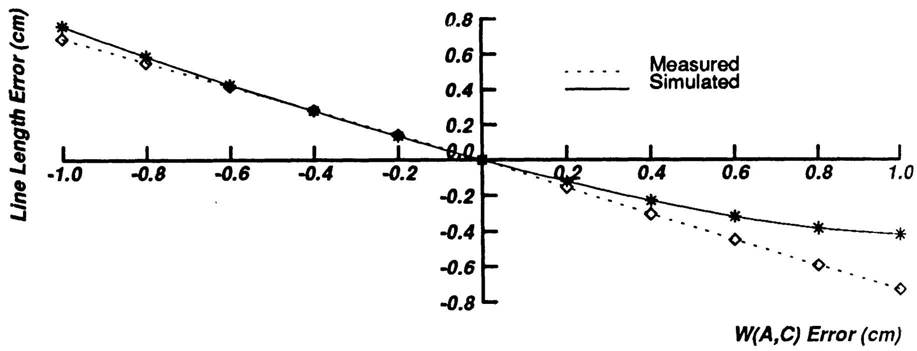
The goal in verifying the simulator is to demonstrate that is produces realistic and reliable data. Our simulator id divided into two independent parts, one which deals with the performance of a D-Model Controller and one which deals with the performance of an S-Model Controller. The verification of each these parts can be performed independently.

In verifying the former part we would like to demonstrate that the statistical performance of a D-Model Controller predicted by the simulator is consistent with the statistical performance of actual Puma 560 robots which utilize design model based control. To accomplish this task, we would normally want to vary one or more of the input parameters to both the physical system and the simulator, monitor the outputs of both the physical system and the simulator, and then compare the two. Unfortunately in our case, there are no parameters of the real system which can be varied so as to effect the performance of the robot. Manufacturing errors are physically embedded and fixed within the physical structure of each robot. Furthermore, we cannot realistically compare the statistical performance of only seven robots with that of the simulator since a sample size of seven can hardly be considered statistically meaningful. Although the observed performance of the seven Puma 560 robots is consistent with the performance distributions predicted by the simulator additional studies would be required to rigorously verify the manufacturing error model (7.1).

Verification of the sensor error model and simulated signature identification process is more easily accomplished. Experiments were conducted in which the calibrated distance, $W_{A,C}$, between Microphones A and C (refer to Appendix E) is intentionally misaligned. The alignment errors ranged from -1.0 to 1.0 cm. While alignment errors of this magnitude are unlikely in practice, their use ensures that the measured variations in the performance of a robot is statistically significant in comparison to the variability due to other random errors. In the laboratory, the alignment errors are produced by modifying the calibration data used by the triangulation algorithms. For each of eleven different alignment errors we identify the arm signature of one of the robot's in the laboratory and apply the signatures to control the position and orientation of the end-effector. The one-dimensional grid touching task is performed in several times and a cathetometer is used to measure the performance. The analogous experiments are conducted in simulation.

The simulated and experimentally measured performance of the Puma 560 is plotted as a function of the error in $W_{A,C}$ in Figure 7-3. In Figure 7-3, performance is measured in terms of line length error. Line length errors for three lines are shown: one parallel to the base X axis which is 60.0 cm long, one parallel to the Y axis which is 40.0 cm long, and one parallel to the Z axis which is 50.0 cm long. The simulated performance curves represent the mean performance and the vertical bars (i.e., confidence intervals) delineate the 3 standard deviation confidence bands [29]. These results confirm that the simulator produces realistic and reliable data. The experimentally measured performance curves lie within the confidence intervals of the simulated performance curves over most of the range in $W_{A,C}$. It is only for extreme errors in $W_{A,C}$ that the correspondence between the simulator and the actual performance differ significantly. In practice, such extreme alignment errors would not occur.





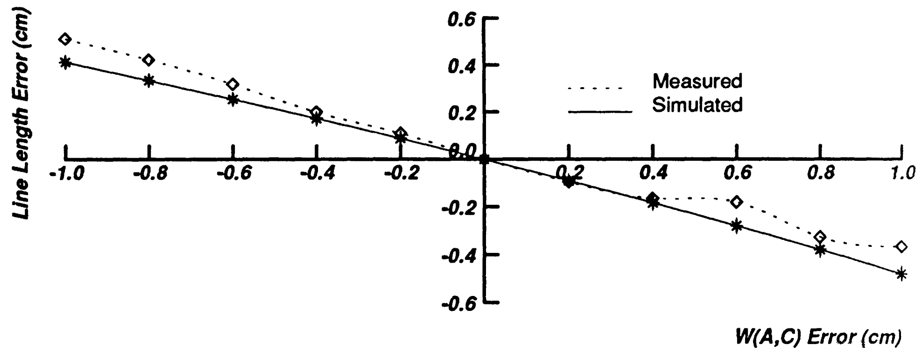


Figure 7-3: Simulated and Experimental Performance as a Function of Errors in $W_{A,C}$ (a) X Axis Line (b) Y Axis Line (c) Z Axis Line

7.4. Results

Our analysis of the statistical effect of manufacturing errors and identification errors upon end-effector positioning accuracy is divided into five sections. In Section 7.4.1, we consider the effects of encoder calibration errors upon the performance of the D-Model Controller. The effects of machining and assembly errors upon the performance of the D-Model Controller are discussed in Section 7.4.2. Section 7.4.3 discusses the relationship between the performance of the S-Model Controller and the sensor system accuracy. The relationship between the number of measurements and the performance of the S-Model Controller is described in Section 7.4.4. Section 7.4.5 describes the relationship between the target radii and the performance of the S-Model Controller.

7.4.1. Encoder Calibration Errors

The procedure for calibrating the joint encoders of a Puma 560 can often lead to significant encoder calibration errors. These errors result in fixed biases between the actual and measured positions of the joints. Five simulations were conducted using model (7.1) to evaluate the effect of random encoder calibration errors upon robot performance. In these simulations, the positional error standard deviation and orientational error standard deviation were $\sigma_o = .01$ *cm* and $\sigma_p = .01$ *deg*, respectively. The standard deviation of the encoder calibration error σ_e was varied from 0.1 deg to 0.5 deg. The statistical variations in the performance indices of 500 simulated robots are plotted as a function of σ_e in Appendix G.

The standard deviations of all 34 performance indices are seen to increase linearly as the standard deviation of the encoder calibration errors increase. To illustrate, the radial position errors at Vertices 1, 4, 5, and 8 are

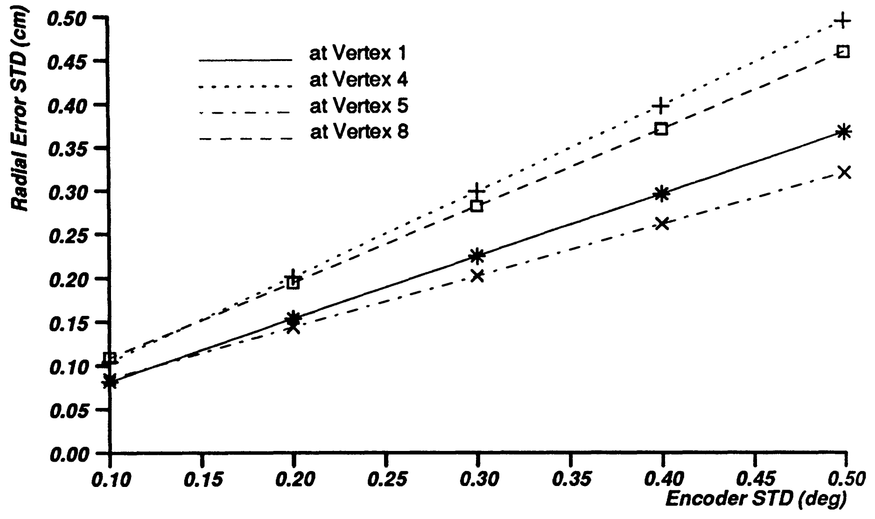


Figure 7-4: Radial Position Error Standard Deviations as a function of σ_e

plotted as a function of σ_e in Figure 7-4. From Figure 7-4, it is also observed that the rate at which different index standard deviations increase varies. For instance, the standard deviation of the Vertex 1 radial position error increases 1.4 times faster than that at Vertex 4. It is believed that the variability in the rate at which the the performance index standard deviations increase as the encoder calibration error standard deviation increases is partly due to the variability in robot performance over the workspace. Robots with revolute joints, particularly those with revolute joints in the positioning system (i.e., the first three joints), tend to amplify joint positioning errors. Roughly speaking, the amplification factors are the effective radii between the axes where the errors occur and the end-effector. Thus, as a robot extends its reach the various amplification factors tend to increase. In design model control, encoder calibration errors translate into joint positioning errors. Thus, we would expect robot kinematic performance to decrease as the desired end-effector position approaches the outer limits of the robot's workspace. Figure 7-4 and the remaining plots in Appendix G clearly support this hypothesis. The standard deviation of the radial error at Vertices 4 and 8 are .299 cm and .282 cm, respectively, while the standard deviation of the radial error at Vertices 1 and 5 are .225 cm and .202 cm, respectively -- when $\sigma_e^2 = 0.3deg$. From Table 7-1 we see that Vertices 4 and 8 lie near the maximum reach of the Puma 560 while Vertices 1 and 5 lie relatively close to the base of the robot.

There is another important feature of the effect of manufacturing errors upon performance which is not apparent from the plots in Appendix G. Even though the simulated manufacturing errors have zero mean, the means of the performance indices are nonzero. In fact for certain performance indices, the indices' mean values may be larger than their standard deviations. Furthermore in some cases, the index means vary linearly with the encoder calibration error standard deviation. Examples of this are illustrated in Figure G-2.

The mean value of the Line 11 line length error decreases from 0.0268 cm to 0.0038 cm as σ_e increases from 0.1 deg to 0.5 deg. Other index mean values may even change sign. Robot kinematics, especially those of the Puma 560, are highly nonlinear and coupled. As is often the case for nonlinear systems, the expected value of the system output, in this case robot performance, is both a function of the mean and standard deviation of the input noise. These results are in direct contradiction to the analytic results derived by Wu [32]. Our results indicate that in the presense of zero-mean gaussian random manufacturing errors the expected positioning accuracy of an arbitrary robot will be nonzero. While a reduction in manufacturing errors will tend to increase robot positioning accuracy, this approach by itself will not necessarily eliminate the expected positioning errors inherent to robots which utilize a design model control strategy.

7.4.2. Machining and Assembly Errors

To evaluate the effect of positional and orientational machining and assembly errors upon robot performance, five more simulations were performed. In these simulations, the standard deviations of the positional and orientational errors, σ_p and σ_o were varied simultaneously from 0.01 cm and 0.01 deg to 0.05 cm and 0.05 deg, respectively, while the encoder calibration error standard deviation, σ_ε , was fixed to 0.1 deg. The standard deviations of the 34 performance indices are plotted in Appendix G as a function of σ_p and σ_o. These sample standard deviations are based upon the performance of 500 robots.

The variations in the performance index standard deviations in these simulations are quite different from those in Section 7.4.1. For instance, consider Figure 7-5 where the radial position error standard deviations at Vertices 1, 4, 5, and 8 are plotted as a function of σ_p and σ_o. In contrast to Figure 7-4, the relationships shown in Figure 7-5 are distinctly nonlinear. Similar nonlinear relationships are exhibited by the remaining performance index standard deviations in Figure G-3. Except for the Vertex 1 and Vertex 4 absolute orientational error standard deviation, the performance index standard deviations all increase monotonically as σ_p and σ_o increase. Furthermore, the slopes of all 34 curves increase monotonically.

In further contrast to the curves in Figure 7-4, the curves in Figure 7-5 do not all diverge from one another. For example, as σ_p and σ_o increase the difference between the variability in the radial positioning error at Vertex 1 and at Vertex 5 decreases from 0.0224 to 0.0138. In some cases the curves may even intersect as shown in Appendix G.

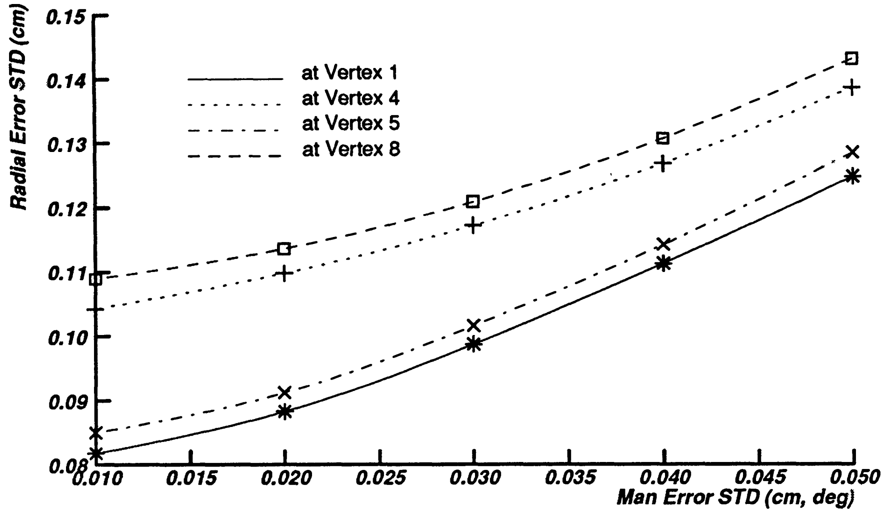


Figure 7-5: Radial Position Error Standard Deviation as a Function of σ_p and σ_o.

The variations in the performance index means as a function of σ_p and σ_o are much more predictable than when as a function of σ_e. As σ_p and σ_o increase, all of the performance index means increase monotonically. Plots of the performance index means as a function of σ_p and σ_o are presented in Figure G-4. For Controller D, the presence of positional and orientational manufacturing errors leads to an average decrease in end-effector positioning accuracy.

The effect of positional and orientational manufacturing errors on the variability of end-effector positioning accuracy is more complex than that for encoder calibration errors as evidenced by the nonlinear relationships in Figure 7-5. However, the effect of these manufacturing errors upon the expected end-effector positioning accuracy of a robot is much more simple than for the effect due to encoder calibration errors. Finally, the sensitivity of the performance index standard deviations to simultaneous variations in σ_p and σ_o is less than the sensitivity of the performance index standard deviations to variations in σ_e.

7.4.3. Sensor Measurement Errors

Equations (6.12) and (6.13) describe the relationship between feature estimate accuracy and the sensor system accuracy, σ, as well as the identification parameters, N_i and R_i. Unfortunately, the effect of feature estimate accuracy upon the performance of an S-Model Controller cannot be derived analytically. Therefore, the simulator described in Section 7.2.3 was applied to empirically determine the qualitative relationship between measurement noise standard deviation, σ, and the expected performance of a Puma 560 robot which uses signature-based control. Six separate simulations were conducted each using a different values of σ in (7.9), to identify the robot's arm signature. In each simulation, the value of σ was computed according to (7.9) and then multiplied by a factor k ranging from 1.0 to 10.0.

The standard deviations of the 34 performance indices as function of k are plotted in Appendix G. The plot of the Radial Position Error standard deviation is reproduced below in Figure 7-6. The standard deviations of all the performance indices increase linearly with the noise measurement factor. The more inaccurate the sensor system is the more inaccurate the identified signatures are and hence the greater the variability in the performance of the S-Model Controller. The significance of these results is that the sensitivity of the performance of the S-Model Controller to variations in sensor system accuracy is consistent with the relationships (6.12) and (6.13) which govern the feature estimate accuracy. The orientational accuracy of the estimated

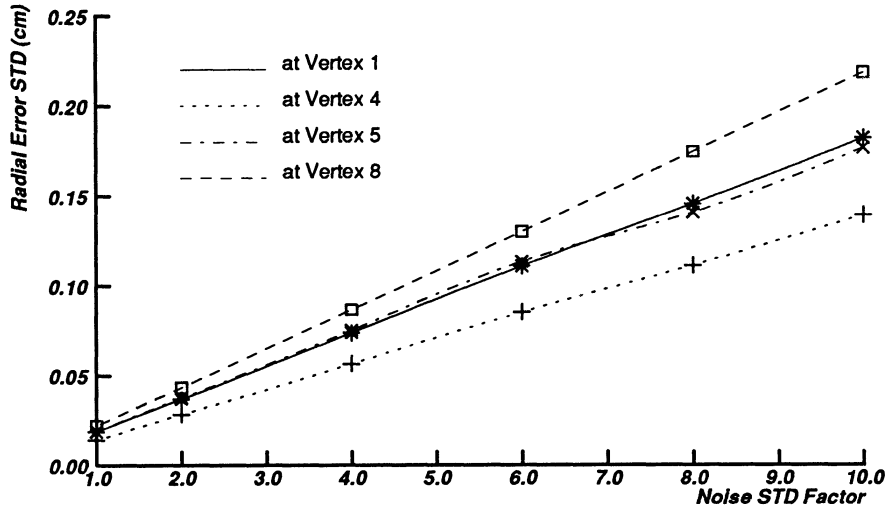


Figure 7-6: Radial Position Error Standard Deviation as a Function of the Measurement Noise Factor *k*

planes-of-rotation (6.12) and the positional accuracy of the estimated centers-of-rotation (6.13) are both directly proportional to the measurement noise standard deviation. Feature estimate accuracy has a direct impact upon the performance of an S-Model Controller.

Nineteen of the Performance indices are defined such that they can take on any real value (e.g., the various Line Length Error indices). Using S-Model control the mean values of these nineteen indices are all zero which is in strong contrast to the effect which manufacturing errors have upon the performance of the D-Model Controller. The significance of this is that on the average we can expect the S-Model identification algorithm to correctly identify the true kinematic parameters of a robot and thus eliminate end-effector positioning errors. For the remaining 15 performance indices, the index's mean values are directly proportional to the measurement noise standard deviation, σ.

7.4.4. Number of Measurements

Equations (6.12) and (6.13) describe the relationship between feature estimate accuracy and the S-Model identification parameter N_i . Unfortunately, the effect of feature estimate accuracy upon the performance of Controller S cannot be derived analytically. Therefore, the simulator described in Section 7.2 was applied to empirically determine the qualitative relationship between N_i and the expected performance of a Puma 560 robot which uses Controller S. Five separate simulations were conducted each using a different number of measurements per joint, N_i , to identify the robot's arm signature.

The standard deviations of the 34 performance indices for $N_i = 20, 40, 60, 80$, and 100, are plotted in Appendix G. For comparison purposes, the plot of the Radial Position Error standard deviation is reproduced below in Figure 7-7. In general, the standard deviations of all 34 performance indices are inversely proportional to the square root of N_i . By increasing the number of measurements used in identifying a robot's arm signature, substantial increases in end-effector positioning accuracy can be achieved. For instance, in Figure 7-7 the Radial positioning errors are reduced by a factor of 2.34. Such findings are consistent with the analytic relationships developed in Section 6.6 regarding feature estimate accuracy. The orientational accuracy of the estimated planes-of-rotation (6.12) and the positional accuracy of the estimated centers-of-rotation (6.13) are both inversely proportional to the square root of the number of measurements. Feature estimate accuracy thus has a direct impact upon the performance of Controller S.

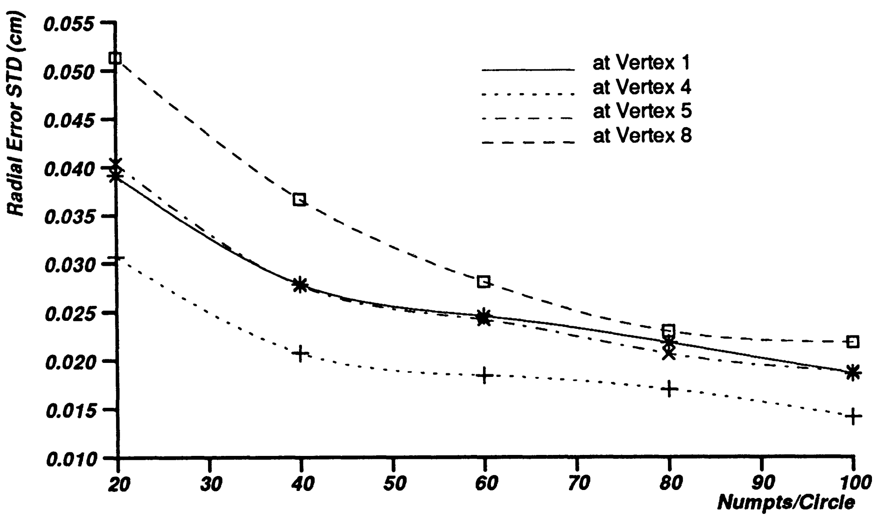


Figure 7-7: Radial Position Error Standard Deviation as a Function of the Number of Points per Circle, N_i

Consider again the nineteen performance indices which are defined such that they can take on any real value. Using signature-based control the mean values of these nineteen indices are all zero (i.e., negligible in comparison with the corresponding standard deviations) which is in strong contrast to the effect which manufacturing errors have upon the performance of D-Model Control. On the average we can expect the S-Model identification algorithm to correctly identify the true kinematic parameters of a robot and thus eliminate end-effector positioning errors. For the remaining 15 performance indices, the index's mean values are inversely proportional to the square root of N_i as illustrated in Appendix G.

7.4.5. Effect of Target Radius

In this section, we apply the simulator used in the previous section to analyze the relationship between end-effector positioning accuracy and the target radii. Naturally, the values of certain target radii will have greater impact upon the overall performance of the robot than will others. Since it would be impractical to simulate all the possible combinations of the values of the six target radii and interpret the results, we have limited our analysis to the situation in which all the target radii are equal. Thus in each simulation, $\mathbf{R}_1 = \mathbf{R}_2 = \ldots = \mathbf{R}_6 = \mathbf{R}_{nom}$. Five simulations were performed in which the value of $\mathbf{R}_{nom}$ is varied between 30.0 cm and 50.0 cm. The standard deviations of the 34 performance indices as a function of $\mathbf{R}_{nom}$ are plotted in Appendix G. The plot of the Radial Position Error standard deviations is reproduced in Figure 7-8.

While the parameter N_i has an effect upon both the accuracy of the identified planes-of-rotation and centers-of-rotation, the parameter $\mathbf{R}_{nom}$ only effects the accuracy of the planes-of-rotation. Taken independently, N_i and $\mathbf{R}_{nom}$ are both inversely proportional to the accuracy of the identified planes-of-rotation. From the findings in the previous section we might thus expect that increasing $\mathbf{R}_{nom}$ would have less of an overall effect upon the performance index standard deviations than does N_i . However, from Figure 7-8 and the plots in Appendix G this is not the case. The curves in Figure 7-8 are approximated more closely by an inverse square relationship rather than by an inverse relationship. The cumulative accuracy of an identified arm signature appears to be more sensitive to orientational errors than positional errors. This could be explained by the fact that orientational errors, especially those involved in the description of a manipulators positional subsystem, not only propagate into orientational errors at the end-effector but they are also transformed and amplified into positional errors at the end-effector.

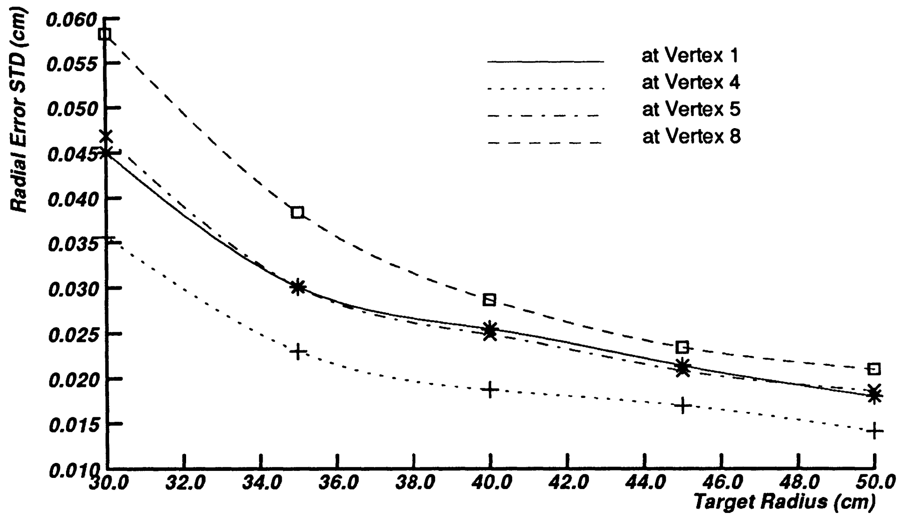


Figure 7-8: Radial Position Error Standard Deviation as a Function of Target Radius

The amount of amplification being proportional to the lengths of the links and the configuration of the robot.

The variations in R_{nom} have no effect upon the mean values of the standard deviations of the performance indices which are defined over all the real numbers. As in the previous section, the mean values for these 19 performance indices are zero. For the remaining 15 performance indices, the indices' mean values are, in general, inversely proportional to R^2_{nom} as illustrated in Appendix G.

These findings suggest that the best strategy for increasing signature accuracy is to first increase the target radii followed by increases in the number of measurements. The cost of increasing the number of measurements is the increased time required to obtain all the target measurements. This cost is incurred during the identification of each individual signature. In contrast, the cost associated with increasing $\mathbf{R}_{nom}$ is simply the reconstruction of a set of target mounting fixtures. This cost has only to be incurred once since a single set of fixtures can be applied to all robots of the same model.

7.5. Conclusions

In this chapter, we have formulated a methodology by which to evaluate and compare the statistical performance of a conventional design model-based kinematic controller and a signature-based kinematic controller. The design model controller incorporates the robot kinematic parameters obtained from the mechanical design specifications. In contrast, the signature-based controller incorporates the robot kinematic parameters estimated by the S-Model identification algorithm. Furthermore, we have applied this methodology to evaluate and compare the statistical performance of these two controllers when applied to control the joint positions of a Puma 560 robot. This work complements our experimental evaluation of the performance of seven Puma 560 robots which was described in Chapter 6.

Our methodology utilizes Monte-Carlo simulation techniques for the generation of end-effector positioning and orienting error distributions. The evaluation of the performance of the D-Model Controller, requires the formulation of a robot kinematic model which explicitly incorporates the manufacturing errors as parameters. A kinematic model of a Puma 560 robot with manufacturing errors was developed and serves to illustrate how such a model could be developed for other robots. The performance of the S-Model Controller depends upon the types of the errors introduced into the arm signature identification process and not upon the presense of robot manufacturing errors. Having developed a model of our ultrasonic sensor system, we simulated the arm signature identification process using a Puma 560 robot and applied the identified signatures to control the robot in performing a series of standardized tasks. The positioning and orienting accuracy of the end-effector is used to evaluate of how well S-Model Controller performed. This approach can be applied to analyze the statistical performance of the S-Model Controller for any robot with any type of sensor system by interchanging simulator modules. In Section 7.3, we devised experiments to verify that our simulator produced realistic and reliable results.

In Sections 7.4.1 and 7.4.2, we applied our Monte-Carlo simulator to investigate the relationship between encoder calibration errors and robot performance, and the relationship between machining and assembly errors and robot performance, respectively. From this investigation we have discovered that

- The performance indices standard deviations increase linearly with the standard deviation of the encoder calibration errors.
- The performance indices standard deviations increase non-linearly and monotonically with simultaneous changes in the standard deviations of the positional and orientational manufacturing errors.
- The performance of a D-Model Controller is more sensitive to encoder calibration errors than to the positional and orientational manufacturing errors.
- The expected performance of a D-Model Controller is a function of both the means and standard deviations of the manufacturing errors.

In practice then, the average performance of a robot using design model based control will be less than perfect. This finding clearly supports our claim that arm signature identification techniques are needed to improve end-effector positioning and orienting accuracy.

In Sections 7.4.3 - 7.4.5, we have also applied our Monte-Carlo simulator to understand the relationships which govern the performance of an S-Model Controller. Specifically, we were interested in the relationships between the sensor system accuracy, the identifier parameters N_i and R_i, and the end-effector positioning accuracy. We have found that,

- The performance indices' standard deviations are directly proportional to the sensor measurement error standard deviation σ and that this relationship is consistent with the analytic relationships between σ and the measures of feature estimate accuracy derived in Section 6.6.
- The performance indices' standard deviations are inversely related to the square root of the number of measurements, N_i and that this relationship is consistent with the analytic relationships between N_i and the measures of feature estimate accuracy derived in Section 6.6.
- The performance indices standard deviations are inversely related to the square of the target radii.
- The performance of an S-Model Controller is more sensitive to orientational identification errors than to positional identification errors.
- The expected performance of an S-Model Controller depends only upon the expected value of the measurement errors.

The identifier parameters, N_i and R_i , provide a simple mechanism for increasing the accuracy of the identified arm signatures and hence the performance of an S-Model Controller by a predetermined amount. Conversely, then can be used to reduce the need for extremely accurate sensor systems to measure the targets positions. Finally, the S-Model identification algorithm and control approach will, on the average, provide perfect end-effector positioning and orienting accuracy in the presense of zero-mean target measurement errors. The elimination of biases in the expected performance of a robot is a important advantage of the our approach.

Chapter 8
Summary and Conclusions

8.1. Introduction

In this chapter, we summarize (in Section 8.2) our contributions and identify (in Section 8.3) areas for future research.

8.2. Summary and Contributions

In this section, we review and enumerate the contributions of the dissertation. Our research and contributions lie in three areas: *robot kinematic modeling, robot kinematic parameter identification,* and *robot kinematic control.* We proceed through the development of the dissertation.

In Chapter 2, we reviewed robot kinematic modeling, identification, and control techniques, and established a framework in which to present our research contributions. The fundamental problem in the development of robot kinematic models is the use of geometric and trigonometric principles to systematically specify the relative positions and orientations of robot joints. The goal of a kinematic identification algorithm is to identify the parameters of a kinematic model which describe the actual position and orientation of the end-effector in terms of the measured joint positions, and which incorporates the geometrical variations in the structure caused by manufacturing errors . The kinematic control problem focuses upon the computation of the joint positions required to locate the end-effector at a desired Cartesian position and orientation.

In Chapter 3, we introduced the formulation and properties of a new model for describing the kinematic structure of any robotic manipulator with rigid links. This kinematic model, which we call the S-Model, is designed to facilitate kinematic parameter identification. Like the Denavit-Hartenberg model, which has had widespread application to robot control, the S-Model is conceptually based upon the assignment of Cartesian coordinate frames fixed relative to each of the links. The S-Model possesses three important features which make it directly applicable to kinematic identification. First, there is a considerable amount of flexibility in assigning the locations of the link coordinate frames. Second, the locations of the link coordinate frames are independent of the locations of the other link coordinate frames. And third, the Denavit-Hartenberg parameters can be easily extracted from the S-Model parameters.

In Chapter 4, we formulated the S-Model identification algorithm which can be applied to identify the kinematic parameters of any robot with rigid links. The objective of S-Model identification is to estimate the S-Model kinematic parameters from a set of $2n_r + n_p$ kinematic/mechanical features inherent to the manipulator, where n_r is the number of revolute joints and n_p is the number of prismatic joints. The kinematic features of a revolute joint are a *plane-of-rotation* and a *center-of-rotation,* and the kinematic feature of prismatic joint is a *line-of-translation.* The S-Model identification algorithm consists of four steps. The first step involves the explicit identification of the robot's kinematic features from measurements of the Cartesian position of targets placed on the robot's links. In the second step, the identified features are used to establish the positions and orientations of the S-Model link coordinate frames with respect to the sensor coordinate frame. In the third step, the elements of the S-Model link transformation matrices are evaluated and used to solve for the $6{\cdot}n$ S-Model parameters. Finally, in the fourth step, the robot's Denavit-Hartenberg parameters are determined. The advantage of the S-Model identification algorithm is that it decouples the robot kinematic identification problem into a set of simpler identification problems which can be solved using standard linear regression techniques.

In Chapter 5, we presented two algorithms for inverting general arm signature models and compared both their numerical performance and computation complexity. The inverse kinematic problem requires the solution of a set of n coupled nonlinear equations. The first algorithm is based upon Newton's method and the second algorithm is the Jacobi iterative algorithm applied specifically to the inversion of signature models.

In Chapter 6, we developed a complete prototype arm signature identification system and applied it improve the performance of seven Puma 560 robots. This system incorporates an ultrasonic range sensor (described in Section 6.3) to measure the three-dimensional Cartesian positions of the target sparkers placed on the robot's links. The relative Cartesian positioning accuracy of the sensor system is approximately $\pm$ 0.02 cm. The general characteristics of the sensor design and the overall system design, which exploits averaging over many sensor readings, offer numerous advantages for arm signature identification. In this chapter, we also applied the prototype system to improve the kinematic performance of seven Puma 560 robots. Three methods were devised to evaluate manipulator kinematic performance during which the Puma is programmed to position its end-effector at the vertices of a one-, two-, or three-dimensional grid. At each of the grid vertices, the actual relative position of the end-effector is measured. In the three-dimensional case, the actual relative orientation of the end-effector is also measured. For these robots, relative end-effector positioning and orienting accuracy was improved on the average by a factor of 5-10.

In Chapter 7, we considered the statistical performance of robot position control methods, and discussed the origin of robot positioning errors. In conventional design-model robot control, manufacturing errors contribute most to robot positioning errors. With arm signature-based robot control, robot performance is limited by sensor errors which contribute to the inaccuracy of the identified arm signature model. Monte-Carlo simulation techniques have been applied to study the relationship between the performance of design model control and manufacturing errors and the relationship between the performance of signature-based control and target measurement errors.

We believe that the contributions of this dissertation are:

- The design (in Chapter 3) of a new robot kinematic model for describing the relationship between the Cartesian position and orientation of the end-effector and the robot base coordinate frame as a function of the joint positions and a set of fixed kinematic parameters. In contrast to other models, our model called the S-Model is directly applicable to the kinematic identification problem. The analytic relationship between S-Model and the Denavit-Hartenberg Model is also derived.
- The identification (in Chapter 4) of the three mechanical/kinematic features of robotic manipulators. The plane-of-rotation and center-of-rotation features describe the behavior of revolute joints and the line-of-translation feature describes the behavior of prismatic joints. These features inherently describe the kinematic structure of a robot and provide a foundation for the design of the S-Model kinematic parameter identification algorithm.
- The development (in Chapter 4) of a general and practical technique for identifying the kinematic parameters of any robotic manipulator with rigid links. The technique, called the S-Model identification algorithm, uses measurements of the Cartesian position of targets placed on the robot's links and in the absence of measurement noise will identify the true kinematic parameters of a robot including the geometrical variations in the structure caused by robot manufacturing errors.

- The formulation of the Newton-Raphson and Jacobi Iterative algorithm (in Chapter 5) for solving the inverse kinematics problem for robots with non-simple kinematic structures. The convergence properties of both algorithms have been studies and compared based upon simulation results. The computational burden of both algorithms is also assessed. The Newton-Raphson algorithm is applied (in Chapter 6) to control the end-effector of several Puma 560 robots using identified arm signature parameters.
- The implementation of a complete prototype arm signature identification system. This prototype system has been applied (in Chapter 6) to improve robot end-effector positioning accuracy. The contributions derived from the implementation are:
    - The design (in Chapter 6) of an ultrasonic range sensing system to measure the three-dimensional positions of targets attached to the links of the robot. The targets are implemented by an ultrasonic source (sparker) and an array of four ultrasonic range detectors are used to measure the time-of-flight of the emitted ultrasonic wave.
    - The identification of the arm signatures of seven Puma 560 robots.
    - The introduction of three methods for evaluating robot kinematic performance. These methods, referred to as the one-, two-, and three-dimensional grid touching tasks, provide a practical means for evaluating robot kinematic performance using relatively inexpensive equipment.
    - The comparative evaluation of the actual performance of the Puma 560 robots using design model based control and signature-based control. As anticipated the identified arm signatures significantly improved the performance of all seven robots in comparison to their performance with design model based control.
    - The demonstration of the applicability and practicality of the S-Model identification algorithm towards improving robot end-effector positioning and orienting accuracy.
- The formulation of general methodology (in Chapter 7) for

evaluating and comparing the statistical performance of robots using design model based control and signature-based control. The methodology is based upon the use of Monte-Carlo simulation techniques to generate end-effector positioning and orienting errors arising from the presence of either robot manufacturing errors or measurement errors in the kinematic parameter identification process. This formulation included:

- The development (in Chapter 7) of a technique for modeling the kinematics of robots with manufacturing errors. We apply this technique to a Puma 560 and use the resulting model to analyze the effects (in Sections 7.4.1 and 7.4.2) of random manufacturing errors upon end-effector position accuracy.
- The evaluation and comparison (in Chapter 7) of the statistical performance of a Puma 560 robot using design model based control and signature-based control. The evaluation leads to *design guidelines* for specifying the signature identification parameters (i.e., the target radii and the number of measurements) and the accuracy of the sensor system used to obtain the target measurements.

8.3. Suggestions for Future Research

We suggest the following research directions to extend the contributions of this dissertation:

- The development of alternative sensor systems for measuring the three-dimensional positions of the targets attached to the robot's links. The performance of the S-Model identification algorithm is limited by the accuracy of the sensor system. The ultrasonic sensor system used in our prototype system is believed to have an accuracy of approximately $\pm$ 0.02 cm. This estimate of the accuracy was obtained indirectly, however, since the physical locations of the reference surfaces of the microphones are impossible to specify. We suspect that there are in fact small biases in the individual range measurements but have been unable to derive a method for identifying them. Methods for measuring and/or identifying the biases in such systems are needed to further improve the accuracy of identified arm signatures using ultrasonic range measurements. The presence of such biases eventually lead to bias errors in the estimated target positions and hence limit the accuracy of the identified signatures.

- The development of techniques for modeling and identifying the kinematics of robots with compliance.
- The development of customized and systematically-organized algorithms to reduce the computational requirements for the online implementation of the Newton-Raphson and Jacobi Iterative Iterative algorithms.
- The development of special purpose hardware for implementing kinematic control algorithms to reduce the overall controller sampling rate. As mentioned in Chapter 5 a significant percentage of the computations required by the Jacobi Iterative algorithm can be performed in parallel. Computer architectures specifically designed to exploit this structure and/or the structure of other kinematic inversion algorithms would be of great interest to the robotics community.

We believe that progress in these directions will extend the contributions of this dissertation thus leading to further improvements in robot kinematic performance and will hopefully expand the range of robotic applications.

Appendix A

Primitive Transformations

The *six* primitive homogeneous transformations matrices are

$$\mathrm{Trans}(x, 0, 0) = \begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Trans}(0, y, 0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Trans}(0, 0, z) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Rot}(x, \theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Rot}(y, \theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathrm{Rot}(z, \theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Appendix B
Ideal Kinematics of the Puma 560

The following sections list the forward and inverse kinematic relations for Puma 560 robot assuming negligible manufacturing errors (i.e., the ideal kinematics). A derivation of the inverse kinematics can be found in [19][10].

B.1. Forward Kinematics

The forward kinematics of the Puma 560 robot are defined by the matrix

$$T_6 = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{B.1}$$

where

$$\begin{aligned} n_x &= C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - \\ &\quad S_1[S_4C_5C_6 + C_4S_6] && (B.2) \\ n_y &= S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] + \\ &\quad C_1[S_4C_5C_6 + C_4S_6] && (B.3) \\ n_z &= -S_{23}(C_4C_5C_6 - S_4S_6) - C_{23}S_5C_6 && (B.4) \\ o_x &= C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] - \end{aligned}$$

[10]The derivation in [19] contains two errors which have been corrected in the above equations.

$$\begin{aligned}
& S_1[-S_4C_5S_6 + C_4C_6] && (B.5)\\
o_y &= S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] + \\
& C_1[-S_4C_5S_6 + C_4C_6] && (B.6)\\
o_z &= -S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6 && (B.7)\\
a_x &= C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5 && (B.8)\\
a_y &= S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5 && (B.9)\\
a_z &= -S_{23}C_4S_5 + C_{23}C_5 && (B.10)\\
p_x &= C_1(d_4S_{23} + a_3C_{23} + a_2C_2) - d_3S_1 && (B.11)\\
p_y &= S_1(d_4S_{23} + a_3C_{23} + a_2C_2) + d_3C_1 && (B.12)\\
p_z &= -(-d_4C_{23} + a_3S_{23} + a_2S_2) . && (B.13)
\end{aligned}$$

S_i, C_i, S_{ij}, and C_{ij} refer to $\sin\theta_i$, $\cos\theta_i$, $\sin(\theta_i+\theta_j)$, and $\cos(\theta_i+\theta_j)$, respectively.

B.2. Inverse Kinematics

The inverse kinematics of the Puma 560, in terms of the known elements of $\mathbf{T}_6$ in (B.1), are

$$\theta_1 = \tan^{-1}\frac{p_y}{p_x} - \tan^{-1}\frac{d_3}{(r_1^2 - d_3^2)^{1/2}} \qquad (B.14)$$

$$\theta_2 = (\theta_2+\theta_3) - \theta_3 \qquad (B.15)$$

$$\theta_3 = \tan^{-1}\frac{a_3}{-d_4} - \tan^{-1}\frac{r_2}{\pm(r_3 - r_2^3)^{1/2}} \qquad (B.16)$$

$$\theta_2+\theta_3 = \tan^{-1}\frac{(d_4+a_2S_3)(p_xC_1+p_yS_1) - (a_2C_3+a_3)p_z}{(a_2C_3+a_3)(p_xC_1+p_yS_1) + (d_4+a_2S_3)p_z} \qquad (B.17)$$

$$\theta_4 = \begin{cases} \tan^{-1}\dfrac{-a_xS_1 + a_yC_1}{C_{23}(a_xC_1 + a_yS_1) - a_zS_{23}} & \text{if } \theta_5 > 0 \\[2ex] \tan^{-1}\dfrac{-a_xS_1 + a_yC_1}{C_{23}(a_xC_1 + a_yS_1) - a_zS_{23}} + 180 & \text{if } \theta_5 < 0 \end{cases} \qquad (B.18)$$

$$\theta_5 = \tan^{-1}\frac{C_4[C_{23}(a_xC_1 + a_yS_1) - a_zS_{23}]}{S_{23}(a_xC_1 + a_yS_1) + a_zC_{23}} + \qquad (B.19)$$

$$\frac{S_4[-a_xS_1 + a_yC_1]}{S_{23}(a_xC_1 + a_yS_1) + a_zC_{23}}$$

$$\theta_6 = \tan^{-1}\frac{-C_5\{C_4[C_{23}(o_xC_1+o_yS_1)-o_zS_{23}]+}{-S_4[C_{23}(o_xC_1+o_yS_1)-o_zS_{23}]}$$

$$\frac{S_4[-o_xS_1+ \quad o_yC_1]\}}{+\ 0\ +}$$

$$\frac{+\ S_5\{S_{23}(o_xC_1+o_yS_1)+o_zC_{23}\}}{+\ C_4[-o_xS_1+o_yC_1]} \qquad (B.20)$$

where

$$r_1 = (p_x^2 + p_y^2)^{1/2} \ , \qquad (B.21)$$

$$r_2 = (p_xC_1+p_yS_1)^2 + p_z^2 - d_4^2 - a_3^2 - a_2^2 \ , \qquad (B.22)$$

and

$$r_3 = 4a_2^2a_3^2 + 4a_2^2d_4^2 \qquad \text{(Constant)} \ . \qquad (B.23)$$

When $\theta_5 = 0$, the manipulator becomes degenerate. Both the Joint 4 and Joint 6 axes are then aligned. In this case, either θ_4 or θ_6 can be chosen arbitrarily.

Appendix C

Inverse Kinematics

C.1. Newton-Raphson Computations

Step 1: Evaluate the elements of the manipulator Jacobian, J, using the method described in [18]. In doing so the forward signature model will also be evaluated (i.e., $\mathbf{T}_6$ in (5.1)).

Step 2: Compute the positional and orientational differential matrix, ${}^0\Delta^k$, with respect to the manipulator's base coordinates. The matrix ${}^0\Delta^k = \mathbf{T}_6^* - \mathbf{T}_6$.

Step 3: Invert the matrix $\mathbf{T}_6$.

Step 4: Express differential matrix, ${}^0\Delta^k$, in terms of end-effector coordinates. In other words, compute the elements of the matrix ${}^6\Delta^k = [\mathbf{T}_6]^{-1}\ {}^0\delta^k$.

Step 5: Express differential matrix, ${}^6\Delta^k$, in vector form as denoted by $\vec{\delta}^k$ [18].

Step 6: Solve the system of linear equations $\vec{\delta}^k = \mathrm{J}\,\vec{dq}$ where $\vec{dq} = \vec{q}^k - \vec{q}^{k-1}$.

Step 7: Update solution vector according to $\vec{q} = \vec{q} + \vec{dq}$

Step 8: Check if the elements of the vector $\vec{dq}$ are less than the resolution of the corresponding joint encoders. If all are then terminate, else return to Step 1.

C.2. Jacobi Iterative Computations

Step 1: Compute the sines and cosines of the six joint positions.

Step 2: Compute the forward signature model (i.e., the elements of the matrix $\mathbf{F}_s(\vec{\mathbf{q}}^k)$ in (5.8)).

Step 3: Compute the forward design model (i.e., the elements of the matrix $\mathbf{F}_d(\vec{\mathbf{q}}^k)$ in (5.8)).

Step 4: Invert the forward signature model matrix $\mathbf{F}_s(\vec{\mathbf{q}}_i)$.

Step 5: Multiply the matrices $\mathbf{F}_d(\vec{\mathbf{q}}_i)$, $[\mathbf{F}_s(\vec{\mathbf{q}}_i)]^{-1}$, and $\mathbf{T}_6^*$.

Step 6: Evaluate the inverse kinematics of the design model (refer to Appendix B) using the matrix evaluated in Step 5 as the input.

Appendix D

Identified Arm Signtaures

Table D-1: Identified Arm Signature Parameters

Name	Link	Variable	θ_{offset} (deg)	d (cm)	a (cm)	α (deg)
Robot 3	1	θ_1	110.308	-20.678	-0.005	-90.066
	2	θ_2	79.977	11,210.213	7.6071	-0.217
	3	θ_3	-79.483	-11,194.976	-2.039	90.519
	4	θ_4	-0.296	43.278	-0.001	-90.007
	5	θ_5	-0.497	-0.018	0.002	89.990
	6	θ_6	-89.081	10.300	0.000	0.000
Robot 4	1	θ_1	109.570	-20.823	-0.037	-89.992
	2	θ_2	14.786	1,636.110	41.707	-0.394
	3	θ_3	-14.883	-1,621.207	-2.059	90.002
	4	θ_4	-0.101	43.388	-0.019	-90.036
	5	θ_5	0.106	-0.070	-0.011	90.023
	6	θ_6	-89.021	10.275	0.000	0.000
Robot 5	1	θ_1	110.849	-20.428	-0.023	-90.013
	2	θ_2	63.467	2,783.580	19.084	-0.803
	3	θ_3	-63.276	-2,768.574	-2.028	90.134
	4	θ_4	0.328	43.299	-0.009	-89.991
	5	θ_5	-0.126	-0.054	0.007	89.969
	6	θ_6	-89.517	10.273	0.000	0.000

Table D-1: (Continued)

Name	Link	Variable	θ_{offset} (deg)	d (cm)	a (cm)	α (deg)
Robot 6	1	θ_1	110.455	-20.672	0.013	-89.997
	2	θ_2	65.427	4,931.060	17.932	-0.459
	3	θ_3	-64.355	-4,916.063	-2.120	89.938
	4	θ_4	0.420	43.235	-0.022	-90.053
	5	θ_5	-1.081	-0.026	-0.003	90.033
	6	θ_6	-89.809	10.307	0.000	0.000
Robot 7	1	θ_1	110.849	-20.935	-0.006	-89.985
	2	θ_2	80.179	3,832.102	7.341	-0.639
	3	θ_3	-79.172	-3,817.059	-2.189	89.977
	4	θ_4	-0.688	43.387	0.010	-89.974
	5	θ_5	-1.428	-0.026	0.006	89.994
	6	θ_6	-89.771	10.282	0.000	0.000

Appendix E
Sensor Calibration

E.1. Calibration Rods

Two calibration rods are depicted in Figure 6-4. Conceptually, a calibration rod is an apparatus for securing two sparkers at a known and fixed distance apart. The nominal distance between these sparkers is 100 cm. The upper calibration rod contains sparkers 1 and 2, and the lower calibration rod contains sparkers 3 and 4. The upper calibration rod is positioned such that the tips of sparkers 1 and 2 and the acoustic zero point of Microphone B are collinear and level. Similarly, the lower calibration rod is positioned such that the tips of sparkers 3 and 4 and the acoustic zero point of Microphone D are collinear and level.

These calibration rods provide a means for accurately determining the speed of sound at a particular instance in time. Assuming that the speed of sound is constant along the line between Sparker 2 and Microphone B and along the line between Sparker 4 and Microphone D, the calibration rods provide a mechanism to accurately estimate the speed of sound at Microphone B and D. Using the upper calibration rod, the ratio of the speed of sound at Microphone B to the speed of sound internal to the GP-8-3D, S^*, is

$$f_B = \frac{D_{1,2}}{s_2 - s_1} \tag{E.1}$$

$$= \frac{\text{Calibrated distance between sparkers 1 and 2}}{\text{Measured distance between sparkers 1 and 2}} \,,$$

where s_1 and s_2 are the measured slant ranges between Sparker 1 and Microphone B and Sparker 2 and Microphone B, respectively. The ratios f_B and f_D are applied in Section E.2 to compensate the three primary slant range measurements between the target sparker and Microphones A, B, and C.

E.2. Slant Range Compensation

The method we have developed for correcting the slant ranges obtained from the GP-8-3D (i.e., the raw slant ranges) compensates for both overall variations in the speed of sound and spatial variations in the speed of sound. Applying this method, we have achieved relative Cartesian positioning accuracies on the order of the resolution of the GP-8-3D. The resulting slant range compensation algorithms are based upon a simplified model of the operating environment. According to this model,

- The effect of air currents upon the speed of sound is negligible (B-1).
- All variables upon which the speed of sound depends, such as temperature and humidity, vary slowly relative to the time required to gather an *individual* target measurement (e.g., time constants on the order of minutes) (B-2).
- The temperature of the air within the active region of the sensor varies linearly with vertical height and is constant in the two horizontal directions (B-3).
- The temperature differential between the air at the top and bottom of the sensor's active region is less than 2 °F (B-4).
- The average temperature of the air within the active volume of the sensor is between 65 °F and 75 °F (B-5).

Assumptions (B-1) - (B-5) are consistent with environmental characteristics of our laboratory at the Robotics Institute of Carnegie-Mellon University. Our laboratory, originally designed as a computer machine room, is approximately 30 ft x 30 ft with a 12 ft ceiling. Only a small portion of the laboratory is actually utilized. The room contains a raised floor under which cooled air from adjacent machine rooms is free to circulate. Due to the lack of heating or air conditioning vents, the temperature within the room is governed by the conduction or radiation of heat through the walls to the adjacent machine rooms. This feature insures that the temperature and humidity within the room remains relatively constant. Use of the ultrasonic sensor system for accurate measurements requires attention to these issues.

In practice, robot manufacturers who want to identify the kinematic parameters of their manipulators as an integral part of the manufacturing process may enclose the sensor system in a moderately-controlled environmental chamber such as that used for testing and qualification.

The relationship between temperature and the speed of sound is approximately linear for small variations in temperature (i.e., less than 5 °F). Thus, according to our model of the environment, the speed at which sound travels along a straight line between the target sparker and a microphone (i.e., the line of sight) varies linearly with distance traveled. Thus,

$$\frac{dx}{dt} = \mu x + \beta_2 \ , \qquad (E.2)$$

where $x = x(t)$ is the distance traveled by the ultrasonic wave front in time t (i.e., $x(0) = 0$ at the target), μ is the rate of change of the speed of sound along the line of sight, and β_2 is the speed of sound at the microphone. Integrating (E.2) from time $t = 0$ to $t = t^*$, the *actual* time of flight, the distance between the target and microphone

$$x(t^*) = \frac{\beta_2}{\mu}[\exp(\mu t^*) - 1] \ . \qquad (E.3)$$

By definition,

$$\mu = \frac{\beta_2 - \beta_1}{x(t^*)} \ , \qquad (E.4)$$

where β_1 is the speed of sound at the target sparker. If the line of sight is level, then $\mathbf{V}_1 = \mathbf{V}_2$ and the actual slant range $x(t^*) = \beta_2 t^*$.

Estimates for β_1 and β_2 must be obtained in order to solve (E.3). Using the upper and lower calibration rods depicted in Figure 6-4, we measure the speed of sound at two distinct heights (i.e., $\mathbf{f}_B$ and $\mathbf{f}_D$). Then, based upon the known relative heights of the microphones and the calibration rods, we linearly interpolate to estimate the speed of sound at Microphones A, B, and C. An additional step is required to estimate β_1 since the height of the target is initially unknown. The objective of this step is to determine an initial estimate for the Cartesian coordinates of the target from which an estimate for its height can be computed. We define the average speed-of-sound ratio over the active region of the sensor array to be

$$\mathbf{f}_{avg} = \frac{\mathbf{f}_B - \mathbf{f}_D}{2} \ . \qquad (E.5)$$

By multiplying the three raw slant ranges, s_1, s_2, and s_3, by (E.5), we compensate for overall variations in the speed of sound relative to the nominal value internal to the GP-8-3D. The compensated slant ranges are used in (6.1) - (6.6) to compute the initial estimate of the targets Cartesian coordinates. We then compute the height of the target and linearly interpolate to estimate β_1 in (E.4).

We apply Jacobi's Iterative algorithm to solve (E.3) for each of the three primary slant ranges. The resulting final estimates for the slant ranges are used in (6.1) - (6.6) to compute the final estimate of the target's Cartesian coordinates. Our method of compensating the GP-8-3D's slant range measurements requires that all five sparkers must be sparked at least once to determine the target's location. Subsequent to the sparking of the sparkers, the GP-8-3D will transmit twenty slant range measurements to the host computer. Only seven of the measurements are used in our algorithms. The remaining measurements are discarded. In practice, 10-20 sets of slant range measurements are obtained and statistically averaged to reduce the effects of random measurement noise inherent in the GP-8-3D.

Appendix F
Simulator Components

F.1. Robot Manufacturing Error Model

To understand the derivation of the model (7.1), consider the task of manufacturing the actual physical base link of a Puma 560, shown in Figure F-1.Conceptually, the base link is a cylinder. To manufacture the link, one end of the cylinder must be machined to a flat surface, the normal of which should be coincident with the axis of the cylinder. Similarly, the other end should be machined flat such that the two end surfaces are a desired distance apart. Then, a bearing race must be mounted within the top of the cylinder such that the centroid of the bearing is coincident with the end surface and the axis of the bearing is coincident with the axis of the cylinder. Later on the base link and Link 1 will be joined. The lower end of Link 1 will contain the mating portion of the bearing race which is also characterized by an axis and a centroid. Ideally, during the process the axes and centroids of the two mating pairs will be perfectly aligned. Similar manufacturing processes can be envisioned for the remaining links.

Imagine two coordinate frames fixed to the components of the link. Let the Z axis of the first coordinate frame be coincident with the axis of the cylinder and let its origin be coincident with the center of the bottom surface of the cylinder. Let the Z axis of the second coordinate frame be coincident with the axis of the bearing and let its origin be coincident with the centroid of the bearing. Kinematically then, a perfectly manufactured base link is

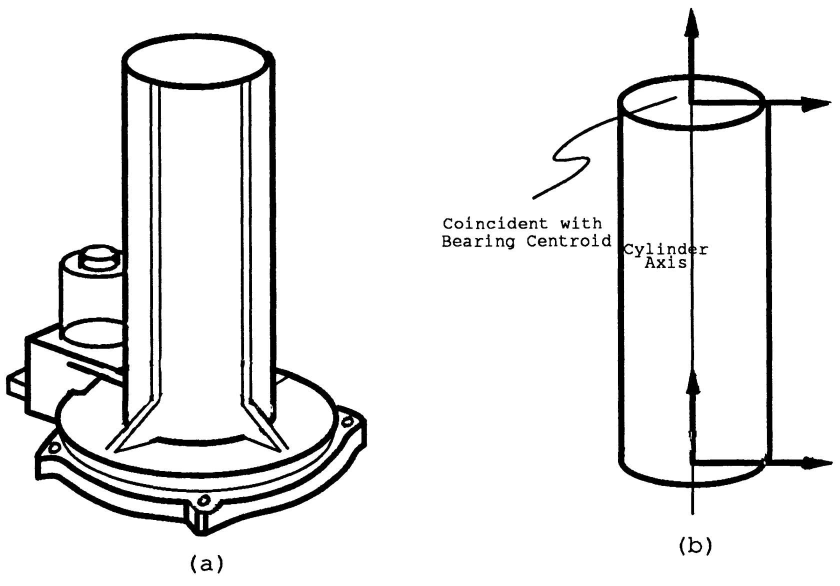


Figure F-1: Base Link of a Puma 560. (a) Actual (b) Conceptual

described by the transformation matrix Trans (0.0, 0.0, 67.4*cm*). However, when manufacturing errors are present these two coordinate frames will not be perfectly aligned. For instance, the bottom surface of the link may not be precisely perpendicular to the axis of the cylinder. In this case, we can describe the kinematic structure of the link by the new transformation matrix

$$\mathrm{Rot}(x, \varepsilon_4)\,\mathrm{Rot}(y, \varepsilon_5)\,\mathrm{Trans}(0.0, 0.0, 67.4cm) \quad , \qquad (F.1)$$

where ε_1 and ε_2 represent small random deviations in the orientation of the machined bottom surface. The machining of the top of the cylinder and the mounting of the bearing race may lead to similar orientational errors at the other end of the link. Taking these errors into account (F.1) becomes

$$\mathrm{Rot}(x, \varepsilon_4)\,\mathrm{Rot}(y, \varepsilon_5)\,\mathrm{Trans}(0.0, 0.0, 67.4cm)\,\mathrm{Rot}(x, \varepsilon_6)\,\mathrm{Rot}(y, \varepsilon_7)\ . \qquad (F.2)$$

By modifying (F.2) further, to account for possible errors in the length of the cylinder, errors in locating the center of the base, and errors in calibrating the joint encoder we arrive at the model (7.2) to describe the actual kinematic structure of the base link. This same methodology has been applied to develop the models (7.3) - (7.8) of Links 1 through 6.

F.2. Simulator Input Parameters

This section defines input parameters must be specified in order to simulate the performance of Controller S. The values for these parameters which coincide with our prototype arm signature identification system are also included. Linear parameters are measured in centimeters and angular parameters are measured in degrees.

The spatial transformation between the sensor coordinate frame and the robot base coordinate frame is

$$\begin{aligned} {}^{s}\mathrm{T}_b = \ & Trans(106.16, 56.98, 87.82)\,Rot(z, 45.0) \\ & Rot(x, -90.0)\,Rot(z, 155.0) \\ & Trans(0.0, -20.30, -67.183)\ . \end{aligned} \qquad (F.3)$$

The spatial transformation between the robot base coordinate frame and the Denavit-Hartenberg Link 0 coordinate frame is

$${}^{b}\mathrm{T}_0 = Trans(0.0, 0.0, 67.183) \qquad (F.4)$$

Six targets are attached to the robot's links during the identification procedure. By definition, Target i is fixed to Link i. The position of Target i with respect to the base coordinate frame is specified by the position vector component of the transformation matrix

$$A_1 \cdot A_2 \cdot \ldots \cdot A_{i-1} \cdot Rot(z, \theta_i) \cdot Offset_i \cdot Trans(\mathbf{R}_i, 0.0, 0.0) \qquad (F.5)$$

where the matrices A_j are the Denavit-Hartenberg link transformation matrices in (2.9), θ_i is the angular position of Joint i, and $\mathbf{R}_i$ is the Target i nominal radius. Values for the nominal target radii are listed in Table 6-1. The offset transformation matrices

$$Offset_i = Trans(0.0, 0.0, z_i) \cdot Rot(x, rx_i) \qquad \text{for } i = 1, 2, \ldots, 6 \qquad (F.6)$$

are defined by the parameters z_i and rx_i. The values for these parameters are listed in Table F-1.

Table F-1: Offset Transformation Matrix Parameters

Transformation	Parameter Values	
	z_i (cm)	rx_i (deg)
$Offset_1$	20.300	49.00
$Offset_2$	33.309	14.47
$Offset_3$	18.500	-82.25
$Offset_4$	53.531	-180.00
$Offset_5$	3.200	-90.00
$Offset_6$	10.225	-180.00

When the position of joint i is zero, the offset transformation, $Offset_i$, defines the spatial relationship between the Denavit-Hartenberg Link i-1 coordinate frame and a coordinate frame whose Z axis is coincident the the joint i axis and whose X axis is directed towards Target point i.

The actual distances between the microphones in the sensor array are $W_{A,B}$ = 143.0 cm, $W_{B,C}$ = 143.0 cm, and $W_{A,C}$ = 203.0 cm (See Appendix E).

Appendix G
Simulation Results

G.1. Encoder Calibration Errors

The statistical variations in the performance indices of 500 simulated robots are plotted in Figures G-1 and G-2 as a function of σ_e. In the plots in Figure G-1, the vertical axes denote the standard deviation in the respective performance index. In the plots in Figure G-2, the vertical axes denote the mean in the respective performance index.

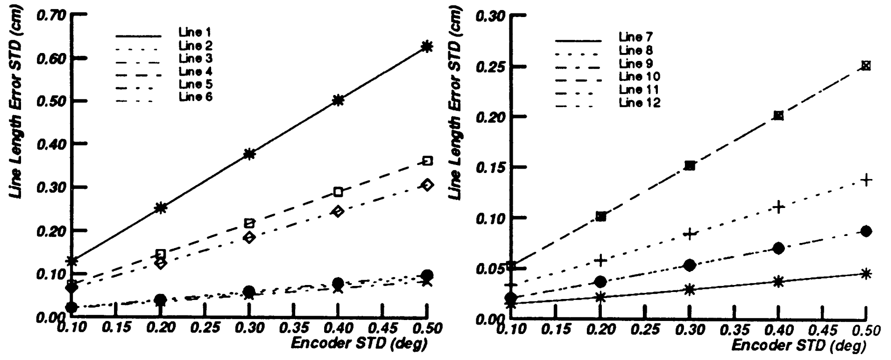


Figure G-1: Performance Index Standard Deviations as a Function of the Encoder Calibration Error Standard Deviation, σ_e

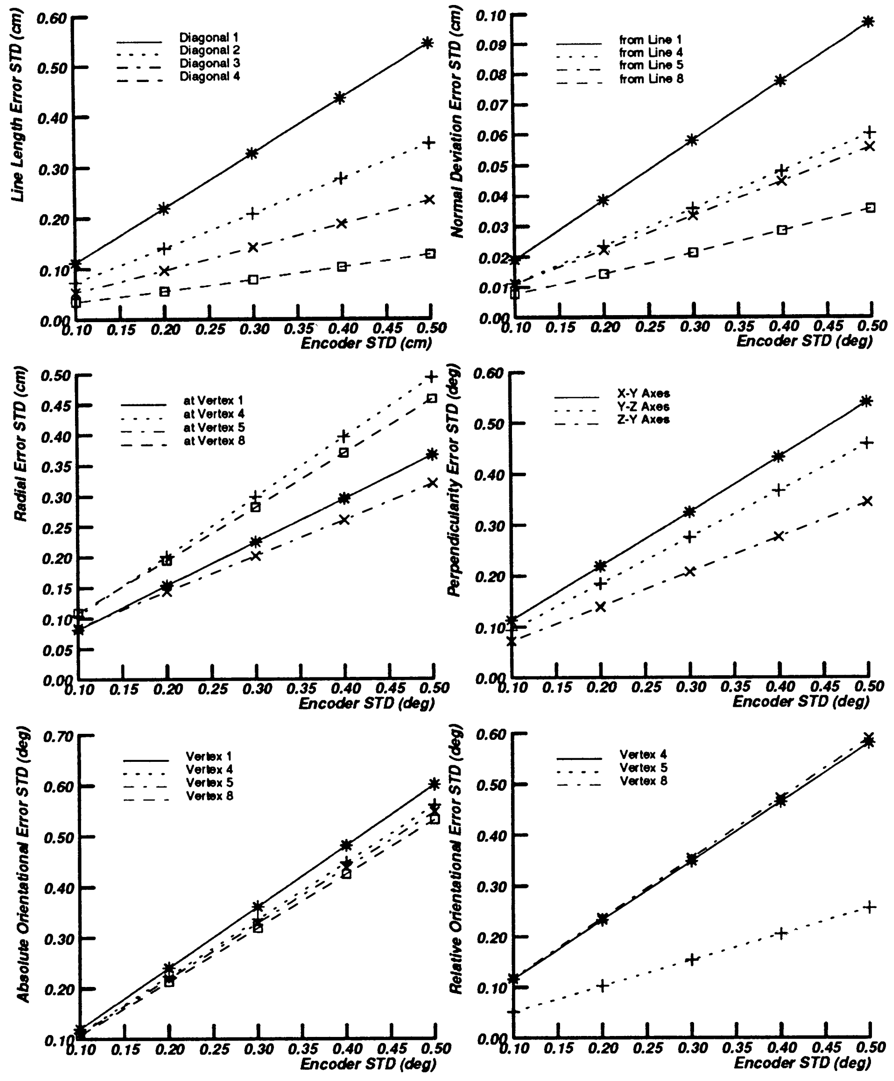


Figure G-1: (Continued)

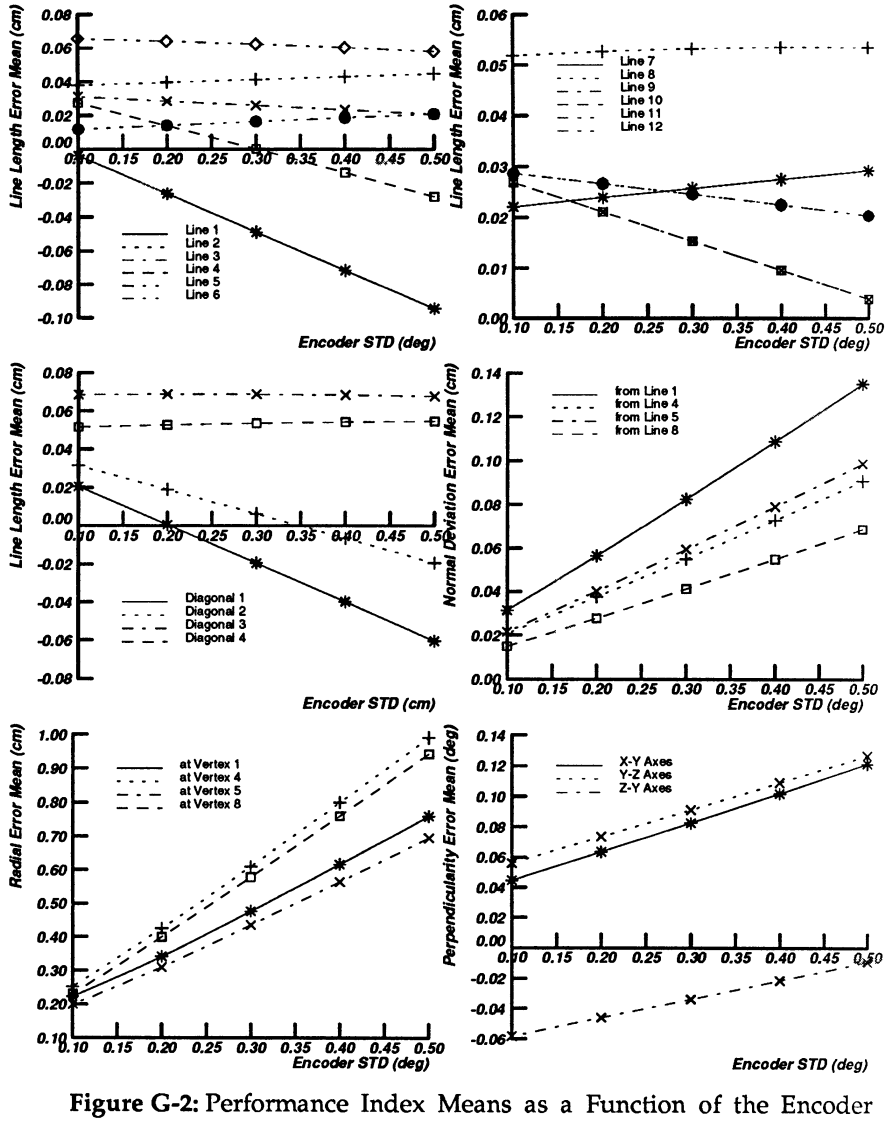


Figure G-2: Performance Index Means as a Function of the Encoder Calibration Error Standard Deviation, σ_e

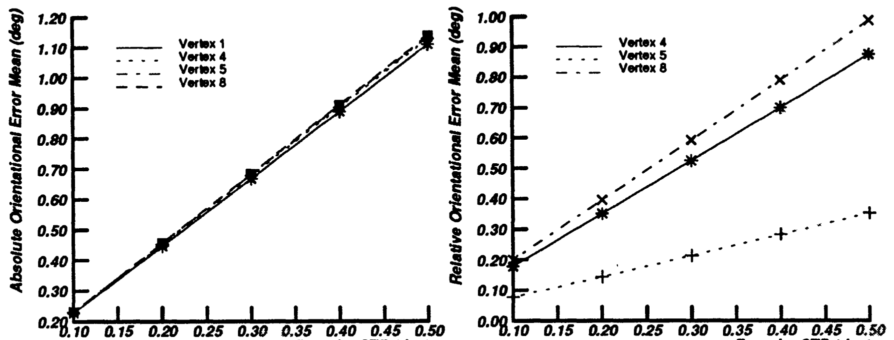


Figure G-2: (Continued)

G.2. Machining and Assembly Errors

The statistical variations in the performance indices of 500 simulated robots are plotted in Figures G-3 and G-4 as a function of σ_e. In the plots in Figure G-3, the vertical axes denote the standard deviation in the respective performance index. In the plots in Figure G-4, the vertical axes denote the mean in the respective performance index.

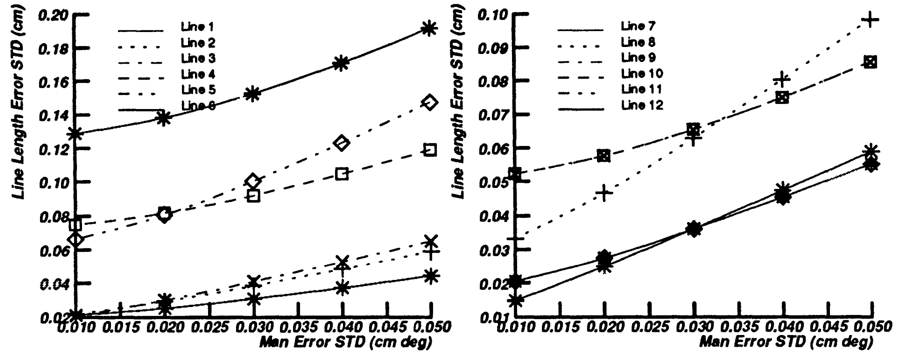


Figure G-3: Performance Index Standard Deviations as a Function of Simultaneous Variations in the Positional Error Standard Deviation and the Orientational Error Standard Deviation, σ_p and σ_o

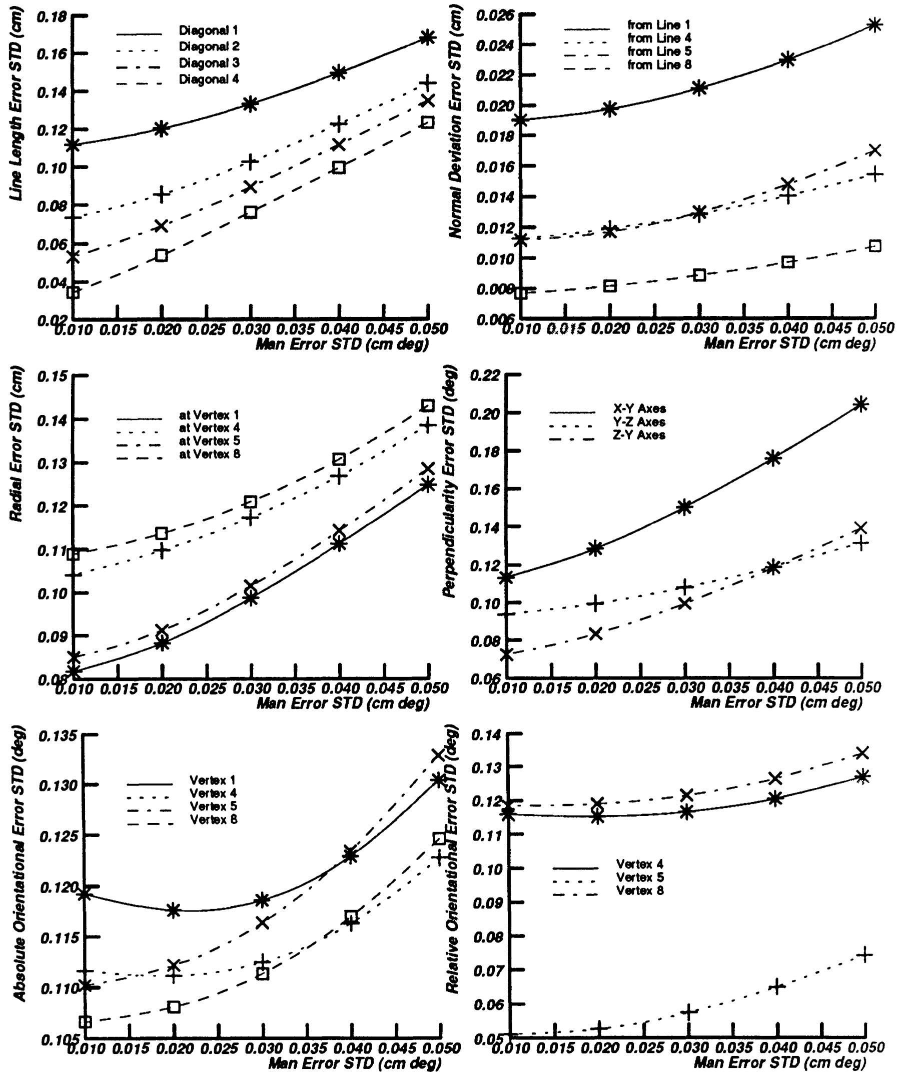


Figure G-3: (Continued)

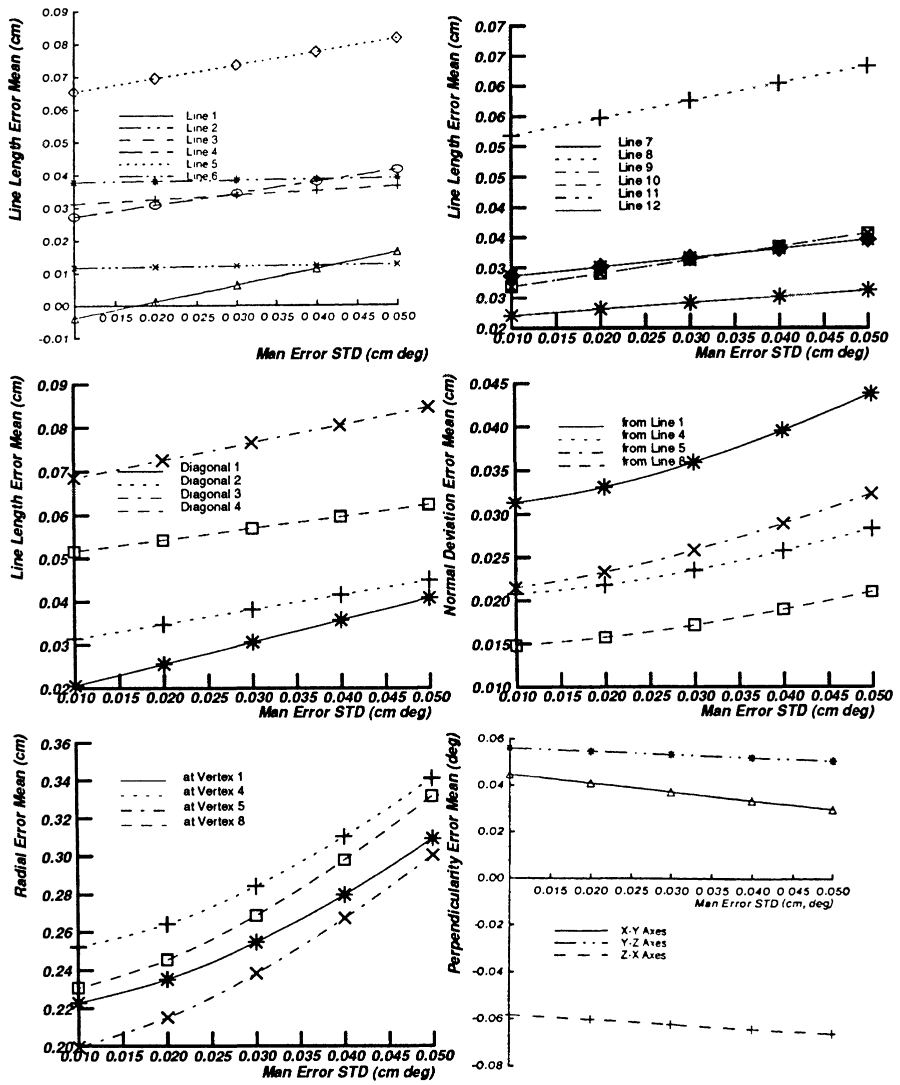


Figure G-4: Performance Index Means as a Function of Simultaneous Variations in the Positional Error Standard Deviation and the Orientational Error Standard Deviation, σ_p and σ_o

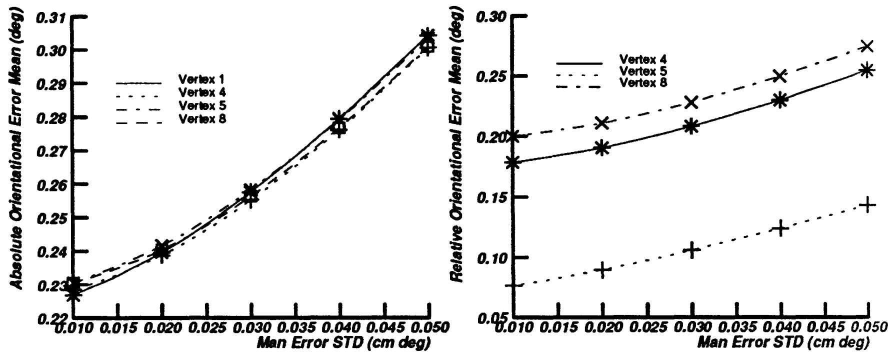


Figure G-4: (Continued)

G.3. Sensor Measurement Errors

The statistical variations in the performance indices of 500 simulated robots are plotted in Figure G-5 and G-6 as a function of N_i. In the plots in Figure G-5, the vertical axes denote the standard deviation in the respective performance index. In the plots in Figure G-6, the vertical axes denote the standard deviation in the respective performance index.

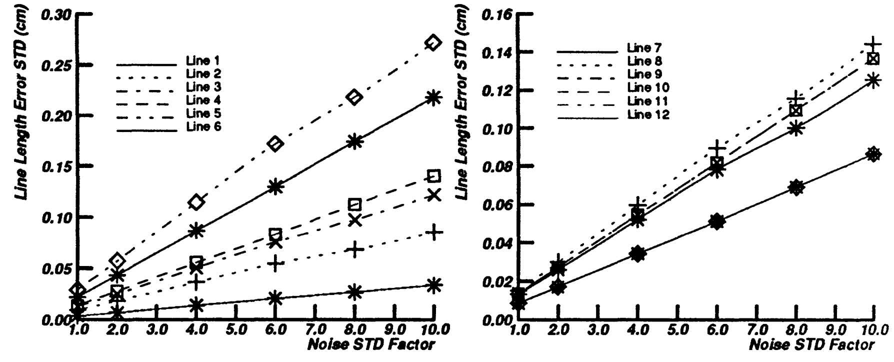


Figure G-5: Performance Index Standard Deviations as a Function of the Measurement Error Standard Deviation, σ.

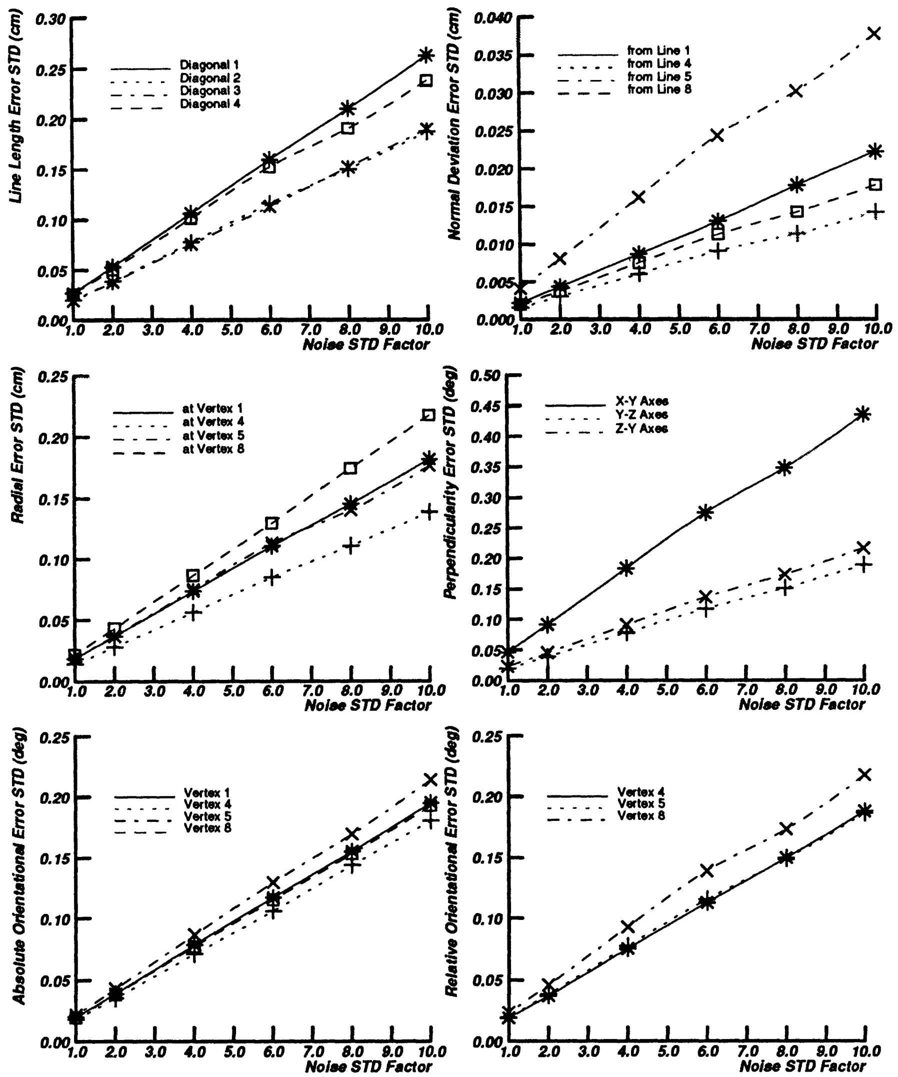


Figure G-5: (Continued)

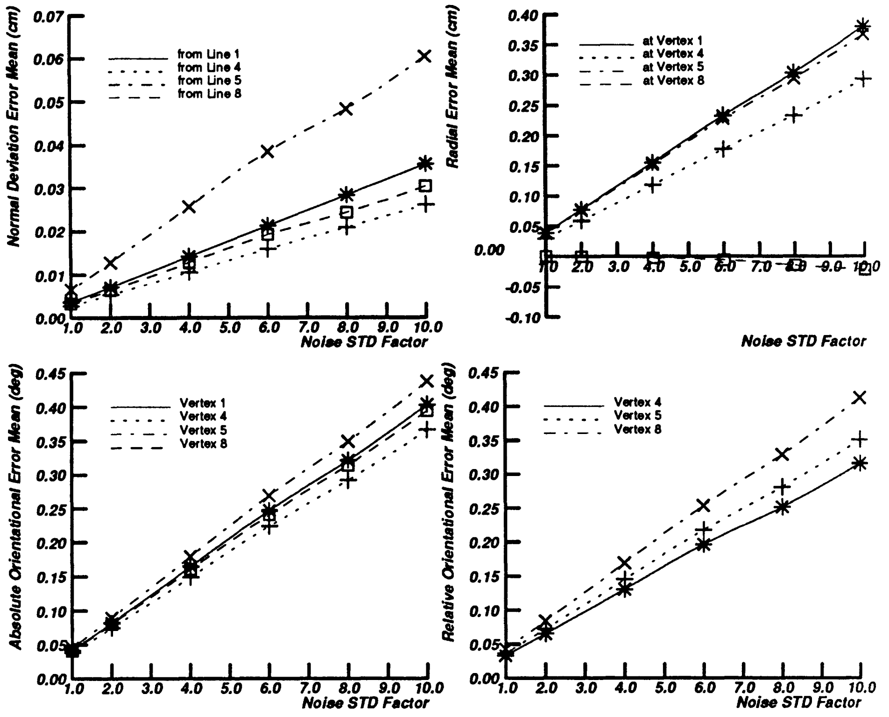


Figure G-6: Performance Index Means as a Function of the Measurement Error Standard Deviation, σ.

G.4. Number of Measurements

The statistical variations in the performance indices of 500 simulated robots are plotted in Figure G-7 and G-8 as a function of N_i. In the plots in Figure G-7, the vertical axes denote the standard deviation in the respective performance index. In the plots in Figure G-8, the vertical axes denote the standard deviation in the respective performance index.

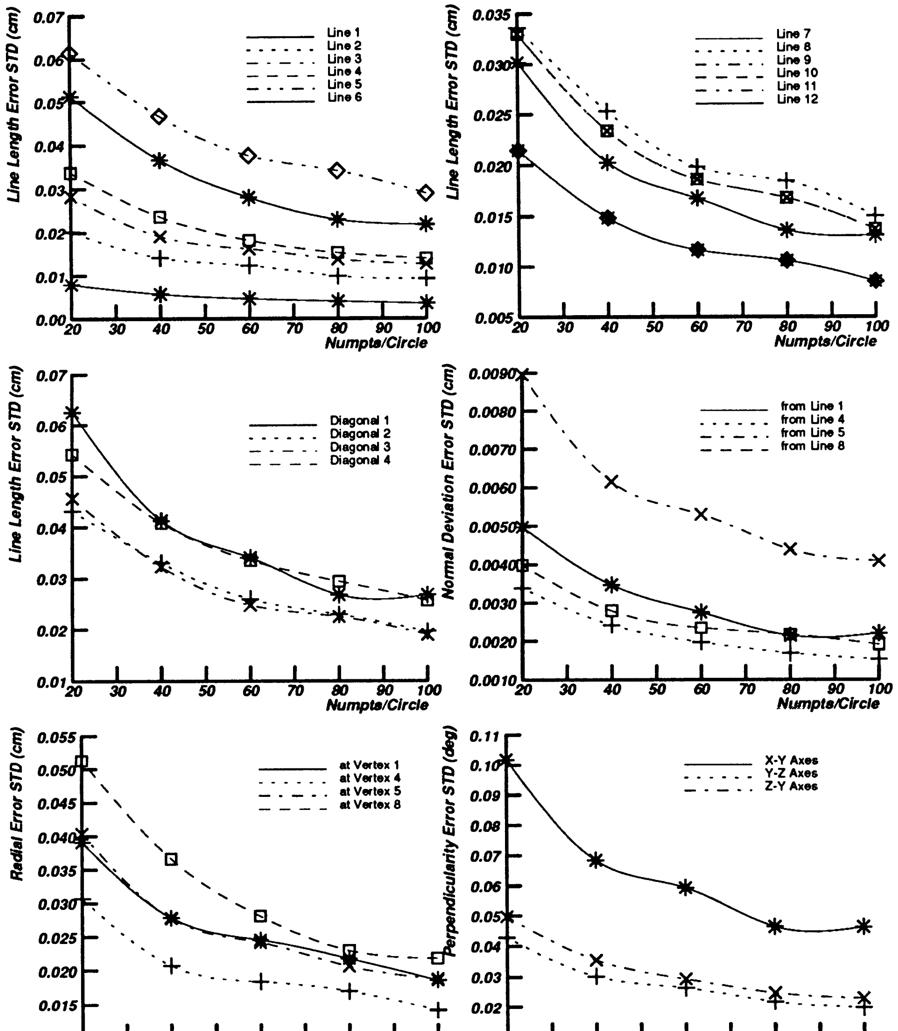


Figure G-7: Performance Index Standard Deviations as a Function of the Number of Points per Circle, N_i

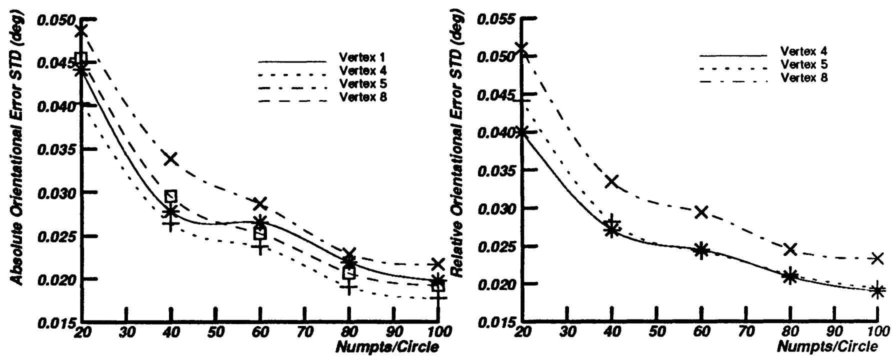


Figure G-7: (Continued)

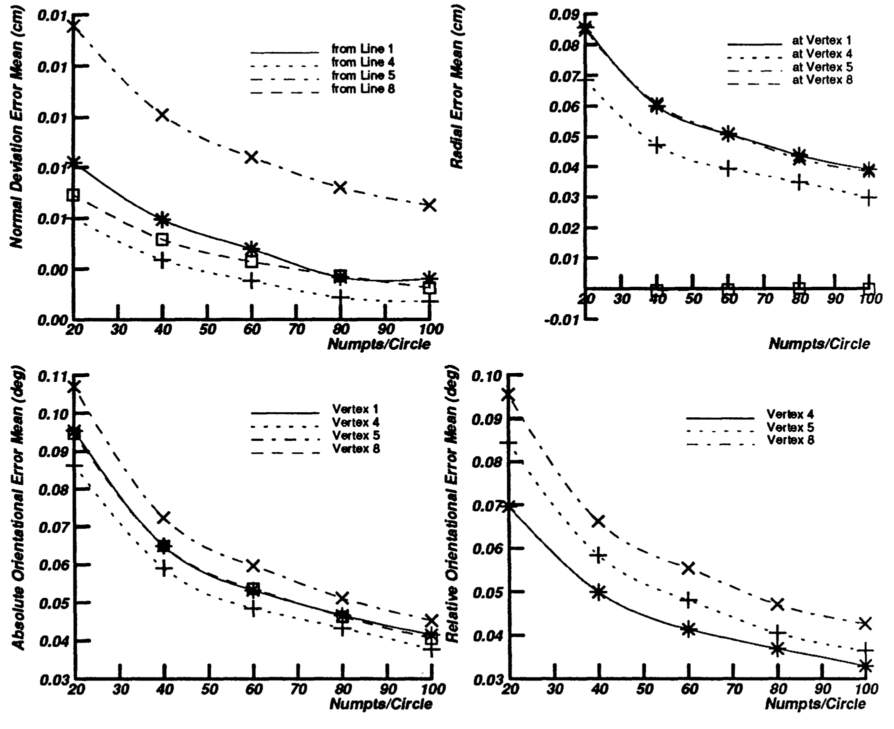


Figure G-8: Performance Index Means as a Function of the Number of Points per Circle, N_i

G.5. Target Radius

The statistical variations in the performance indices of 500 simulated robots are plotted in Figure G-9 and G-10 as a function of Target Radius. In the plots in Figure G-9, the vertical axes denote the standard deviation in the respective performance index. In the plots in Figure G-10, the vertical axes denote the mean in the respective performance index.

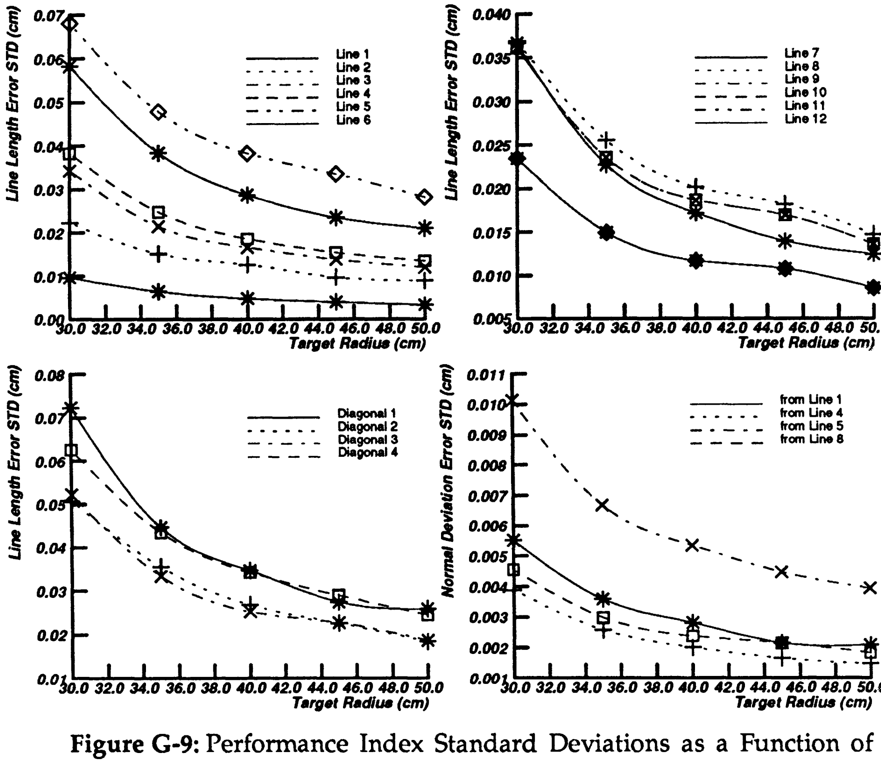
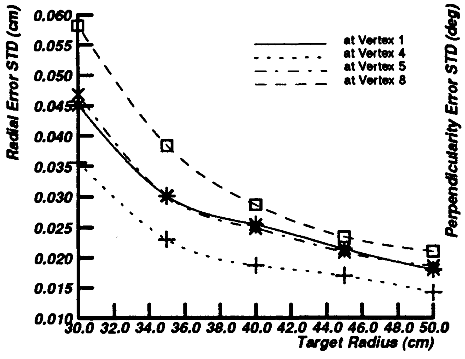
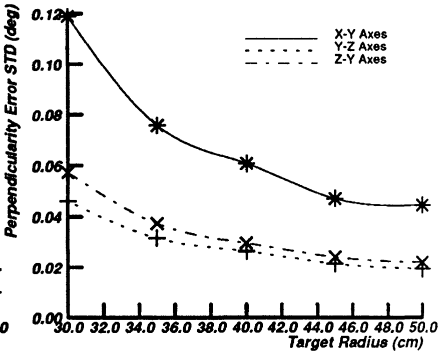
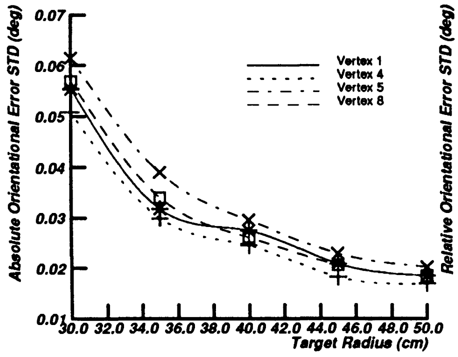


Figure G-9: Performance Index Standard Deviations as a Function of Target Radius







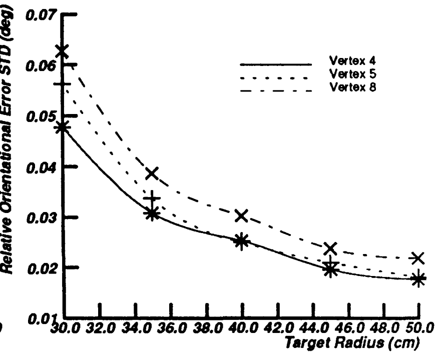
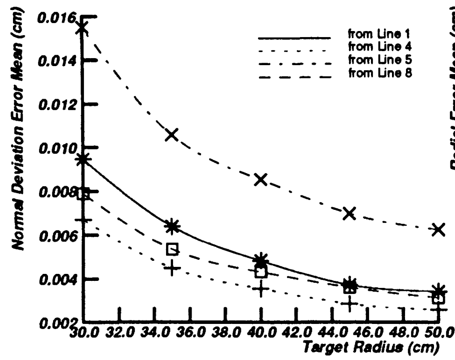


Figure G-9: (Continued)



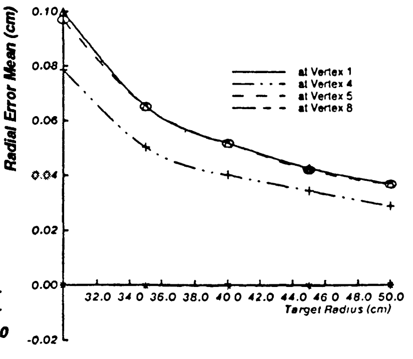


Figure G-10: Performance Index Means as a Function of Target Radius

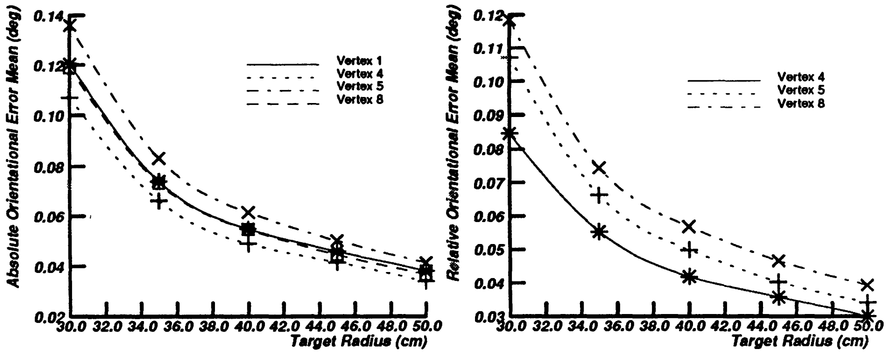


Figure G-10: (Continued)

REFERENCES

[1] Agin, G. J.
Fitting Ellipses and General Second-Order Curves.
Technical Report CMU-RI-TR-81-5, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, July, 1981.

[2] Ball, R. S.
The Theory of Screws.
Cambridge University Press, Cambridge, MA, 1900.

[3] Ballard, D. H. and Brown, C. M.
Computer Vision.
Prentice Hall, Englewood Cliffs, NJ, 1982.

[4] Chen, J., Wang, C. B., and Yang, C. S.
Robot Positioning Accuracy Improvement through Kinematic Parameter Identification.
In *the Proceedings of the Third Canadian CAD/CAM in Robotics Conference*, pages 4.7-4.12. June, 1984.
Toronto, Canada.

[5] Denavit, J. and Hartenberg, R. S.
A Kinematic Notation for Lower-Pair Mechanisms Based on Matrices.
Journal of Applied Mechanics 77(2):215-221, June, 1955.

[6] Duda, R. O. and Hart, P. E.
Pattern Classification and Scene Analysis.
John Wiley and Sons, New York, 1973.

[7] Goldenberg, A. A., Benhabib, B., and Fenton, R. G.
A Complete Generalized Solution to the Inverse Kinematics of Robots.
Journal of Robotics and Automation RA-1(1):14-20, March, 1985.

[8] Hayati, S. and Mirmirani, M.
Puma 600 Robot Arm Geometric Calibration.
In *The First IEEE International Conference on Robotics*. March 13, 1984.
Atlanta, GA.

[9] Henrici, P.
Essentials of Numerical Analysis, with Pocket Calculator Demonstrations.
John Wiley and Sons, New York, 1982.

[10] Isaacson, E. and Keller, J. B.
Analysis of Numerical Methods.
John Wiley and Sons, New York, 1966.

[11] Kernighan, B. W. and Ritchie, D. M.
The C Programming Language.
Prentice-Hall, Englewood Cliffs, NJ, 1978.

[12] Khosla, P. K., Neuman, C. P., and Prinz, F. B.
An Algorithm for Seam Tracking Applications.
The International Journal of Robotics Research 4(1):27-41, Spring, 1985.

[13] Kirchner, H. O. K., Gurumoorthy, B., and Prinz, F. B.
A Perturbation Approach to Robot Calibration.
Technical Report CMU-RI-TR-85-9, Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, April, 1985.

[14] Mooring, B. W.
The Effect of Joint Axis Misalignment on Robot Positioning Accuracy.
In *the Proceedings of the International Computers in Engineering Conference*, pages 151-155. ASME, New York, August, 1983.
Chicago, IL.

[15] Mooring, B. W. and Tang, G. R.
An Improved Method for Identifying the Kinematic Parameters in a Six Axis Robot.
In *the Proceedings of the International Computers in Engineering Conference*, pages 79-84. ASME, New York, August, 1984.
Las Vegas, NV.

[16] Motorola Incorporated.
MC68020 32 Bit Microprocessor User's Manual
Prentice Hall, Englewood Cliffs, NJ, 1985.

[17] Motorola Incorporated.
MC68881 Floating-Point Coprocessor User's Manual
Prentice Hall, Englewood Cliffs, NJ, 1985.

[18] Paul, R. P.
Robot Manipulators: Mathematics, Programming, and Control.
The MIT Press, Cambridge, MA, 1981.

[19] Paul, R. P., Shimano, S., and Mayer, G. E.
Kinematic Control Equations for Simple Manipulators.
IEEE Transactions on Systems, Man, and Cybernetics SMC-11(6):449-455, June, 1981.

[20] Pieper, D. L.
The Kinematics of Manipulators under Computer Control.
Artifical Intelligence Laboratory Memo AIM-72, Department of Computer Science, Stanford University, Stanford, CA, 1968.

[21] Ranky, P. G.
Test Method and Software for Robot Qualification.
The Industrial Robot 11(2):111-115, June, 1984.

[22] *GP-8-3D Sonic Digitizer Operator's Manual*
Science Accessories Corporation, Southport, CN, 1985.

[23] Stauffer, R. N.
Robot Accuracy.
Robotics Today 7(2):43-49, April, 1985.

[24] Strang, G.
Linear Algebra and Its Applications.
Academic Press, New York, 1980.

[25] Sugimoto, K. and Duffy, J.
Application of Linear Algebra to Screw Systems.
Mechanism and Machine Theory 17(1):73-83, 1982.

[26] Suh, C. and Radcliffe, C. W.
Kinematics and Mechanism Design.
John Wiley and Sons, New York, 1978.

[27] *Unimate Puma Mark II 500 Series Equipment Manual 398U1*
Unimation Incorporated, Danbury, CN, 1985.

[28] *Val II Programming Language*
Unimation Incorporated, Danbury, CN, 1985.

[29] Weisberg, S.,
Applied Linear Regression.
John Wiley and Sons, New York, 1980.

[30] Whitney, D. E. and Lozinski, C. A.
Industrial Robot Calibration Methods and Results.
In *the Proceedings of the International Computers in Engineering Conference,* pages 92-100. ASME, New York, August, 1984.
Las Vegas, NV.

[31] Whitney, D. E., DeFazio, T. L., Gustavson, R. E., Rourke, J. M., Seltzer, D. S., Edsall, A. C., Lozinski, C. A., and Kenwood, G. J.
Short-Term and Long-Term Robot Feedback: Multi-Axis Sensing, Control and Updating.
In *the Proceedings of the Eleventh Conference on Production Research and Technology,* pages 147-152. Carnegie-Mellon University, Pittsburgh, PA, May, 1984.

[32] Wu, C.
The Kinematic Error Model for the Design of Robot Manipulators.
In *the Proceedings of the American Control Conference,* pages 497-502. San Francisco, CA, June, 1983.

Index