ROBOT CALIBRATION BASED ON NONLINEAR FORMULATION FOR MODULAR RECONFIGURABLE ROBOTS (MRRS)

by

Yu Lin
Bachelor of Mechanical Engineering
Fuzhou University
Fuzhou, P. R. China, 2005

A thesis
presented to Ryerson University
in partial fulfillment of the
requirement for the degree of
Master of Applied Science
in the Program of
Mechanical Engineering.

Toronto, Ontario, Canada, 2008

ABSTRACT

Robot Calibration Based on Nonlinear Formulation

for Modular Reconfigurable Robots (MRRs)

Yu Lin

A thesis for the degree of

Master of Applied Science, 2008

Department of Mechanical Engineering, Ryerson University

Developed in this thesis is a full pose kinematic calibration method for modular reconfigurable robots (MRRs). This method is based on a nonlinear formulation as opposed to the conventional linear method that has a number of critical limitations. By avoiding linearization of the nonlinear robot forward kinematic equations, these nonlinear equations are directly used to identify the robot calibration parameters. A hybrid search method is developed to solve the nonlinear error equations by combining genetic algorithms with Monte Carlo simulations to ensure a global search over the robot workspace with good accuracy. A number of comparisons are made between the proposed method and the conventional linear method, indicating the advantages of the former over the latter by eliminating two critical limitations. The first one is the orthogonality sacrifice that leads to ill-conditioning of the Jacobian used in the linear method. The second one is quadrant sensitivity during the determination of the (Tait) Bryan angles from inverting the rotation matrix.

ACKNOWLEDGMENTS

I am deeply indebted to my supervisor Dr. Jeff Xi, to whom I owe a debt of gratitude for the invaluable support, guidance and kindly encouragement throughout the past two-year study. With his strong academic background and innovative ideas, he has been really helpful to me for solving many difficulties whenever I needed help. It has been my pleasure to work under his supervision, and I look forward to working with him for the Ph.D. study in the future.

I would also like to thank NSERC and Engineering Services Inc. (ESI) for providing adequate research funding for me.

I also wish to thank my lab mates Mr. Richard P. Mohamed and Mr. Daniel Finistauri, who were also working on reconfigurable robots, for their advice when I was in need of an opinion. Furthermore, I would like to thank Mr. Haibin Jia, and Mr. Liang Liao, for their moral support over the past two years.

Finally, I would like to sincerely thank my family and close friends for their encouragement, support and understanding over the school years.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

Symbol	Definition	Units
x, y, z	x, y or z coordinate position	m
p, r, s	pitch, roll, and yaw angles	rad (or degree if specified)
$\boldsymbol{p}$	position vectors	m
$\boldsymbol{R}$	rotation matrix	-
$\boldsymbol{b}$	body vectors	m
$\boldsymbol{b}$`	local body vectors	m
$\boldsymbol{q}$	motion parameters (s and θ)	-
$\boldsymbol{s}$	linear translational displacement	m
$\boldsymbol{\theta}$	rotation angles	rad (or degree if specified)
$\boldsymbol{I}$	identity matrix	-
$\boldsymbol{J}$	Jacobian matrix	-
$\boldsymbol{g}$	geometry of a robot (**bs** and **Rs**)	m and rad
$\boldsymbol{X}$	end-effector pose of a robot	-
$\Delta\boldsymbol{g}$	robot kinematic parameters error	m and rad
$\Delta\boldsymbol{X}$	end-effector pose errors	-
f	function of forward kinematics	-
d	distance	m

Subscript	Definition
i	i^{th} module
n	total number of modules
ICSU	Initial configuration setup
s	Static
m	Motion
j	Joint
tip	Variables at the tip of module
min	Minimum
max	Maximum

Throughout the thesis, bold lower case indicates vector; bold upper case indicates matrix; and regular indicates scalar.

CHAPTER 1 INTRODUCTION

1.1 Background

In general, accuracy is defined by repeatability and bias [1]. Lack of repeatability is due to random error and quantified by the variance of a number of measurements. Bias is a systematic error and determined by the mean value [2]. Sometimes, they are distinguished as repeatability and absolute accuracy. Repeatability of a robot is the precision with which its end-effector achieves a particular pose (position and orientation) under repeated commands by the same set of joint variables, while "absolute accuracy" is the closeness to which the robot's actual pose matches the pose predicted by its controller [3].

A high repeatability is of prime importance for a variety of robot applications such as pick and place, spray painting, and welding. On the other hand, tasks involving off-line planning (OLP) depend on the absolute accuracy of the robot [3], especially for high precision applications, such as robotic surgery. In reality, while the repeatability can reach an order of 0.1 mm for the majority of today's robots, the absolute positioning accuracy is on the order of 1 mm or even worse, leading to the accuracy/repeatability ratio in the range from 3 to 20 [4]. Low accuracy of a robot is currently regarded as one of major obstacles for a wider range of applications.

While it is difficult to compensate for the random error, compensation for the systematic error can be done effectively by means of calibration [2]. Robot calibration has been explored and investigated since the 1980s. It evolved into a mainstream of robotics research area in the 1990's. At that time, the robot accuracy issue became crucial with many robots and off-line programming (OLP) software packages introduced into the world market. Calibration is even more important for a modular reconfigurable robot (MRR) since after each reconfiguration by re-assembling the modules, the MRR will lose its accuracy. Although variations in the kinematic model mostly arise from imprecision in manufacturing processes, the direct improvement of manufacturing processes is costly [5]. This is another reason to calibrate the robot after it is built.

Basically, a robot calibration technique is the process of improving robot positioning and orientation accuracy by identifying and then modifying the geometric parameters in robotic kinematic models rather than changing or altering the mechanical structures or redesigning the robot [6].

Since the sources of errors vary from one robot design to another, calibration procedures can vary in their scope and complexity. For example, some robot calibration procedures consider only the joint variables while others may involve changes in the robot kinematic model. Roth et al. [6] classified calibration into three levels: joint level, kinematic model

level, and dynamic model level. The goal of joint level calibration is to determine the correct relationship between the signals produced by joint displacement transducers and actual joint displacements. This usually involves calibration of the kinematics of the drives and the joint sensors. The kinematic model level calibration is the entire robot kinematic model calibration, and its purpose is to determine the correct geometric parameters of the robot. The dynamic model level calibration is non-geometric calibration, and its concern is about the effects such as joint compliance, friction and clearances, as well as link compliance.

The research reported in this thesis addresses the problem of kinematic model calibration considering both position and orientation. The purpose is the determination of an accurate relationship between the joint movements and the pose of the robot's end-effector. The assumption is that the robot is composed of rigid links and joints, ignoring joint backlash and servo errors.

In general, a calibration procedure can be divided into four steps: modeling, measurement, identification and implementation (or correction) [6]. Modeling refers to the choice of a functional relationship between the robot parameters and the pose of the robot's end-effector. Measurement is to collect the information about the inputs and the outputs. Identification is the determination of the errors that could affect calibration.

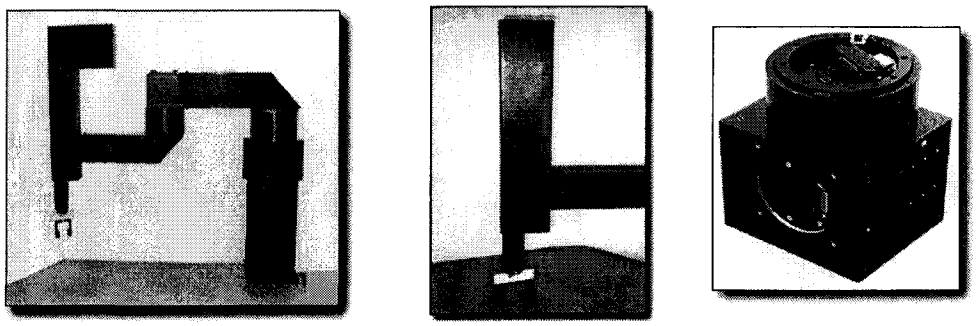
Implementation is to use the corrected model for robot kinematic control.

Mathematically, robot calibration can be considered as a nonlinear optimization problem, which can be solved in two different ways, linear and nonlinear formulations. The linear method is to linearize the nonlinear kinematic equations, which leads to a Jacobian matrix by ignoring higher order items. The Jacobian matrix, also called the mapping matrix, establishes a linear relationship between the errors of the kinematic parameters and the errors of the end-effector's pose. The main disadvantage of the linear approach is the singularity issue in inverting the Jacobian matrix. The nonlinear method, on the other hand, is to solve the nonlinear equations directly without linearization. Because the Jacobian matrix is not involved, the singularity issue can be avoided. In this thesis, both methods are studied and compared.

1.2 Modular Reconfigurable Robots (MRRs)

Traditional robots are fixed structure robots, meaning that a certain type of robot is built for a certain type of task. Reconfigurable robots are designed such that their structures can be changed to perform a number of different tasks that normally require a number of different types of traditional robots. A modular reconfigurable robot (MRR) is a reconfigurable robot that is built based on a number of modules. These modules can be rearranged through disconnecting and reconnecting in different ways to form a new

configuration enabling new functionalities. Figure 1.1 shows an example of a MRR made by a company called Engineering Service Inc. [8].



(a) (b) (c)

Figure 1.1: (a) Modular and reconfigurable robot (MRR-1);
(b) 2 DOF module (MRS) and (c) Rotary module (MRJ);
Photograph courtesy of Engineering Service Inc. [8].

Modular reconfigurable robots (MRRs) offer three main advantages over the traditional fixed-structure robots: versatility, simplicity, and low cost [9]. Firstly, applications in which MRRs surpass purpose-specialized robots include those in which versatility is critical, such as those requiring a variety of robots to perform multiple tasks of a similar nature. While traditional robots are only capable of performing a particular task for which they have been designed, MRRs offer the advantage of employing a single robot with the structural adaptability to perform multiple tasks based on demands by means of reconfiguration. Employing similar modules in a robotic arrangement leads to simplicity in design when compared with a traditional robot composed of various customized

components. Secondly, except for offering multi-purposeful tasks, MRRs enhance simplicity in the method of reconfiguration by employing a common locking mechanism for all joins and modules. Lastly, in addition to offering simplicity in design and versatility, modularity significantly reduces the design and manufacturing costs [10].

There are three categories of reconfigurable robots: self-assembly, self-configuring and manual-configuring [11]. Self-assembly robots are the robots with the highest level of reconfigurability because they are able to detach from and attach into a robotic system automatically. For example, the National Mechanical Engineering Laboratory in Japan developed a self-assembly robotic system that uses electro-magnetic disks as the basic units that can attract and repel each other through computer control for automatic reconfiguration [12]. Self-configuring robots cannot perform self-assembly. However, they can fulfill reconfiguration after a robotic system is assembled with some form of manual assistance. For example, robotic cubes were developed in the United Kingdom with an embedded active driving mechanism [13]. Once attached manually, these cubes can slide on each other's faces for reconfiguration. Since the cubes are made in different sizes and can be combined together, the robot is called the fractal shape-changing robot. The manual-configuring types are in fact the modular robots, as shown in Figure 1.1. They can only be reconfigured with some form of manual assistance. The modular units are built with embedded controllers, and the host computer has the capability to quickly

recognize new configurations and then achieve the objective of system control. This research work includes the studies at Stanford [14] and Carnegie Mellon University [15].

1.3 Robot Kinematics

To address the problem of robot calibration, the basis is the robot forward kinematics.

A significant number of papers and monographs have been published on different kinematic modeling approaches for calibration [5] [16] [17] [18] [19] [20] [21] [22]. A satisfactory kinematic model should exhibit three qualities: completeness, equivalence, and proportionality [19]. According to [18], the end effector's pose of a robot can be defined by a nonlinear function as follows

$$\mathbf{X} = \mathrm{f}(\mathbf{q}, \mathbf{g}),\ \mathbf{q}_i = [\boldsymbol{\theta}_i \quad \mathbf{s}_i],\ i=1,...,n \tag{1.1}$$

where $\mathbf{q}$ is the set of joint variables including rotational angles $\boldsymbol{\theta}_i$ for revolute joints and translational displacements $\mathbf{s}_i$ for prismatic joints, with i ranging from 1 to n, (n is the total number of joints); and $\mathbf{g}$ is the set of geometric parameters of the robot.

The end-effector's pose can be further detailed as the position vector and orientation matrix, described in the global reference frame as [18],

$$\mathbf{p}_{n+1} = \sum_{i=1}^{n} \mathbf{R}_{0i} \mathbf{b}_i' , \tag{1.2}$$

$$\mathbf{R}_{0n} = \prod_{i=1}^{n} \mathbf{R}_{(i-1)i} , \tag{1.3}$$

where $\mathbf{b}'$ is the body vector in the local coordinates, which is transformed to the global first and then summed to obtain the position of the end-effector in Equation (1.2); $\mathbf{R}_{(i-1)i}$ represents the rotation matrix between two adjacent bodies, which is sequentially multiplied together to determine the orientation of the end-effector in Equation (1.3).

Since only three of the nine elements of the rotation matrix are independent, the robot orientation is usually specified with three independent parameters. The common approach is to use three angles, such as Euler angles and the (Tait) Bryan angles (often called pitch, roll, and yaw, (PRY)). The angle-axis representation has also been introduced by using a unit vector parallel to the axis of rotation matrix in Equation (1.3) and an angle rotating about the axis. However, this invariant vector may correspond to two possible rotation matrices [21] [23]. To solve this ambiguity, Angeles [23] modified this angle-axis representation and showed that the modified one has a linearity relation with the corresponding rotation matrix, hence leading to simple expressions of the Jacobian matrix.

1.4 Problem Formulation

Due to a number of error sources, including manufacturing, assembly, deflections, measurements, and clearances [24], the robot forward kinematic model (Equation (1.1)) will not accurately represent the real robot system. In other words, the manipulator will

not be able to locate the end-effector at the desired pose $\mathbf{X}_o$ calculated from the nominal model given in Equation (1.1). The actual pose $\mathbf{X}_m$ is different, as illustrated in Figure 1.2. The difference between the two is the end-effector pose error. In order to determine the end-effector pose error $\Delta\mathbf{X}$, the actual pose $\mathbf{X}_m$ usually needs to be measured after moving the robot by joint variables $\mathbf{q}_o$. It is given as

$$\Delta\mathbf{X} = \mathbf{X}_o - \mathbf{X}_m \tag{1.4}$$

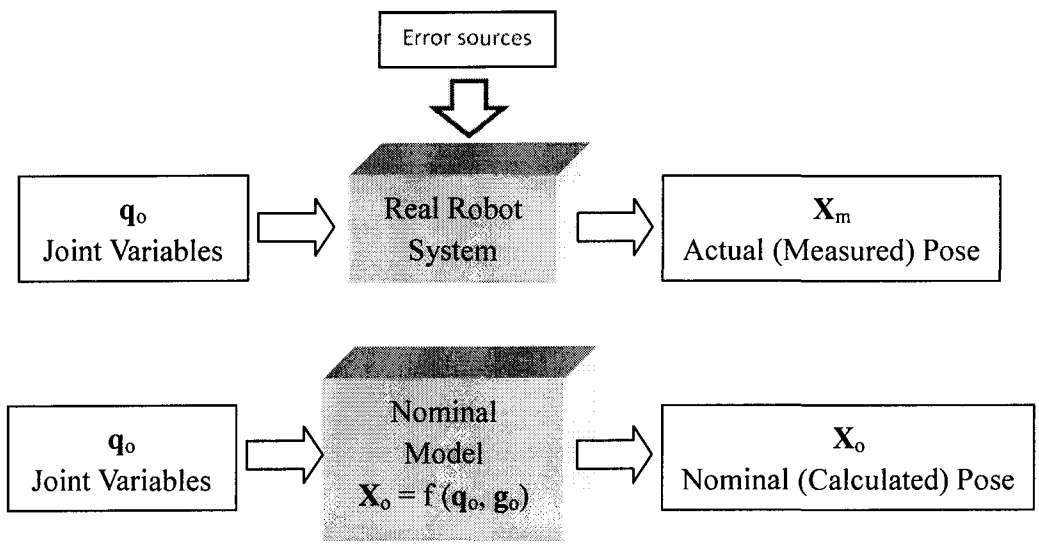


Figure 1.2: Actual (measured) and nominal (calculated) poses.

Apparently, the basic consideration is how to eliminate or compensate the pose error in Equation (1.4) in order to reach the desired pose. As one of the solutions, calibration technique aims at identifying the geometric errors $\Delta\mathbf{g}$ of the robot, and then make a correction in the nominal model, i.e., $\mathbf{X}_o = f(\mathbf{q}_o, \mathbf{g}_o+\Delta\mathbf{g})$. As a result, the nominal pose of

the end effector will be getting closer to the measured ones under the corrected model, i.e., $\mathbf{X}_o \rightarrow \mathbf{X}_m$, as shown in the Figure 1.3. A well calibrated robot will be able to reach any position within its workspace at the required accuracy.

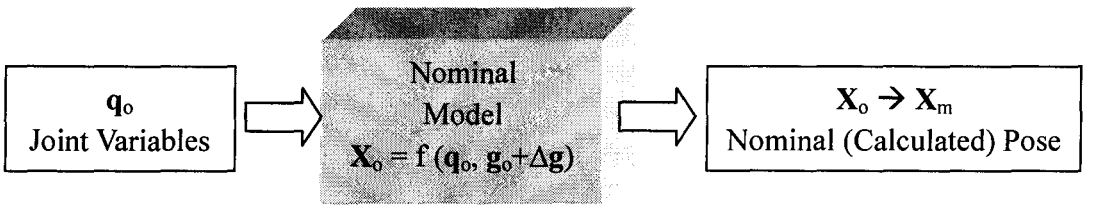


Figure 1.3: Nominal pose after calibration.

As an alternative approach, we could adjust joint movements by adding $\Delta\mathbf{q}$ to the original joint variables, i.e.,

$$\mathbf{X}_o = f(\mathbf{q}_o+\Delta\mathbf{q}, \mathbf{g}_o) \rightarrow \mathbf{X}_m. \tag{1.5}$$

This is called compensation, rather than the calibration focused on in this thesis. The details of this approach are outside the scope of this thesis.

As mentioned above, two methodologies are available to determine $\Delta\mathbf{g}$, namely, linear and nonlinear formulations [25]. The linear least squares method is used for the linearized error model as given below:

$$\Delta\mathbf{X} = \mathbf{J}\,\Delta\mathbf{g}. \tag{1.6}$$

From equation (1.6), $\Delta\mathbf{g}$ can be solved for by a proper linear least-square method depending on the dimensions of the Jacobian matrix. This process is usually carried out

iteratively until the tolerance of $\Delta \mathbf{X}$ becomes less than a given threshold. The second approach is the nonlinear formulation given as below:

$$\Delta \mathbf{X}(\Delta \mathbf{q}) = \mathbf{X}_m - f(\mathbf{q}_o, \mathbf{g}_o + \Delta \mathbf{g}) \rightarrow 0. \tag{1.7}$$

Two types of optimization methods can be applied to solve equation (1.7): gradient-based methods and direct-search methods. For gradient-based methods, the identification Jacobian is again required in a manner that differs from that of the linear least-square algorithm [26]. The Levenberg-Marquardt algorithm is one of the popular methods and works surprisingly well even for large residual problems, although in such cases the rate of convergence may be quite slow [17]. This technique is designed to overcome problems related to singularity of the matrix $\mathbf{J}^T\mathbf{J}$ by adding a time varying nonnegative scalar coefficient, μ, leading to the matrix $\mathbf{J}^T\mathbf{J} + \mu \mathbf{I}$. More details can be found in Scales [27]. Alternatively, equation (1.7) can be minimized using direct-search methods, such as Simplex search, Hooke-Jeeves pattern search, Powell's conjugate-direction method [28], genetic algorithm [61] [62] and neural networks [63]. Convergence of all direct search algorithms is generally very slow, but no Jacobians are needed. The linear method was the only possible way to conduct the robot calibration before, and now the nonlinear method becomes possible due to the advent of powerful computers.

1.5 Outline of Thesis

The remaining thesis is organized as follows.

Chapter 2 provides a literature review on previous researches on the robot calibration, including modeling, measurement, identification, and correction.

Chapter 3 presents a kinematic modeling method for MRRs. Pose analysis is introduced using six-parameter representations.

Chapter 4 discusses the conventional linear formulation for robot calibration. Error models are derived from linearization of the forward kinematic model in two schemes, depending on how the orientation of an end-effector is measured.

Chapter 5 investigates the nonlinear formulation without linearizing the kinematic model. A genetic algorithm (GA) is applied to solve this problem. Furthermore, Monte Carlo simulations are conducted to take into consideration of the random error by running GA.

Chapter 6 provides the simulations and comparisons between the linear and nonlinear formulation of calibration.

Chapter 7 applies the proposed calibration methods to MRR robots in consideration of reconfiguration.

Chapter 8 provides the summary of conclusions, contributions, and future work.

CHAPTER 2 LITERATURE REVIEW

This chapter describes a literature review on previous researches that have been done on robot kinematic calibration, including kinematic modeling, measurement, identification, and correction.

2.1 Kinematic Modeling for Calibration

A significant number of methods have been proposed for modeling robot kinematics for the purpose of calibration since the 1980s. For serial robots, joints are either revolute or prismatic. Three important qualities that a satisfactory kinematic model should exhibit are completeness, equivalence, and proportionality [19]. First, the model should contain a sufficient number of parameters to completely satisfy the motion of the robot. Second, there must exist a functional relationship between any two acceptable models, i.e., models should be equivalent in a functional sense. Finally, small changes in the robot geometry should reflect small variations in the model parameters.

The most common method for robot kinematic representation is the use of four Denavit and Hartenberg parameters [29], including link length a_i, joint twist α_i, link offsets d_i, and joint offsets θ_i, as shown in Figure 2.1.

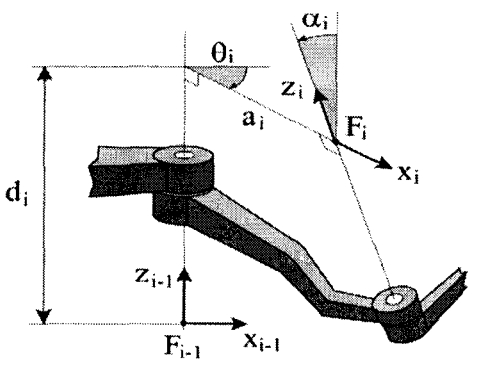


Figure 2.1: Link frames attached and four DH parameters [30].

Paul [16] demonstrated the use of the Denavit-Hartenberg technique for modeling a serial link manipulator. For each link, a coordinate frame is attached at the joint, and then related consecutively by the four parameters using a homogeneous transformation matrix. All these consecutive link transformation matrices are multiplied to produce a total transformation matrix that relates the last local coordinate system attached to the end effector to the base global coordinate system. The resulting matrix is a nonlinear function of the joint variables (θ_i) and three link parameters (a_i α_i d_i) that describe the geometry of the manipulator. Furthermore, the Denavit-Hartenberg model has been used for the calibration problem by many investigators [5]. For example, Wu [31] [32] used it to examine robots with small variations in their kinematic arrangement.

However, the Denavit-Hartenberg representation was found to exhibit ambiguity when two successive axes are parallel as pointed out by Mooring [33] and Hayati [22]. The link

length a_i, representing the length of common normal between two consecutive joint axes, varies wildly when two adjacent joint axes deviate slightly from nominally parallel configuration.

For this reason, the Denavit-Hartenberg model was modified. In the nearly-parallel axes case, the parameters are treated to vary in proportion to the degree of misalignment, as proposed by Hsu and Everett [34]. Ibarra and Perreira [35] modified the Denavit-Hartenberg based transformation matrix by a differential screw matrix for small misalignments. Hayati [22] pointed out the singularity of the Denavit-Hartenberg representation, and introduced an additional rotation β in the y-direction in the DH link transformation to describe a misalignment of two consecutive parallel axes. A number of papers subsequently used this modified Denavit-Hartenberg representation as reviewed in Hollerbach [5]. The most recent application can be found in Gatla et al. [36] and Lightca et al. [37]. Gatla et al. [36] combined Craig's modified DH [21] and Hayati's (HR) [22] together to build either DH or HR with the four-parameter depending on whether the neighboring joint axes are parallel. This model, however, requires two sets of four link error parameters according to the geometry of link, which makes the modeling task unnecessarily complex. Furthermore, Hayati's model does not cover the transformation from the last link to the tool coordinate frame which in general requires a separate treatment [38].

Although the description of the robot kinematics needs at most four parameters per joint, this minimal set of parameters displays proportionality only when the selection of the joint coordinate frames is tailored to a particular manipulator [20]. Thus, a priori knowledge of the nominal manipulator geometry is required for any algorithm that uses such a mode. On the other hand, it is possible to construct a six-parameter model that does not depend on the link geometry and will always display proportionality. As a result, a number of papers have abandoned the Denavit-Hartenberg parameters entirely and treated the general case of two coordinate systems related by six parameters [5]. Three parameters present the coordinate origin displacements, and three parameters for relative coordinate system orientation. Papers differ in terms of the representation of orientation; some use Euler angles, introduced in Chapter 3, and others use variations of the Euler vector, namely, associated rotation about an equivalent vector, detailed in Craig [21].

Mooring [33] also pointed out the problem with the Denavit-Hartenberg representation at the same time as Hayati [22], but proposed a six-parameter representation using Euler vectors. An estimation procedure was outlined but not implemented, in which single joints were moved to two positions. The position and orientation of the gripper were measured at these two positions, and the six parameters for each joint were determined by a direct solution. Mooring and Tang [39] modified the procedure to use three points in a fixture, avoiding the need to determine the gripper position and orientation by external

sensing. On the other hand, six-parameters with Euler angles were used [40], and both geometric and non-geometric parameters were modeled. Chen et al. [41] also employed six-parameter transformation with Euler angles. Later, Chen and Chao [42] extended this work to include non-geometric models for gravitational joint deflection and backlash. Stone et al. [43] used a general-purpose "S-model" to model kinematic errors using six parameters per link and then convert them to the Denavit and Hartenberg parameters.

2.2 Computation for Calibration

As mentioned before, identification of parameters of the robot kinematic model is generally considered as a nonlinear optimization problem, which can be approached in two ways, namely linear or nonlinear [25]. Many investigators linearized the kinematic equations leading to a linear error model and then applied an iterative least-square estimation procedure, while others applied other nonlinear estimation procedures directly to the nonlinear kinematic equations.

A recursive least squares procedure was employed by Chen et al. [41] to the linearized equations. Chen and Chao [42] suggested linearization as a means for indentifying dependent parameters. At the same time, Chao and Yang [44] applied the nonlinear Levenberg-Marquardt procedure to the same data used by Chao and Chao [42], and achieved identical results.

An interesting insight of Mooring and Tang's work [39] was to estimate orientation parameters before position parameters. While the former leads to a nonlinear estimation problem, the latter can then be treated as a linear estimation problem. For the orientation errors, the sum of squares of Euler angle differences is minimized by a finite-difference Levenberg-Marquardt algorithm, which is commonly employed for nonlinear estimation problems. The estimated orientation parameters are then incorporated to apply direct linear estimation to the position parameters. However, this method did not consider the effect of the angular parameter errors on position errors. In other words, it ignored the coupling terms between orientation and position errors in the Jacobian matrix and derived two simpler and separable least squares problems. Simulations indicate better results could be obtained by including the full pose Jacobian of both orientation and position [5].

Lightcap et al. [37] proposed a two-level Levenberg-Marquardt nonlinear least-square optimization algorithm for the calibration of geometric and flexibility parameters in a serial manipulator without the computation of the generalized Jacobian matrix. Link parameters were determined through an outer optimization loop, while robot coordinate system parameters were determined in an inner loop.

Pathre and Driels [25] conducted simulations that have shown that the linear method is four to eight times faster than the nonlinear method. However, nonlinear method, for

example the Levenberg-Marquardt algorithm, is much more robust than linear method especially for large parameters.

One of the important problems in calibration is the observation strategy that refers to the selection of robot configurations and the number of observations to be made during the calibration experiment. The selection of measurement configurations during robot calibration plays an important role in determining the accuracy and speed of convergence of the least-square identification algorithms [17]. The number of necessary observations can be reduced if the measurements are performed at robot configurations in which the error model is the most sensitive to changes in model parameters, e.g., where it is the least sensitive to unmodeled error sources [20]. Simulation experiments can significantly help in evaluation of the observation strategy [25]. This strategy may be modified based on the results of simulations if necessary. Two groups of references provide the literature background for the observation strategy. The first approach [45] focuses on the familiar numerical analysis concept of "condition number" of the Jacobian. The second one [46] [47] adopts an observability index as a performance measure. Driels and Pathre [45] have shown that the condition number of the Jacobian is related to the observability index proposed by Menq and Borm [46]. Increase in the observability index implies the reduction of the condition number.

2.3 Measurement for Calibration

Measurement is an essential part of the calibration procedure. It is conducted by measuring the actual robot end effector's position and orientation at given set of joint displacements. A set of measurement data is obtained by moving the robot to one location within the workspace, recording joint displacements, and then using an external measuring system to determine the robot's position and/or orientation. The robot is then moved to another location to repeat the process and continue till sufficient data is acquired.

There are two aspects of the measurement process that need to be given careful consideration. The first is what measurement system should be used, and the second is how to plan the observation strategy correctly. There are only a few systems that have the necessary precision to make adequate pose or partial pose measurements. Each has its own characteristics such as precision, speed and ease of use, level of measurement noise, cost, and the amount of information that can be obtained from each robot pose. In general, the measurement process is time consuming, laborious, and prone to human error. Textbooks, such as the one by Doebelin [48], provide a much more exhaustive treatment of this field.

2.3.1 Noncontact Measurement Technology

It is desirable that the measuring instrumentation facilitates non-contact sensing, so that

the influence of the measurement to the robot performance characteristics is eliminated [20]. The most popular technique of the non-contact measurement is based on utilizing a system of theodolites (Duelen and Schroer [49]; Judd and Knasinski [50]; Caenen and Angue [51]), which facilitates measurements in a wide range and with high accuracy on the order of 0.05 mm [52]. By combining the theodolites with a low-resolution vision system for automatic tracking, focusing and centering, it is possible to build a high-quality system for automatic calibration measurements [53].

Researchers have also been investigating camera measurement, which basically has two different setups: a moving camera approach (hand-mounted cameras) and a stationary camera setup. In the moving camera approach, one or more cameras are mounted on the end-effector of a robot and several targets or fixtures are fixed in the workspace. In the stationary setup, cameras are fixed and the fixture or targets are mounted on the robot end-effector.

Generally speaking, high resolution and large field-of-view may be two conflicting requirements [38]. The stationary camera setup suffers from this conflict and has to sacrifice measurement accuracy in order to have a large field-of-view, or the cost of system may increase dramatically by using higher resolution camera. The moving camera approach solves this conflict by using a precise fixture and a stereo camera system. It

means that the robot can move to a wide range of configurations and still has a calibration which is as accurate as desired.

Zhuang [38] applied a stereo hand-eye system consisting of a pair of CCD cameras mounted on the robot's end-effector, a camera calibration board, a robot calibration fixture and a PC-based image processing system (Figure 2.2a). Without using expensive or labor-intensive equipment, such as theodolites, laser tracking system, high-resolution opto-camera systems etc., Zhuang [38] stated that the mobile camera system provides low-cost, efficient and fully automated features which are suitable for academic research as well as industrial applications. On the other side, An et al. [54] conducted calibration experiments by using systems of immobile cameras equipped with optoelectronic detectors, and additional LED diodes as targets fixed at the robot's end-effector (Figure 2.2b). This technique may also yield satisfactory results, but in a significantly smaller workspace.

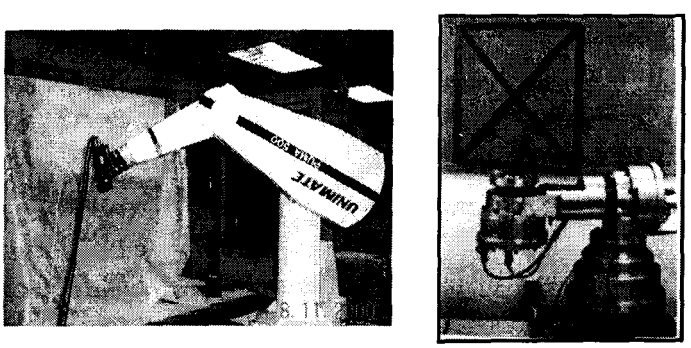


Figure 2.2: (a) Moving camera setup [38]; (b) Stationary camera setup [54].

A stationary camera setup provides a non-invasive method, where the camera is often placed outside the robot workspace and needs not be removed after calibration. However, it is necessarily invasive for the moving camera approach, where the camera has to be removed after calibration. Another disadvantage for the moving camera approach is that it only performs local measurements, whereas the global information on the robot end-effector pose is provided through a stationary calibration fixture. For the stationary camera setup, there is no need to identify the transformation relating the camera frame to the end-effector frame.

2.3.2 Contact Measurement Technology

Contact measurements vary from coordinate measuring machines (CMM), dial indicator, linear-variable differential transformer or LVDT, precisely located targets with force sensors, to a simple ruler etc. A CMM assures extremely high accuracy on the order of 0.01 mm, compared with the other less accurate but less expensive contact measuring techniques.

The three-point method has been utilized commonly in various measuring systems in the past, such as Mooring et al. [17], Lightcap et al. [37], Goswami et al. [3], and Xi et al. [55]. An orientation change can be determined from the initial and final three-point positions, as detailed in Chapter 3.

Lightcap et al. [37] used a CMM to measure the locations of three tooling-ball apparatus attached to PA 10-6CE robot (Figure 2.3), to determine the transformation between the CMM coordinate system and a user-defined tooling ball coordinate system. External loads were lowered onto the weight rack and transmitted through the end-effector of the robot directly to avoid the deflection in the rods.

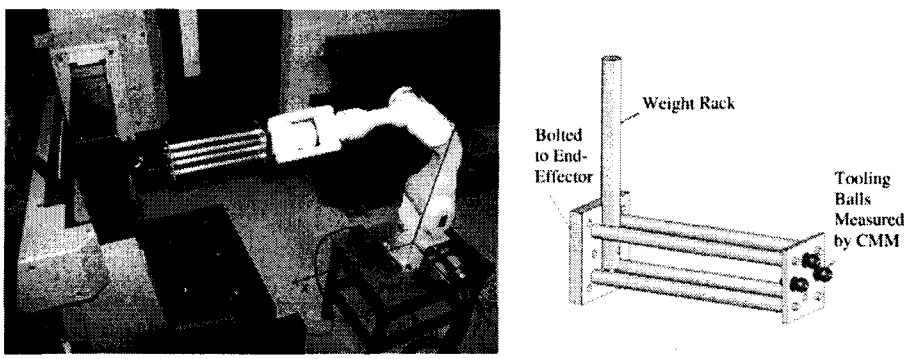


Figure 2.3: Tooling-ball apparatus attached to a robot and CMM [37].

Goswami et al. [3] created a telescopic ball-bar measurement (Figure 2.4) for the calibration of PUMA 560 using the Stewart platform analogy. The ball-bar system consists of a single LVDT, which is connected between one steel sphere attached to the robot endpoint and a magnetic chuck mounted on the table. Imagine that the robot endpoint triangle (with three steel spheres) and the base triangle (with three magnetic chucks) are interconnected through six ball-bars, which presents the Stewart platform geometry.

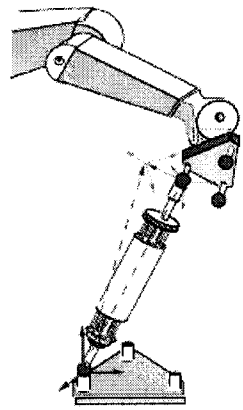


Figure 2.4: The telescopic ball-bar (LVDT) measuring system [3].

Mooring et al. [17] proposed a three-point moving target and a LVDT measuring fixture which is mounted accurately in the workspace of a robot. A set of three spheres whose relative positions are already known, can be inserted in an array of short-range displacement transducers (Figure 2.5). In this case, these transducers are LVDTs but capacitance probes and dial indicators have also been reported.

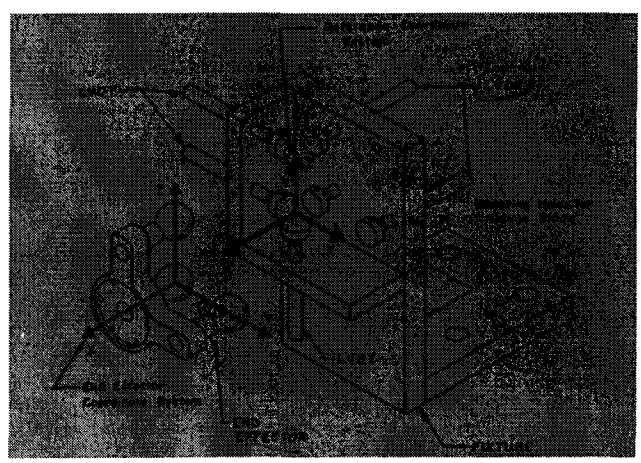


Figure 2.5: Three-point moving target and measurement (LVDT) fixture [17].

2.4 Self-calibration

Without using external measuring devices introduced in the previous section, self-calibration methods depend on internal measurements, such as joint angle measurements. Self-calibration is more desirable or practical on a manufacturing floor or in a production line where external measurements are expensive and difficult to implement.

The existing techniques of serial robot calibration can be classified into open-loop and closed-loop approaches [36]. Open-loop methods involve measuring the end-effector pose, which requires measuring equipments, such as theodolites, ball-bar, CMMs, laser tracking system, cameras, etc., which can be found in the last section. On the other hand, closed-loop methods use the internal joint angle measurements already in the robot without external measurement devices, and therefore, can be considered self-calibrating. Usually, these methods impose some constraints on the end-effector, and the joint readings alone are used to calibrate the robot using kinematic closed-loop equations.

Some researchers in the past have applied linear constraints on the end-effector positions allowing the end effector to slide along a line. For example, Neqman et al. [56] used a laser line. Ikis et al. [57] and Zhuang et al. [58] imposed plane constraints on the end-effector positions. However, it may be problematic to use a plane constraint since it is difficult to assure that the end-effector is exactly on the surface.

One of recent proposed close-loop methods is the "virtual closed kinematic chain" by Gatla et al. [36], which did not require any physical constraints used in the previous closed-loop methods. A laser pointer tool was applied on the end-effector to aim at a fixed location on a distant object (Figure 2.6), and only joint readings were used to calibrate the robot. The laser tool on the robot acts as a virtual telescopic (prismatic) link giving the robot 7 DOFs, the seventh joint being the length of the laser line from the end effector to the projected laser point on an object. Thus, aiming the laser pointer at a fixed point creates a virtual closed kinematic chain. The main advantage of this method is that the distant laser point is very sensitive to joint values, which facilitates acquiring more accurate joint values for the calibration.

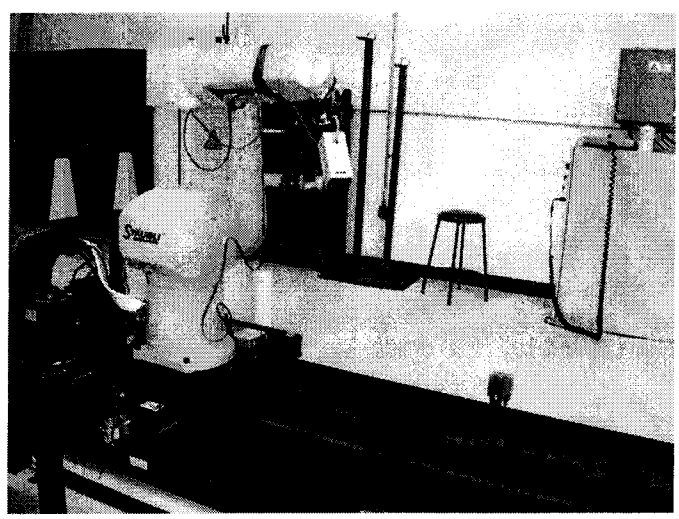


Figure 2.6: A laser pointer tool carried by Staubli RX-130 robot [36].

CHAPTER 3 KINEMATIC MODELING

This chapter presents the forward kinematic modeling of a modular reconfigurable robot with rigid modules. In an error-model-based kinematic calibration process, the selection of a proper kinematic model is one of the keys to the success of a calibration task. In this chapter, six-parameter representations, rather than the Denavit-Hartenberg parameters, are used for the robot kinematic modeling.

The main topic of the robot forward kinematics is to compute the position and orientation of the end-effector relative to the base coordinate as a function of the joint variables. As shown in Figure 3.1, an MRR system under study can be considered as a multiple module system. The pose of each module is represented using six parameters, three for position and three for orientation. The position vector and rotation matrix are first introduced in this chapter and then the general motion (translation and rotation) of a single module and a multi-module system are discussed along with the three-point measurement methods.

3.1 Position and Orientation

In this section, the position vector and rotation matrix are introduced to represent the pose of each module.

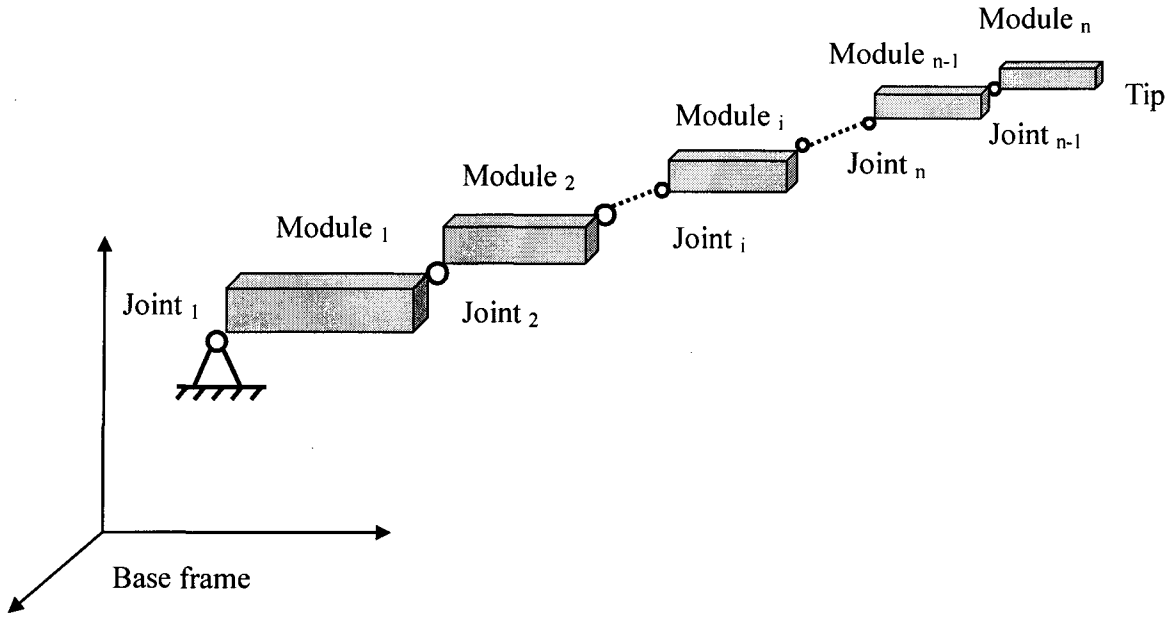


Figure 3.1: Multiple modules system for a MRR.

3.1.1 Position Vector

The position of a point in space is represented with respect to a coordinate frame using a vector. In general, the vector components in the Cartesian coordinate are expressed as

$$\mathbf{p} = [p_1, p_2, p_3]^T . \tag{3.1}$$

A position vector, as shown in Figure 3.2, can be expressed in terms of the frame axes in linear combination as

$$\mathbf{p}=p_1\mathbf{e}_1+ p_1\mathbf{e}_1+ p_1\mathbf{e}_1 \tag{3.2a}$$

or

$$\mathbf{p}=\mathbf{E}p \tag{3.2b}$$

where $\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$. In the Cartesian coordinate system, $\mathbf{E} = [\mathbf{x}, \mathbf{y}, \mathbf{z}]$, and $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are the

unit vectors along x, y, and z axes relatively, that is, $\mathbf{x} = [1, 0, 0]^T$, $\mathbf{y} = [0, 1, 0]^T$, $\mathbf{z} = [0, 0, 1]^T$.

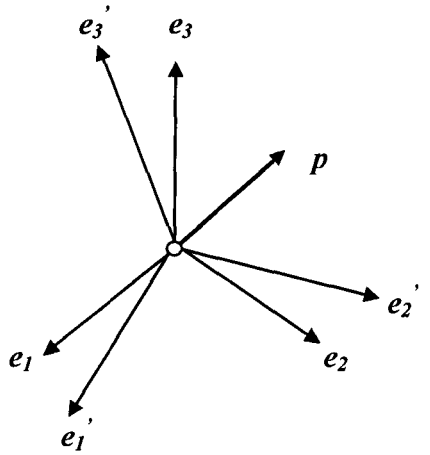


Figure 3.2: Position vector **p** [18].

3.1.2 Rotation Matrix

A rotation matrix represents a linear transformation between two coordinate frames. In Figure 3.2, position vectors can be expressed either in frame $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ (Equation 3.2a) or frame $\{\mathbf{e'}_1, \mathbf{e'}_2, \mathbf{e'}_3\}$,

$$\mathbf{p}=p'_1\mathbf{e'}_1+p'_2\mathbf{e'}_2+p'_3\mathbf{e'}_3 \tag{3.3a}$$

or

$$\mathbf{p}=\mathbf{E'}\boldsymbol{p'} . \tag{3.3b}$$

Since Equations (3.2) and (3.3) represent the same position vector, so

$$\mathbf{E'}\boldsymbol{p'}=\mathbf{E}\boldsymbol{p} \tag{3.4}$$

It leads to

$$p = \mathbf{R}p' \tag{3.5}$$

where **R** is the rotation matrix given as

$$\mathbf{R} = \mathbf{E}^{\mathrm{T}} \cdot \mathbf{E}' \tag{3.6}$$

Since $\mathbf{e}_i$ is orthogonal to $\mathbf{e}_j$, then $\mathbf{e}_i \cdot \mathbf{e}_j = \delta_{ij} = 1$ when i=j; $\mathbf{e}_i \cdot \mathbf{e}_j = 0$ when i≠j. Hence, **E** is orthogonal, and $\mathbf{E}^{-1} = \mathbf{E}^{\mathrm{T}}$.

R is in fact defined by the dot product of two unit vectors, i.e., the direction cosine. It is also called the tensor product, defined as

$$\mathbf{R} = (\mathbf{E} \otimes \mathbf{E}') = \begin{bmatrix} \mathbf{e}_1 \cdot \mathbf{e}'_1 & \mathbf{e}_1 \cdot \mathbf{e}'_2 & \mathbf{e}_1 \cdot \mathbf{e}'_3 \\ \mathbf{e}_2 \cdot \mathbf{e}'_1 & \mathbf{e}_2 \cdot \mathbf{e}'_2 & \mathbf{e}_2 \cdot \mathbf{e}'_3 \\ \mathbf{e}_3 \cdot \mathbf{e}'_1 & \mathbf{e}_3 \cdot \mathbf{e}'_2 & \mathbf{e}_3 \cdot \mathbf{e}'_3 \end{bmatrix} \tag{3.7}$$

Reversing the order of Equation (3.4), Equation (3.5) becomes

$$\mathbf{p}' = \mathbf{R}'\mathbf{p} \tag{3.8}$$

where

$$\mathbf{R}' = \mathbf{E}'^{\mathrm{T}} \cdot \mathbf{E} \tag{3.9}$$

Obviously

$$\mathbf{R}' = \mathbf{R}^{\mathrm{T}} = \mathbf{R}^{-1} \tag{3.10}$$

Hence, **R** is orthogonal, and such that all columns are mutually orthogonal and have unit magnitude. In fact, it is proper orthogonal, meaning det(**R**) = 1.

It is clear that the nine elements of a rotation matrix are not all independent. Six

dependencies or constraints between the elements can be easily found from a given rotation matrix, $\mathbf{R} = \left[\bar{\mathbf{X}} \quad \bar{\mathbf{Y}} \quad \bar{\mathbf{Z}}\right]$:

$$\begin{aligned} \left|\bar{\mathbf{X}}\right| &= 1 \\ \left|\bar{\mathbf{Y}}\right| &= 1 \\ \left|\bar{\mathbf{Z}}\right| &= 1 \end{aligned} \tag{3.11a}$$

$$\begin{aligned} \bar{\mathbf{X}} \cdot \bar{\mathbf{Y}} &= 0 \\ \bar{\mathbf{X}} \cdot \bar{\mathbf{Z}} &= 0 \\ \bar{\mathbf{Y}} \cdot \bar{\mathbf{Z}} &= 0 \end{aligned} \tag{3.11b}$$

As a result, three independent parameters representation is developed in the following section in order to express the rotation matrix conveniently, using the angle-set convention.

3.1.3 Angle-set Representation of a Rotation

There are basically two methods to describe the orientation of a frame relative to a reference frame by angle-set conventions. According to Craig [21], one is Euler angles, and one is fixed angles. In the former representation, each rotation is performed about an axis of the moving coordinate system rather than one of the fixed reference frame. For the latter one, each of the three rotations takes place about an axis in the fixed reference frame.

For each method, 12 sets of conventions are employed according to different sequence of rotation about the X-Y-Z axes. One of them is introduced in details in this section, and the

rest can be found in [21]. Usually, there is no particular reason to favor one convention over another, but various authors adopt different ones [21]. The following PRY angles (or X-Y-Z Euler angles) are applied in this thesis.

(Tait) Bryan Angles (Pitch Roll Yaw, PRY) [or X-Y-Z Euler angles]

In terms of pitch, roll, and yaw angle (PRY) [18], the three individual rotation matrices can be given as:

$$\mathbf{R}(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \tag{3.12}$$

$$\mathbf{R}(\theta_y) = \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \tag{3.13}$$

$$\mathbf{R}(\theta_z) = \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

Then, the resulting rotation matrix in the global reference frame is given as:

$$\mathbf{R} = \mathbf{R}(\theta_x)\mathbf{R}(\theta_y)\mathbf{R}(\theta_z) \tag{3.15}$$

If the order is reversed, it will become the rotation matrix in the local frame

$$\mathbf{R}^T = \mathbf{R}^T(\theta_z)\mathbf{R}^T(\theta_y)\mathbf{R}^T(\theta_x) \tag{3.16}$$

Expanding Equation (3.15) leads to

$$\mathbf{R} = \begin{bmatrix} c\theta_y c\theta_z & -c\theta_y s\theta_z & s\theta_y \\ s\theta_x s\theta_y c\theta_z + c\theta_x s\theta_z & -s\theta_x s\theta_y s\theta_z + c\theta_x c\theta_z & -s\theta_x c\theta_y \\ -c\theta_x s\theta_y c\theta_z + s\theta_x s\theta_z & c\theta_x s\theta_y s\theta_z + s\theta_x c\theta_z & c\theta_x c\theta_y \end{bmatrix} \tag{3.17}$$

When given three PRY angles, Equation (3.17) can be used to compute the rotation matrix directly. For the reverse problem, some elements in the Equation (3.17) are selected to determine the PRY angles for a given a rotation matrix, for example:

$$\begin{aligned} \theta_y &= \cos^{-1}\left(\left[\left(r_{23}\right)^2 + \left(r_{33}\right)^2\right]^{1/2}\right) \\ \theta_x &= \cos^{-1}\left(r_{33}/\cos\theta_y\right) \\ \theta_z &= \cos^{-1}\left(r_{11}/\cos\theta_y\right) \end{aligned} \tag{3.18}$$

where $r_{23} = -\sin(\theta_x)\cos(\theta_y)$, $r_{33} = \cos(\theta_x)\cos(\theta_y)$, $r_{11} = \cos(\theta_y)\cos(\theta_z)$.

Different selection of elements from the given rotation matrix leads to different method for a solution. Bai and Teo [59] developed another solution using atan2(y, x), a two-argument arctangent function that uses the signs of both x and y to identify the quadrant in which the resulting angle lies:

$$\begin{aligned} \theta_x &= \text{atan2}\left(-r_{23}, r_{33}\right) \\ \theta_y &= \text{atan2}(r_{13}, -r_{23}\sin(\theta_x) + r_{33}\cos(\theta_x)) \\ \theta_z &= \text{atan2}\left(r_{21}\cos(\theta_x) + r_{31}\sin(\theta_x), r_{22}\cos(\theta_x) + r_{32}\sin(\theta_x)\right) \end{aligned} \tag{3.19}$$

However, none of them is capable of solving this inverse problem with a unique solution

from a rotation matrix in all four quadrants, which is called the quadrant sensitivity problem. Equation (3.18) is only valid for $0 \leq \theta y \geq 90°$, $0 \leq \theta x \geq 180°$, and $0 \leq \theta z \geq 180°$; while Equation (3.19) is valid when θy lies in the first and fourth quadrants.

In other words, if θy is located in the second and third quadrants, the values of $\cos(\theta y)$ would become negative. As a result, the two elements $r_{23} = -\sin(\theta_x)\cos(\theta_y)$ and $r_{33}=\cos(\theta_x)\cos(\theta_y)$ in Equation (3.19) change signs, and the values of θx determined from $\text{atan2}(r_{23}, r_{33})$ are no longer true. It is clearly found from Table (3.1) that there is always 180 degrees offset between the determined values and true ones in this case.

So by modifying Equation (3.19),

$$\cos\theta_y = \sqrt{r_{23}^2 + r_{33}^2} = \sqrt{(-s\theta_x c\theta_y)^2 + (c\theta_x c\theta_y)^2} = \sqrt{c\theta_y^2}\,, \tag{3.20}$$

then θy can also be determined by

$$\theta_y = \text{atan2}\,(r_{13}, \sqrt{r_{23}^2 + r_{33}^2})\,. \tag{3.21}$$

Even though a second solution exists, by using the positive square root in Equation (3.20) for θy, we always can compute the single solution for which $-90^\circ \leq \theta y \geq 90^\circ$, making it a one-by-one mapping orientation representation.

Attention should also be paid to the names pitch, roll and yaw angles, since they are often given to other related but different angle-set conventions; for instance, it is referred to the X-Y-Z fixed angles in Craig [21].

Table 3.1: PRY angles inversed from a rotation matrix,
θx, θy, θz = 45, 135, 225, 315, stands for the full four quadrants,
x(-135, 45, -135) is the calculated value.

	θz = 45	135	225	315
		θx = 45		
θy = 45	√	√	√	√
135	x(-135, 45, -135)	x (-135, 45, -45)	x (-135, 45, 45)	x(-135, 45, 135)
225	x(-135, -45, -135)	x (-135, -45, -45)	x (-135, -45, 45)	x(-135, -45, 135)
315	√	√	√	√
		θx = 135		
θy = 45	√	√	√	√
135	x(-45, 45, -135)	x (-45, 45, -45)	x (-45, 45, 45)	x(-45, 45, 135)
225	x(-45, -45, -135)	x (-45, -45, -45)	x (-45, -45, 45)	x(-45, -45, 135)
315	√	√	√	√
		θx = 225		
θy = 45	√	√	√	√
135	x(45, 45, -135)	x (45, 45, -45)	x (45, 45, 45)	x(45, 45, 135)
225	x(45, -45, -135)	x (45, -45, -45)	x (45, -45, 45)	x(45, -45, 135)
315	√	√	√	√
		θx = 315		
θy = 45	√	√	√	√
135	x(135, 45, -135)	x (135, 45, -45)	x (135, 45, 45)	x(135, 45, 135)
225	x(135, -45, -135)	x (135, -45, -45)	x (135, -45, 45)	x(135, -45, 135)
315	√	√	√	√

3.2 Translation and Rotation

The motion of a module in the workspace can be described using rotation or translation or both.

3.2.1 General Motion of a Single Rigid Module

As shown in Figure 3.4, the general motion of a single module is the combination of rotation and translation, and the position vector **p** is

$$\mathbf{p} = \mathbf{Rb'} + \mathbf{h} = \mathbf{h} + \mathbf{Rb'} \tag{3.40}$$

where **h** is the vector of translation, **b'** is the body vector in local coordinate system, and

R is the rotation matrix.

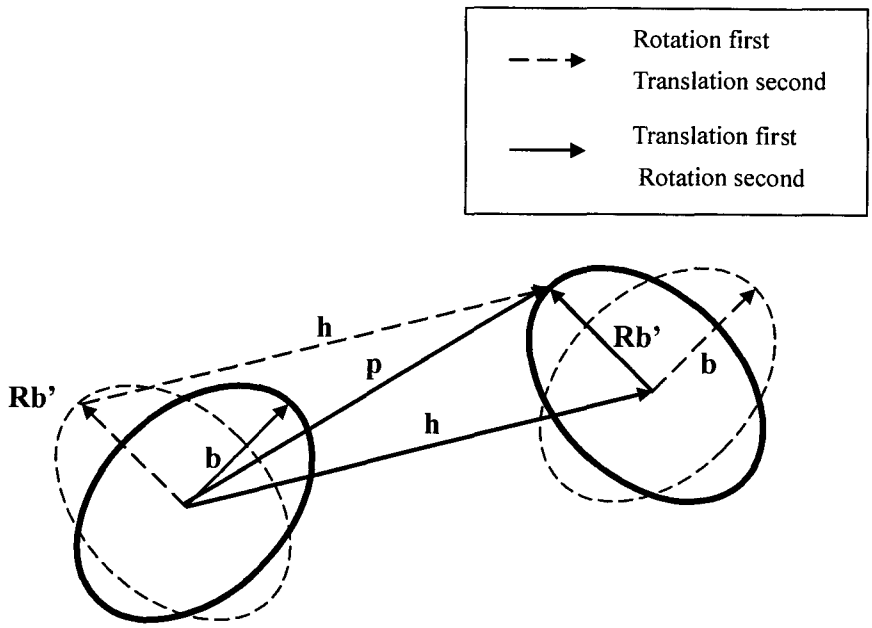


Figure 3.4: General motion of a single module [18].

Clearly, Equation (3.40) is communicative, meaning the order of rotation and translation can be reversed, which can also be found in Figure 3.4. When **h** is null, it becomes pure rotation.

3.2.2 General Motion of Multiple Modules

Figure 3.5 shows the vector method used to compute the position of a multi-module system.

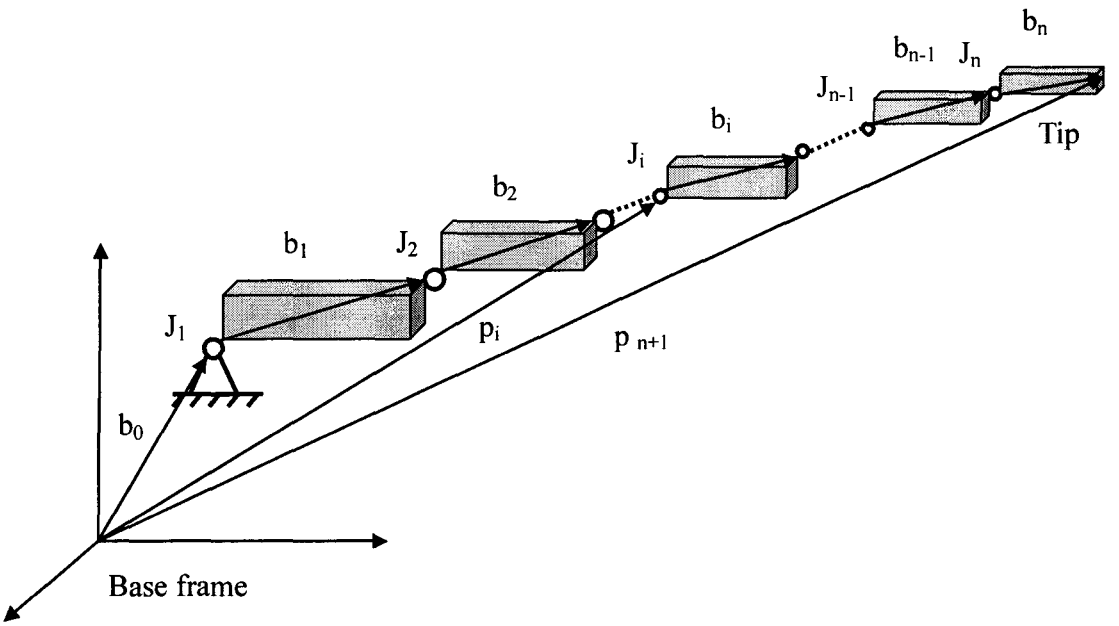


Figure 3.5: Vector method for a multi-module system [18]

The position of each joint from 1, 2 to i can be expressed respectively as:

Joint 1 $\mathbf{p}_1 = \mathbf{b}_0$

Joint 2 $\mathbf{p}_2 = \mathbf{b}_0 + \mathbf{R}_{01}\mathbf{b'}_1 = \mathbf{p}_1 + \mathbf{R}_{01}\mathbf{b'}_1$

Joint 3 $\mathbf{p}_3 = \mathbf{b}_0 + \mathbf{R}_{01}(\mathbf{b'}_1 + \mathbf{R}_{12}\mathbf{b'}_2) = \mathbf{p}_1 + \mathbf{R}_{01}\mathbf{R}_{12}\mathbf{b'}_2$

.....

Joint i $\mathbf{p}_i = \mathbf{b}_0 + \mathbf{R}_{01}(\mathbf{b'}_1 + \ldots + \mathbf{R}_{i-1\,i}\mathbf{b'}_i) = \mathbf{p}_{i-1} + \mathbf{R}_{0\,i-1}\mathbf{b'}_{i-1}$ (3.41)

where $\mathbf{R}_{i-1\,i}$ defines the rotation between two coordinate systems attached to two adjacent modules i-1 and i; $\mathbf{b'}_i$ is the local body vector, representing the translation between two coordinate systems, or defining the ith joint to the (i+1)th joint in the ith local coordinate frame.

Clearly, Equation (3.41) is a recursive method for computing the position of a multi-module system.

Similarly, the recursive method of computing rotation can be given as

$$\mathbf{R}_{0i} = \mathbf{R}_{01}\mathbf{R}_{12}\ldots\mathbf{R}_{i-1i} = \mathbf{R}_{0i-1}\mathbf{R}_{i-1i} \tag{3.42}$$

Hence, in general, the pose (position and orientation) of the end-effector of a n-module system can be expressed as [18]

Position
$$\mathbf{p}_{n+1} = \sum_{i=0}^{n} \mathbf{R}_{0i}\mathbf{b}'_i = \sum_{i=0}^{n} \mathbf{b}_i \tag{3.43}$$

Orientation
$$\mathbf{R}_{0n} = \prod_{j=1}^{n} \mathbf{R}_{(j-1)j} \tag{3.44}$$

As for a robot system, it usually has the default home configuration or initial configuration setup. Therefore, it should be noted that all the employed parameters here may have static part and motion part. The static part is according to the initial configuration setup, and the motion part represents the movement of each joint.

$$\mathbf{R} = \mathbf{R}_s\,\mathbf{R}_m \tag{3.45}$$

$$\mathbf{b} = \mathbf{b}_s + \mathbf{b}_m \tag{3.46}$$

where $\mathbf{R}_s$ and $\mathbf{b}_s$ are initial configuration setup, which are the geometric parameters need to identify; and $\mathbf{R}_m$ and $\mathbf{b}_m$ are related to active joints, i.e., motors. The static part can be further expressed by the PRY angles rotation as

$$\mathbf{R}_s = \mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \tag{3.47}$$

$$\mathbf{b}_s = b_x\mathbf{x} + b_y\mathbf{y} + b_z\mathbf{z} \tag{3.48}$$

where $\mathbf{R}_x$, $\mathbf{R}_y$, and $\mathbf{R}_z$ are the rotation about x, y and z axis of the configuration setup; b_x, b_y, and b_z are the translation along the x, y, and z axe of the configuration setup.

In terms of different kinematic pairs, they may be expressed differently according to joint movements, as shown in Table 3.3. As for robotics, usually only revolute and prismatic joints are considered in reality.

In Table 3.3, $\mathbf{R}(\theta_z)$, $\mathbf{R}(\theta_y)$, and $\mathbf{R}(\theta_x)$ are the rotation about z, y, and x axis of the joint respectively; while $\mathbf{s}_z$ is the translation along z axis of the joint. Conventionally, the first physical rotation or translation axis of a joint should be defined as axis z, second as y, and last as x. It should be noted this convention is totally different from the sequence of PRY angel set, which is used to represent orientation instead of a rotation matrix.

Table 3.3: $\mathbf{R}_m$ and $\mathbf{b}_m$ of different kinematic pairs [18].

Joint	$\mathbf{R}_m$	$\mathbf{b}_m$
Revolute	$\mathbf{R}(\theta_z)$	0
Prismatic	$\mathbf{R}(0) = 1$	$\mathbf{s}_z$
Cylinder	$\mathbf{R}(\theta_z)$	$\mathbf{s}_z$
Universal	$\mathbf{R}(\theta_z)\mathbf{R}(\theta_y)$	0
Spherical	$\mathbf{R}(\theta_z)\ \mathbf{R}(\theta_y)\ \mathbf{R}(\theta_x)$	0

3.3 Three-point Target Measurements

We know that no device can directly measure the complete pose of an object in space,

because direct measurement of the orientation is difficult. As a result, the triangular three-point target, which is made of three equal spheres whose relative positions are known, is introduced and utilized to determine the tip orientation indirectly (Figure 2.3, 2.4, 2.5, and 3.6).

As shown in Figure 3.6, the robot end-effector has a set of three spheres whose relative positions are already known, and they can be either inserted in an array of short-range displacement measuring fixture [17] or measured precisely in a CMM [37]. The kinematic modeling of the three-point measurement is presented in Figure 3.7. In this Figure, $\mathbf{p}_{o1}$, $\mathbf{p}_{o2}$, and $\mathbf{p}_{o3}$ are the initial positions of the three-point target; while $\mathbf{p}_{f1}$, $\mathbf{p}_{o2}$, and $\mathbf{p}_{o3}$ are the final ones measured by certain devices.

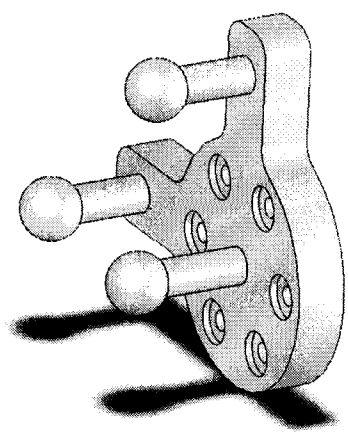


Figure 3.6: Three-point moving target.

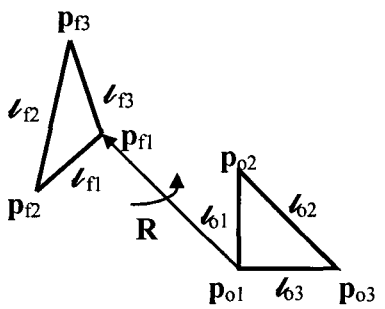


Figure 3.7: Transformation determinations using the three-point position data [55]

The initial and final three line vectors formed by the three-point target are [55]

$$[\mathbf{l}_{o1} \quad \mathbf{l}_{o2} \quad \mathbf{l}_{o3}]=[\mathbf{p}_{o1}-\mathbf{p}_{o2} \quad \mathbf{p}_{o2}-\mathbf{p}_{o3} \quad \mathbf{p}_{o3}-\mathbf{p}_{o1}] \tag{3.50a}$$

and

$$[\mathbf{l}_{f1} \quad \mathbf{l}_{f2} \quad \mathbf{l}_{f3}]=[\mathbf{p}_{f1}-\mathbf{p}_{f2} \quad \mathbf{p}_{f2}-\mathbf{p}_{f3} \quad \mathbf{p}_{f3}-\mathbf{p}_{f1}] \tag{3.50b}$$

Then, the rotation matrix from the initial pose to the final pose can be obtained by

$$\mathbf{D}_f=\mathbf{R}\mathbf{D}_o \tag{3.51}$$

where

$$\mathbf{D}_0=[\mathbf{l}_{o1} \quad \mathbf{l}_{o2} \quad \mathbf{l}_{o1}\times\mathbf{l}_{o2}] \tag{3.52a}$$

$$\mathbf{D}_f=[\mathbf{l}_{f1} \quad \mathbf{l}_{f2} \quad \mathbf{l}_{f1}\times\mathbf{l}_{f2}] \tag{3.52b}$$

Hence, the measured end-effector rotation matrix can be determined by

$$\mathbf{R}_m=\mathbf{D}_f(\mathbf{D}_o)^{-1}. \tag{3.53}$$

CHAPTER 4 LINEAR FORMULATION

Chapter 4 discusses the conventional linear formulation for robot calibration. Error models are derived from linearization of robot forward kinematic equations in two schemes, partial-pose (position) and full-pose calibration, depending on whether the orientation of end-effector is available or not. For the full-pose calibration, two methodologies are also investigated when calculating the end-effector pose error and implementing kinematic model error compensation, from which four full-pose calibration categories are generated. Several limitations are found when discussing these two issues. One downside is the orthogonality sacrifice of the rotation matrix leading to ill-conditioning of the Jacobian in the compensation step, and another is the quadrant sensitivity during the determination of the PRY angles from inversing the rotation matrix. Least-square estimation is applied in the identification of the parameter errors from the error model and tip pose error.

4.1 Error Model

To perform calibration, an error model is developed that takes into consideration all the geometric errors due to imprecision in manufacturing and assembly [18]. Based on this error model, it is shown that the error mapping from the geometric errors to the pose error of the tip of robot depends on the Jacobian matrix. Wu [31] assumed the actual or

measured geometric parameters were quite close to the nominal or calculated values and expanded the total transformation as a Taylor series about the nominal values. Keeping only the first order terms resulted in a linear expression for the differential deviations of the Denavit-Hartenberg parameters. Such an idea can be extended here to generate the linear error formulation of the complete kinematic model in Equation (1.1).

With linearization by Taylor series expanding, the error model can be derived as

$$\Delta \mathbf{X} = \frac{\partial F}{\partial g} \Delta \mathbf{g} = \mathbf{J} \Delta \mathbf{g} \tag{4.1}$$

where $\Delta \mathbf{X}$ represents the pose error of the end-effector of a robot, which is the difference between the actual (or measured) and the nominal (or calculated) tip poses in Equation (1.4); $\Delta \mathbf{g}$ represents the kinematic errors, which includes manufacturing errors, assembly errors, joint errors, all of which are to be identified together for the robot; $\mathbf{J}$ is the Jacobian matrix, or the error mapping matrix, mapping the kinematic parameter errors to the pose error of the end effector.

Two methods are classified here to do the calibration depending on the availability of the orientation measurement of the tip, namely partial (only position) and full (position and orientation) pose measurements.

4.1.1 Error Model with Full Pose Measurement

With both position and orientation measurements, the error model in Equation (4.1) can be further detailed as

$$\Delta \mathbf{X}_{6\times 1} = \begin{bmatrix} \Delta \mathbf{p} \\ \Delta \boldsymbol{\beta} \end{bmatrix}_{6\times 1} = \begin{bmatrix} \mathbf{J}_p \\ \mathbf{J}_\beta \end{bmatrix}_{6\times 6n} \Delta \mathbf{g}_{6n\times 1} \tag{4.2a}$$

$$\Delta \mathbf{g}_i = \begin{bmatrix} \Delta x_i & \Delta y_i & \Delta z_i & \Delta p_i & \Delta r_i & \Delta s_i \end{bmatrix}^T \tag{4.2b}$$

where $\Delta \mathbf{X}$ is a 6x1 vector representing the robot tip errors including $\Delta \mathbf{p}$ and $\Delta \boldsymbol{\beta}$, which are the position and orientation error of the tip respectively; $\mathbf{J}$ is the Jacobian, as shown in Equation (4.3a), which is a 6x6n matrix, with n joints; $\Delta \mathbf{g}$ is a 6nx1 vector representing the component errors, which are generalized as six infinitesimal errors for each module, including infinitesimal translation Trans(Δx_i, Δy_i, Δz_i) and infinitesimal rotation Rot(Δp_i, Δr_i, Δs_i), where Δx_i, Δy_i and Δz_i represent the linear errors along x, y and z axes of the frame i respectively, and Δp_i, Δr_i and Δs_i indicate the angular errors about x, y and z axes of the frame i respectively. The scripts p, r, and s represent pitch, roll and spin (or yaw) angles, respectively.

Considering the concept of multi-body velocity computation using the Jacobian [18] and the fact that the velocity is the derivative of the position, the velocity formula can be transformed and utilized into the error model by replacing the linear and angular velocities with infinitesimal translational and rotational errors respectively, as below:

$$\begin{bmatrix}\Delta\mathbf{p}\\ \Delta\boldsymbol{\beta}\end{bmatrix}_{6\times1} = \mathbf{J}\Delta\mathbf{g} = \begin{bmatrix}\mathbf{J}_{P1} & \mathbf{J}_{P2} & \cdots & \mathbf{J}_{Pi} & \cdots & \mathbf{J}_{Pn}\\ \mathbf{J}_{O1} & \mathbf{J}_{O2} & \cdots & \mathbf{J}_{Oi} & \cdots & \mathbf{J}_{On}\end{bmatrix}_{6\times6n} \begin{bmatrix}\Delta\mathbf{g}_1\\ \Delta\mathbf{g}_2\\ \vdots\\ \Delta\mathbf{g}_i\\ \vdots\\ \Delta\mathbf{g}_n\end{bmatrix}_{6n\times1} \tag{4.3a}$$

where the i [th] Jacobian matrix $\mathbf{J}_i$ is

$$\mathbf{J}_i = \begin{bmatrix}\mathbf{J}_{Pi}\\ \mathbf{J}_{Oi}\end{bmatrix} = \begin{bmatrix}\mathbf{X}_i & \mathbf{Y}_i & \mathbf{Z}_i & \mathbf{X}_i\times\mathbf{P}_i^{n+1} & \mathbf{Y}_i\times\mathbf{P}_i^{n+1} & \mathbf{Z}_i\times\mathbf{P}_i^{n+1}\\ 0 & 0 & 0 & \mathbf{X}_i & \mathbf{Y}_i & \mathbf{Z}_i\end{bmatrix}_{6\times6} \tag{4.3b}$$

with i ranges from 1 to n, n is the number of joints. $\bar{\mathbf{X}}_i$, $\bar{\mathbf{Y}}_i$, $\bar{\mathbf{Z}}_i$ are the three unit vectors of the global Cartesian coordinate in the ith joint; $\bar{\mathbf{P}}_i^{n+1}$ is the position vector from ith joint to the tip of a robot,

$$\mathbf{P}_i^{n+1} = \sum_{j=i}^{n}\mathbf{R}_{0j}\mathbf{b}_j' = \sum_{j=i}^{n}\mathbf{b}_i \tag{4.3c}$$

Hence, the tip error unit contributed by the ith module is

$$\begin{bmatrix}\Delta\mathbf{p}_i\\ \Delta\boldsymbol{\beta}_i\end{bmatrix}_{6\times1} = \begin{bmatrix}\mathbf{J}_{Pi}\\ \mathbf{J}_{Oi}\end{bmatrix}\Delta\mathbf{g}_i = \begin{bmatrix}\mathbf{X}_i & \mathbf{Y}_i & \mathbf{Z}_i & \mathbf{X}_i\times\mathbf{P}_i^{n+1} & \mathbf{Y}_i\times\mathbf{P}_i^{n+1} & \mathbf{Z}_i\times\mathbf{P}_i^{n+1}\\ 0 & 0 & 0 & \bar{\mathbf{X}}_i & \bar{\mathbf{Y}}_i & \bar{\mathbf{Z}}_i\end{bmatrix}_{6\times6} \begin{bmatrix}\Delta x_i\\ \Delta y_i\\ \Delta z_i\\ \Delta p_i\\ \Delta r_i\\ \Delta s_i\end{bmatrix}_{6\times1} \tag{4.4}$$

Obviously, it can be found that this general error model covers all the robot kinematic parameter errors. Therefore, it can be regarded as a complete parametric calibration model.

4.1.2 Error Model with Position Measurement

Without the orientation measurement of the robot tip, the error model shrinks to

$$\Delta \mathbf{p}_{3\times 1} = \mathbf{J}_{p3\times 6n} \Delta \mathbf{g}_{6n\times 1} \tag{4.5}$$

and the position error of the robot tip contributed by the ith module is

$$\Delta \mathbf{p}_{i3\times 1} = \mathbf{J}_{Pi} \Delta \mathbf{g}_i = \begin{bmatrix} \mathbf{X}_i & \mathbf{Y}_i & \mathbf{Z}_i & \mathbf{X}_i \times \mathbf{P}_i^{n+1} & \mathbf{Y}_i \times \mathbf{P}_i^{n+1} & \mathbf{Z}_i \times \mathbf{P}_i^{n+1} \end{bmatrix}_{3\times 6} \begin{bmatrix} \Delta x_i \\ \Delta y_i \\ \Delta z_i \\ \Delta p_i \\ \Delta r_i \\ \Delta s_i \end{bmatrix}_{6\times 1} \tag{4.6}$$

Compared with the full pose measurements, pure position measurements are easier to perform and the calibration algorithms are easier to implement.

4.2 Pose Error Calculations

The pose error of end effector, $\Delta \mathbf{X}$ in Equation (4.1), is equal to the difference between the actual (or measured) and the nominal (or calculated) value of end-effector pose,

$$\Delta \mathbf{X} = \begin{bmatrix} \Delta \mathbf{p} & \Delta \boldsymbol{\beta} \end{bmatrix}^T \tag{4.7}$$

where $\Delta \mathbf{p}$ and $\Delta \boldsymbol{\beta}$ are the infinitesimal position and orientation error respectively. Two approaches are considered and compared in detail.

4.2.1 Differential Transformation

Firstly, the pose error in Equation (4.7) is regarded as the differential translation and rotation respectively, represented as [16],

$$\mathbf{d} = d_x i + d_y j + d_z k \tag{4.8a}$$

$$\boldsymbol{\delta} = \delta_x i + \delta_y j + \delta_z k \tag{4.8b}$$

Therefore, according to [16], considering the derivative as a differential translation and rotation in terms of the base coordinate frame, the measured pose transformation $\mathbf{T}_m$ can be obtained below,

$$\mathbf{T}_m = \mathbf{T}_n + \mathrm{d}\mathbf{T} = Trans(dx, dy, dz) Rot(\delta x, \delta y, \delta z) \mathbf{T}_n \tag{4.9}$$

where $\mathbf{T}_n$ is the nominal pose transformation; pose transformation matrix is $\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{P} \\ 0 & 1 \end{bmatrix}$.

So the derivative $\mathrm{d}\mathbf{T}$ is developed as

$$\mathrm{d}\mathbf{T} = \left(Trans(dx, dy, dz) Rot(\delta x, \delta y, \delta z) - I\right)\mathbf{T} = \boldsymbol{\Delta}\mathbf{T} \tag{4.10}$$

where $\boldsymbol{\Delta}$, the differential translation and rotation transformation, is given as

$$\boldsymbol{\Delta} = \begin{bmatrix} 0 & -\delta z & \delta y & dx \\ \delta z & 0 & -\delta x & dy \\ -\delta y & \delta x & 0 & dz \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.11}$$

which can be also called pose error transformation.

Since it is easy to calculate the position error $\Delta\mathbf{p}$, i.e., $\Delta\mathbf{p} = \mathbf{p}_{\text{actual}} - \mathbf{p}_{\text{nominal}}$, simply only considering the differential rotation δ, the transformation in Equation (4.11) becomes

$$\boldsymbol{\Delta} = \tilde{\delta} = \begin{bmatrix} 0 & -\delta z & \delta y \\ \delta z & 0 & -\delta x \\ -\delta y & \delta x & 0 \end{bmatrix} \tag{4.12}$$

which is the skew symmetric matrix of the differential rotation δ in Equation (4.8b).

As a result, Equation (4.10) shrinks to

$$d\mathbf{R}=\tilde{\delta}\mathbf{R}_n \tag{4.13}$$

Therefore, from Equation (4.9), the measured end-effecter rotation matrix $\mathbf{R}_m$ is

$$\mathbf{R}_m=\mathbf{R}_n+d\mathbf{R}=\mathbf{R}_n+\tilde{\delta}\mathbf{R}_n=\left(\tilde{\delta}+I\right)\mathbf{R}_n \tag{4.14}$$

where $\mathbf{R}_m$ can be measured by the three-point method as described in Chapter 3; $\mathbf{R}_n$ is the nominal rotation matrix of the end-effecter, which is calculated through the nominal model.

Hence, according to Equation (4.14), two equivalent methods are found to determine the differential rotation δ, if $\mathbf{R}_m$ and $\mathbf{R}_n$ are both known:

$$\tilde{\delta}=d\mathbf{R}\mathbf{R}_n^{\mathrm{T}}=\left(\mathbf{R}_m-\mathbf{R}_n\right)\mathbf{R}_n^{\mathrm{T}} \tag{4.15a}$$

or

$$\tilde{\delta}=\mathbf{R}_m\mathbf{R}_n^{\mathrm{T}}-\mathbf{I} \tag{4.15b}$$

Actually, the first method in Equation (4.15a) can also be derived by the tensor transformation,

$$\boldsymbol{\Omega}=\tilde{\omega}=\dot{\mathbf{R}}\mathbf{R}_n^{\mathrm{T}}=\left(d\mathbf{R}/dt\right)\mathbf{R}_n^{\mathrm{T}} \tag{4.16}$$

also

$$\omega = \delta / dt \tag{4.17}$$

where $\tilde{\omega}$ is the skew symmetric matrix of angular velocity ω.

So, from Equation (4.16) and (4.17),

$$\tilde{\delta} / dt = (d\mathbf{R}/dt)\mathbf{R}_n^T \tag{4.18}$$

Thereby, the skew-matrix of δ is determined as, identical to Equation (4.15a),

$$\tilde{\delta} = d\mathbf{R}\mathbf{R}_n^T = (\mathbf{R}_m - \mathbf{R}_n)\mathbf{R}_n^T \tag{4.19}$$

Consequently, the orientation errors $\Delta\boldsymbol{\beta}$ can be solved as

$$\Delta\boldsymbol{\beta} = \text{vect}(\tilde{\delta}) \tag{4.20}$$

where vect( ) is the vector operation, which transforms a skew symmetric matrix into a column vector.

4.2.2 Inverse of a Rotation Matrix

Alternatively, another simple and straight-forward approach of determining the end-effector orientation error is to calculate the Bryan (PRY) rotation angles directly from the rotation matrix $\mathbf{R}_m$ and $\mathbf{R}_n$:

$$\boldsymbol{\theta} = f^{-1}(\mathbf{R}) \tag{4.21}$$

Many approaches about $f^{-1}( )$ have been proposed so far, however, none of them is able to function correctly for all four quadrants from -180° to 180°. This limitation is called

quadrant sensitivity stated previously in Section 3.1.3. Hence, the calibration configurations should be considered and chosen carefully to avoid those limitations.

Here is one of them:

$$\theta_y = \cos^{-1}\left(\left[\left(r_{23}\right)^2 + \left(r_{33}\right)^2\right]^{1/2}\right)$$
$$\theta_x = \cos^{-1}\left(r_{33}/\cos\theta_y\right) \tag{4.22}$$
$$\theta_z = \cos^{-1}\left(r_{11}/\cos\theta_y\right)$$

where r_{23}, r_{33} and r_{11} are elements from rotation matrix according to the number subscripts, as shown in the Appendix A.

Then, the orientation error can be determined by subtraction between the measured and nominal PRY angles

$$\Delta\boldsymbol{\theta} = \boldsymbol{\theta}_m - \boldsymbol{\theta}_n \tag{4.23}$$

Hence, as the second approach, the orientation error can be determined from $\omega = \boldsymbol{\Phi}\dot{\theta}$ [18], as

$$\Delta\boldsymbol{\beta} = \boldsymbol{\Phi}\Delta\boldsymbol{\theta} \tag{4.24}$$

where

$$\boldsymbol{\Phi} = \begin{bmatrix} 1 & 0 & \sin\theta_y \\ 0 & \cos\theta_x & -\sin\theta_x\cos\theta_y \\ 0 & \sin\theta_x & \cos\theta_x\cos\theta_y \end{bmatrix} \tag{4.25}$$

4.3 Computational Methods

Computational methods involve three aspects. The first one is to perform the inverse of the error mapping matrix **J** which is not a square matrix. The second one is component error compensation, for which two methods are investigated in details. The last aspect is an iterative algorithm for calibration.

4.3.1 Least Squares Method

In order to identify the linear error model in Equation (4.1), two categories of the least squares estimation problem are investigated, namely, underdetermined and over-determined problems, according to the dimension of the error mapping matrix, **J**, or the number of measurements, m. To ensure the accuracy, measurement points should be sufficient to minimize computational errors, and also the condition number of the Jacobian **J** should be kept low enough within a reasonable bound.

4.3.1.1 Underdetermined

If only one set of tip poses is measured, as the number of joints n≥1, the number of unknown geometric error parameters is larger than that of independent equations in the Equation (4.1). This is a case of an underdetermined problem. The pseudo-inverse method can be used to obtain a minimum-norm solution as follows,

$$\Delta \mathbf{g}_{6n\times 1}=\mathbf{J}^{\mathrm{T}}{}_{6n\times 6}\left(\mathbf{J}_{6\times 6n}\mathbf{J}^{\mathrm{T}}{}_{6n\times 6}\right)^{-1}{}_{6\times 6}\Delta \mathbf{X}_{6\times 1},\ n\geq 1 \text{ and } m=1 \tag{4.26}$$

where $\mathbf{J}^{\dagger}=\mathbf{J}^{T}\left(\mathbf{J}\mathbf{J}^{T}\right)^{-1}$ is the pseudo-inverse of **J** for underdetermined situations.

4.3.1.2 Over-determined

A large number of poses must be available to more accurately reflect the robot geometrical characteristics. In order to make Equation (4.1) over-determined, the number of measurement, m, is increased sufficiently to make sure the number of equations, 6m, is larger than the unknown geometric error parameters, 6n. Hence, the pseudo inverse solution is given as when the matrix $\mathbf{J}^T\mathbf{J}$ is nonsingular,

$$\Delta\mathbf{g}_{6n\times 1} = \left(\mathbf{J}^T{}_{6n\times 6m}\mathbf{J}_{6m\times 6n}\right)^{-1}{}_{6n\times 6n}\,\mathbf{J}^T{}_{6n\times 6m}\Delta\mathbf{X}_{6m\times 1}, \quad m \geq n \tag{4.27}$$

where $\mathbf{J}^{\dagger} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{J}^T$ is also called the pseudo-inverse of **J**, when m ≥ n.

The condition number of $\mathbf{J}^T\mathbf{J}$ is used to evaluate the observability and measurement strategy [45]. The better the strategy, the lower the corresponding condition number [19].

4.3.2 Kinematic Model Error Compensation

After identifying the component errors as given below:

$$\Delta\mathbf{g}_i = \left[\Delta\mathbf{b}_i \quad \Delta\boldsymbol{\theta}_i\right] = \left[\Delta x_i \quad \Delta y_i \quad \Delta z_i \quad \Delta p_i \quad \Delta r_i \quad \Delta s_i\right]^T$$

the nominal kinematic model should be updated accordingly until the pose error of the end-effector meets certain accuracy requirements.

Iterative updating for static body vectors, $\mathbf{b}_s$, yields

$$\mathbf{bs}'_{i+1} = \mathbf{bs}'_i + \Delta\mathbf{b}_i \tag{4.28}$$

where

$$\Delta \mathbf{b}_i = [\Delta x_i \quad \Delta y_i \quad \Delta z_i] \tag{4.28a}$$

However, as for the update of static rotation matrix $\mathbf{R}_S$ of each module, two iterative methods are investigated.

The first updating method is more straightforward, which adds the calculated orientation errors into the initial orientation set up and then determines the static rotation matrix accordingly by

$$\mathbf{Rs}_{i+1} = \mathbf{R}_{pry}(\boldsymbol{\theta}_{i+1}) = \mathbf{R}_{pry}(\boldsymbol{\theta}_i + \Delta\boldsymbol{\theta}_i) \tag{4.29}$$

where

$$\Delta\boldsymbol{\theta}_i = [\Delta p_i \quad \Delta r_i \quad \Delta s_i]^T \tag{4.29a}$$

$$\boldsymbol{\theta}_i = [\theta_x \quad \theta_y \quad \theta_z]_i^T$$

$$\mathbf{R}_{pry}(\boldsymbol{\theta}) = \mathbf{R}(\theta_x)\,\mathbf{R}(\theta_y)\,\mathbf{R}(\theta_z) \tag{4.29b}$$

The second updating method is derived from the differential rotation in Equation (4.12) and (4.14). The skew matrix of the identified orientation error for each link is compensated for the local static rotation matrix through the function below,

$$\mathbf{Rs}_{i+1} = \mathbf{Rs}_i + d\mathbf{Rs}_i = \mathbf{Rs}_i\left(\mathbf{I} + \tilde{\delta}_i\right) \tag{4.30}$$

where

$$d\mathbf{Rs}_i = \mathbf{Rs}_i\tilde{\delta}_i \tag{4.30a}$$

$$\tilde{\delta} = \begin{bmatrix} 0 & -\Delta s & \Delta r \\ \Delta s & 0 & -\Delta p \\ -\Delta r & \Delta p & 0 \end{bmatrix} \tag{4.30b}$$

However, after being updated by Equation (4.30), the orthogonality of the static rotation matrix is slightly sacrificed, and it may cause the ill-conditioning of the Jacobian, which would result in the poor performance or non-convergence.

4.3.3 Calibration Algorithm

Calibration is a process of determining a set of parameters in the model that best describes the specific robot under study [17]. As shown in the flowchart (Figure 4.1), the whole calibration algorithm is an iterative loop.

In evaluating the pose error, three parameters are used: RMSPE, RMSOE and RMSE, which are the root mean square of the position, orientation and pose errors defined respectively as [59]

$$\text{RMSPE} = \sqrt{\frac{1}{n}\sum_{1}^{3}|\Delta p|^2}, \ \Delta p = \Delta x, \Delta y, \Delta z \tag{4.31a}$$

$$\text{RMSOE} = \sqrt{\frac{1}{n}\sum_{1}^{3}|\Delta \beta|^2}, \ \Delta \beta = \Delta p, \Delta r, \Delta s \tag{4.31b}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{1}^{6}|\Delta e|^2}, \ \Delta e = \Delta x, \Delta y, \Delta z, \Delta p, \Delta r, \Delta s \tag{4.31c}$$

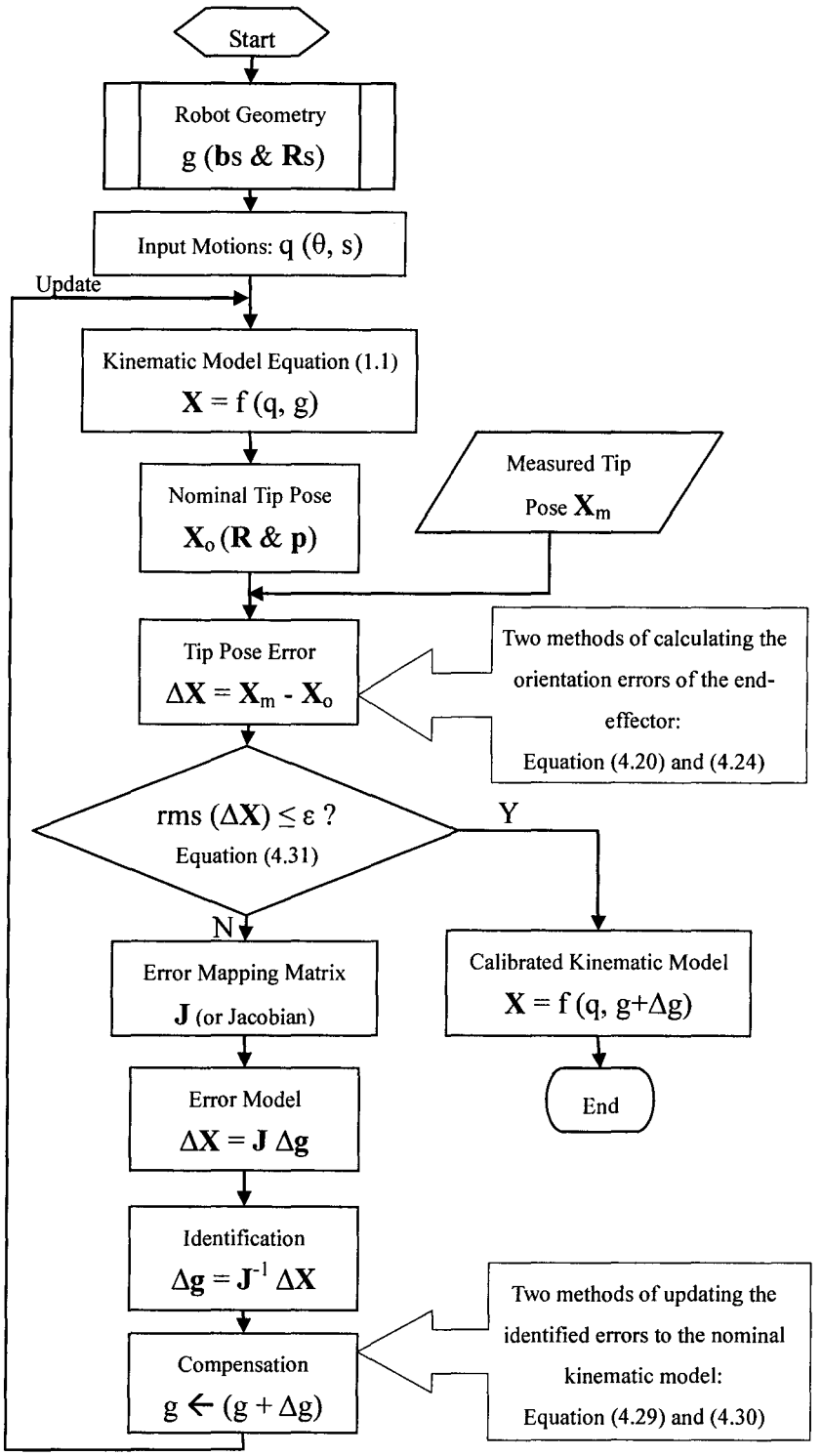


Figure 4.1: Conventional calibration algorithm (linear formulation).

After carrying out the full pose calibration simulation, different convergent features are found between these three parameters. The termination of the iterative looping occurs when the pose deviation of the robot tip is within the certain specified precision, RMSE $\leq$ $|\delta|$.

As mentioned in the Sections 4.2 and 4.3.2, there are mainly two methods of determining the end-effector pose error and compensating the component errors in the nominal kinematic model, which are combined together in Table 4.1. There are four categories studied in the calibration simulations.

Table 4.1: Four categories of calibration simulations

Component Errors Compensation	End-Effector Pose Errors Calculation	
	$\Delta\boldsymbol{\beta}=\text{vect}\left(\tilde{\delta}\right)$ Equation (4.20) and (4.17)	$\Delta\boldsymbol{\beta}=\boldsymbol{\Phi}(\boldsymbol{\theta}_m-\boldsymbol{\theta}_n)$ Equation (4.23) and (4.24)
$\mathbf{Rs}_{i+1}=\mathbf{Rs}_i+d\mathbf{Rs}_i=\mathbf{Rs}_i\left(\mathbf{I}+\tilde{\delta}_i\right)$ Equation (4.30)	1	2
$\mathbf{Rs}_{i+1}=\mathbf{R}_{pry}(\boldsymbol{\theta}_i+\Delta\boldsymbol{\theta}_i)$ Equation (4.29)	3	4

Simulations have showed that the static rotation matrix is no longer orthogonal after being updated recursively using the first compensation method, i.e., Equation (4.30). After losing the crucial orthogonality, a rotation matrix is no longer eligible to represent the orientation for a robot, which indicates that robot kinematic calibration should avoid

this compensation method shown in Equation (4.30). Therefore, the second method in Equation (4.29) would be a better way to update the identified component orientation error into the static rotation matrix for individual links. Furthermore, attention should be paid to quadrant sensitivity in the second method of the end-effector orientation error calculation (Equation (4.23)), which may fail to yield unique Euler angles from a rotation matrix of the end-effector in Equation (4.21).

CHAPTER 5 NONLINEAR FORMULATION

This chapter investigates the nonlinear formulation without linearizing the kinematic model. The norms of tip position error and rotation matrix error are used as the objective function to search for an optimal or global solution. A genetic algorithm (GA) is applied as the search engine. Attention is paid to the population size, generation numbers, crossover or mutation factors, initial population, and other GA options. Furthermore, Monte Carlo simulations are conducted to reduce the stochastic or random error.

5.1 Nonlinear Formulation

Compared with the conventional linear formulation, the nonlinear one determines the pose error of the end-effector between the nominal and measured pose directly,

$$\sum_{i=1}^{n}\|\Delta \mathbf{X}\|_2 = \sum_{i=1}^{n}\|\mathbf{X}_m - \mathbf{X}_o\|_2 = \sum_{i=1}^{n}\|\mathbf{X}_m - f(\mathbf{q}_0, \mathbf{g}_0 + \Delta \mathbf{g})\|_2 < \varepsilon \tag{5.1a}$$

$$\Delta \mathbf{g}_i = \begin{bmatrix} \Delta x_i & \Delta y_i & \Delta z_i & \Delta p_i & \Delta r_i & \Delta s_i \end{bmatrix}^T \tag{5.1b}$$

where $\sum_{i=1}^{n}\|\mathbf{X}_m - \mathbf{X}_o\|_2$ represents the summation of 2-norm values of end-effector pose errors from n different measurements; ε is the acceptable accuracy for the pose error of the end effector, which is the termination condition for the calibration iterations. Equation (5.1a) can be solved using genetic algorithm to search the optimal geometric errors $\Delta \mathbf{g}$ at which the norm value of pose errors is less than ε.

Furthermore, the determination of the pose error can be formulated as follows,

$$\|\mathbf{X}_m - \mathbf{X}_o\|_2 = \|\mathbf{p}_m - \mathbf{p}_o\|_2 + \|\mathbf{R}_m - \mathbf{R}_o\|_2 \tag{5.2}$$

where $\|\mathbf{p}_m - \mathbf{p}_o\|_2$ and $\|\mathbf{R}_m - \mathbf{R}_o\|_2$ are the Euclidean norm of position error and the spectral norm of rotation matrix error of the end-effector respectively. $\|x\|_2 = \sqrt{\sum_{i=1}^{n} |x_i|^2}$ represents the Euclidean norm of x in vector space, $\|A\|_2 = \sqrt{\lambda_{max}(A^*A)}$ represents the spectral norm, defined by the largest singular value of A or the square root of the largest eigenvalue of the positive-semidefinite matrix A*A.

The use of rotation matrices directly instead of PRY angles to represent the orientation in Equation (5.2) avoids the inverse operation that may cause the quadrant sensitivity problem. $\mathbf{p}_m$ and $\mathbf{R}_m$ are the measured position and rotation matrix of the robot's end-effector. For $\mathbf{p}_o$ and $\mathbf{R}_o$, it can be given by the robot forward kinematic equations (3.43) ~ (3.46) in Chapter 3.

5.2 Genetic Algorithm

A genetic algorithm (GA) is a kind of global adaptive probabilistic searching algorithm simulating biological heredity and evolution, first presented by Prof. Holland in Michigan University [60] [61]. GA includes four basic operators, namely, selection, crossover, mutation, and migration. The genetic algorithm repeatedly modifies a population of individual solutions with these operators. Meanwhile, the fitness of each individual in a

population is valued by its cost of fitness function, and the best individual has the lowest cost, which has the priority to be selected into the next generation.

5.2.1 Fitness Function

The fitness function is also known as the objective function in other standard optimization algorithms. This is the ultimate cost function that determines the optimization process and direction of GA. The fitness function assigns a higher selection probability to the individuals with the lower cost values. After the global search based on GA is over, we transform the best individual whose cost value is the lowest into the final indentified kinematic parameter errors [7]. A nonlinear identification approach can be regarded as a typical GA optimization problem by setting Equation (5.1) as the fitness, while adding appropriate weighting factors between position and orientation error, as given below

$$\mathrm{Min}\left(\sum_{i=1}^{n}\left\|\mathbf{X}_m-\mathbf{X}_o\right\|_2\right) \tag{5.3}$$

$$\left\|\mathbf{X}_m-\mathbf{X}_o\right\|_2 = w_1\left\|\mathbf{p}_m-\mathbf{p}_o\right\|_2 + w_2\left\|\mathbf{R}_m-\mathbf{R}_o\right\|_2 \tag{5.4}$$

where w_1 and w_2 are the weighting factors for position and orientation errors of the end effector, respectively, which are utilized to weigh the importance of two errors. When $w_1=1$ and $w_2=0$, it shrinks to a pure position calibration only; when $w_1=0$ and $w_2=1$, it shrinks to a pure orientation one.

Each individual of a population has 6n parameters in total as the variables in the GA search. Meanwhile, the geometric errors $\Delta \mathbf{g}_i$ (Equation 4.2b) are the variables that are optimized to minimize the cost of fitness function given in Equation (5.3). Furthermore, the position and orientation errors can be written as follows

$$\left\| \mathbf{p}_m - \mathbf{p}_o \right\|_2 = \left\| \mathbf{p}_m - \mathbf{p}_o (\mathbf{q}, \mathbf{g}+\Delta \mathbf{g}_i) \right\|_2 \tag{5.5}$$

$$\left\| \mathbf{R}_m - \mathbf{R}_o \right\|_2 = \left\| \mathbf{R}_m - \mathbf{R}_o (\mathbf{q}, \mathbf{g}+\Delta \mathbf{g}_i) \right\|_2 \tag{5.6}$$

where **g** represents the geometric parameters of each link, including the body vectors and initial configuration set up; **q** represents the joint variables, including rotational angles for revolute joints and translational displacements for prismatic joints.

5.2.2 Initial Population

A genetic algorithm starts generating a new population from the initial population that can be either created by a random generator, say, using a uniform distribution, or set manually. The initial population should provide the diversity of individuals. Assuming there are m population size and n number of variables, the initial population should use an m×n matrix to store. For robot kinematic calibration, the identified kinematic parameter errors from the linear formulation with position measurements can be considered to be an initial population. Both methods are evaluated in the calibration simulations, and better performance can be found when using calibrated $\Delta \mathbf{g}$ from the linear formulation. It requires fewer generations and therefore saves calculation times.

5.2.3 Individual Selection and Population Generations

The classical algorithm generates a single point at each iteration, and the sequence of points approaches an optimal solution. While GA generates a population of points at each iteration, the best point in the population approaches the optimal solution. Moreover, GA selects the next population by computation which uses a random number generator, as opposed to selecting the next point in the sequence by a deterministic computation in the classical algorithm [62]. With a large population size, the genetic algorithm searches the solution space more thoroughly, thereby reducing the chance that the algorithm will return a local minimum that is not a global minimum. However, a large population size also causes the algorithm to run more slowly.

To create the next generation, GA selects certain individuals in the current population, called parents, and then uses them to create individuals in the next generation, called children. Typically, the algorithm is more likely to select parents that have better fitness values. There are basically four operations for generation, which are selection, crossover, mutation, and migration. An individual with the lowest cost value of the fitness function, called an elite child, will be selected with priority into next generation automatically and survive to the next generation. Besides elite children, other relative lower individuals are selected as parents that contribute to the population at the next generation with crossover and mutation rules. Crossover operation combines two parents to form children for the

next generation. Mutation rules apply random changes to a single individual in the current generation to create a child. Mutation broadens the search space for GA and creates genetic diversity. Migration copies the best individuals in one subpopulation to replace the worst individuals in another subpopulation when the population size is more than one.

5.3 Calibration Algorithm

In Figure 5.1, a flowchart is given to show the robot calibration procedure based on the nonlinear formulation using GA. The initial population provides the genes for GA. The individuals with lower cost values are selected with priority to be parents for the next generation. The children are generated through the four basic operators, namely, selection, crossover, mutation, and migration. The final best individual is transformed to be the kinematic parameter errors and implemented into the robot kinematic model when it meets the required accuracy.

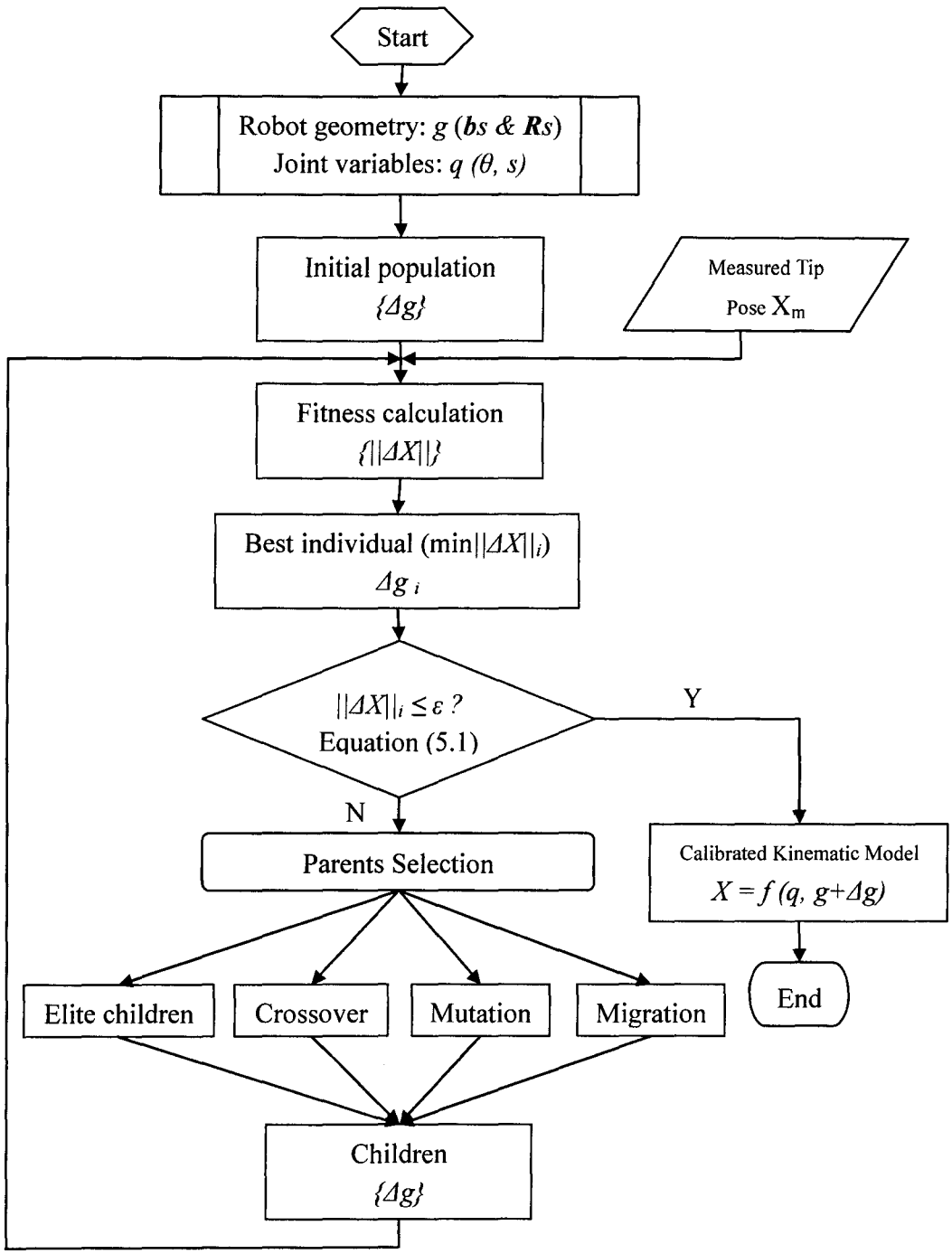


Figure 5.1: Nonlinear calibration using a genetic algorithm.

5.4 Monte Carlo Simulation

After each run, GA (loop 1 in Figure 5.2) returns a set of results for component kinematic parameter errors, $\Delta\mathbf{g}$. In order to reduce stochastic simulation error, Monte Carlo simulation (loop 2 in Figure 5.2) for the genetic algorithm is conducted by running GA for n times, roughly around 50 times, until the mean value of all searched parameter errors of all runs (Equation 5.9) become stable and fluctuates within a certain preset tolerance Δ in Equation (5.7).

$$\left|\overline{\Delta\mathbf{g}}^{k} - \overline{\Delta\mathbf{g}}^{k-1}\right| \le \Delta, 1 \le k \le n \tag{5.7}$$

$$\overline{\Delta\mathbf{g}}^{k-1} = \frac{\sum_{i=1}^{k-1} \Delta\mathbf{g}_i}{k-1} \tag{5.8}$$

$$\overline{\Delta\mathbf{g}}^{k} = \frac{\sum_{i=1}^{k} \Delta\mathbf{g}_i}{k} = \frac{\sum_{i=1}^{k-1} \Delta\mathbf{g}_i + \Delta\mathbf{g}_k}{k} = \frac{\overline{\Delta\mathbf{g}}^{k-1} \times (k-1) + \Delta\mathbf{g}_k}{k} \tag{5.9}$$

Simulations show that the more the run times, the more stable is the mean value. In addition to conducting a Monte Carlo simulation for a single configuration (loop 2), more random m configurations in the workspace are generated to repeat loop 2 until the mean value (Equation 5.10) for all the different configurations becomes stable, which can be considered as the global solution in the whole workspace (loop 3 in Figure 5.2).

$$\Delta\mathbf{g}_{global} = \sum_{j=1}^{m} \left(\overline{\Delta\mathbf{g}}\right)_j \Big/ m \tag{5.10}$$

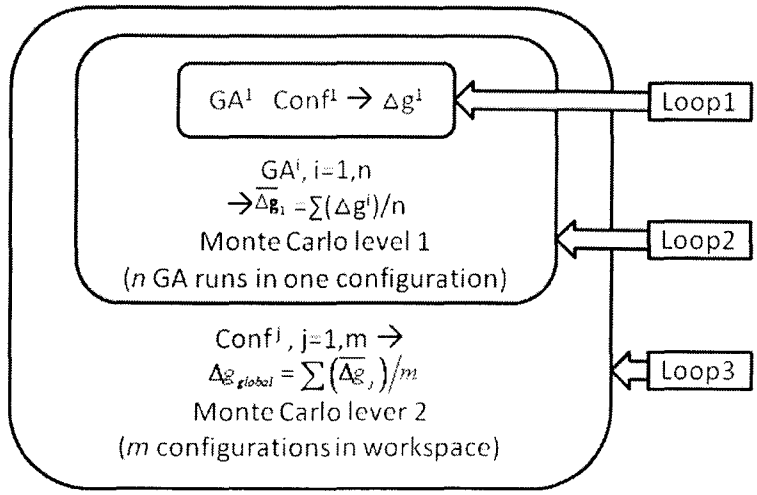


Figure 5.2: Three loops of simulation.

CHAPTER 6 SIMULATIONS AND COMPARISONS

In this chapter, the calibration algorithms, both linear and nonlinear formulations, are first tested individually and then compared. Figure 6.1 shows the graphical user interface (GUI) for the robot kinematic calibration simulation created in MatLab R14. It consists of a static configuration set-up, motion parameters, assumed link errors, the calibration using linear formulation, the calibration using nonlinear formulation, and outputs. The outputs include graphs of the robot configuration representation and the robot end-effector pose errors, and identified kinematic parameters.

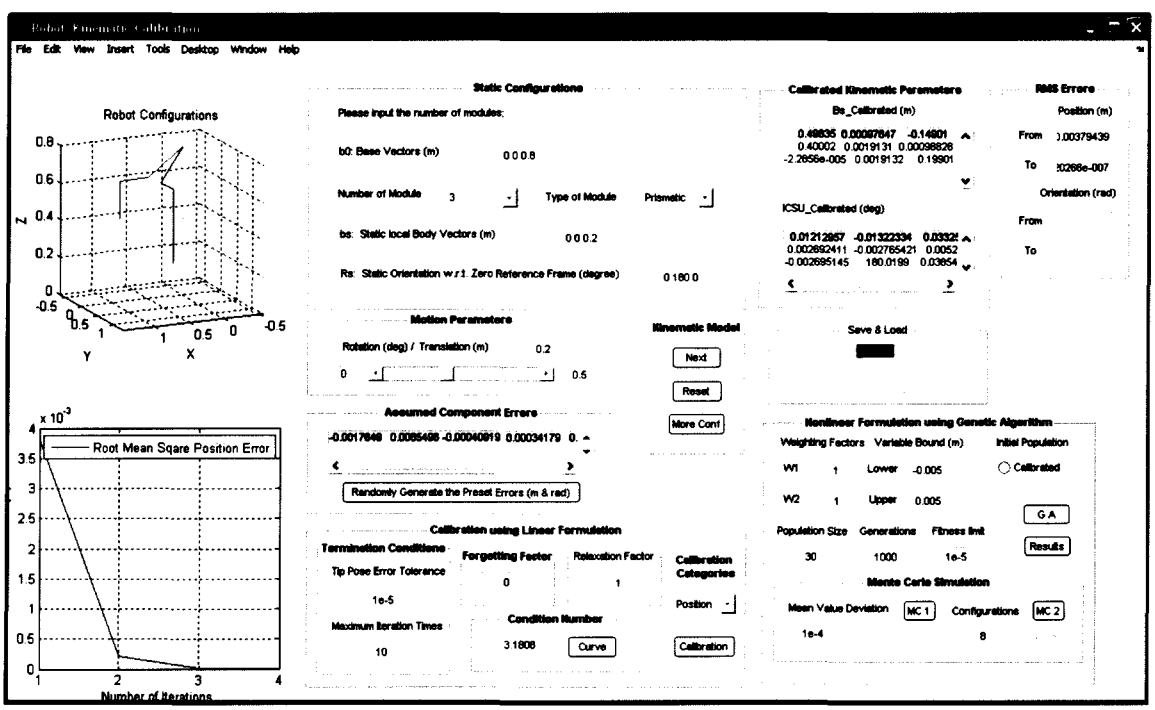


Figure 6.1: Graphical user interface (GUI) for robot kinematic calibration.

The simulations are carried out on the MRR-1 robot, as shown in Figure 6.2. The robot has three joints: two revolute and one prismatic. Table 6.1 lists the robot's kinematic parameters, including local body vectors, initial orientation set-up, and joint motions for simulations.

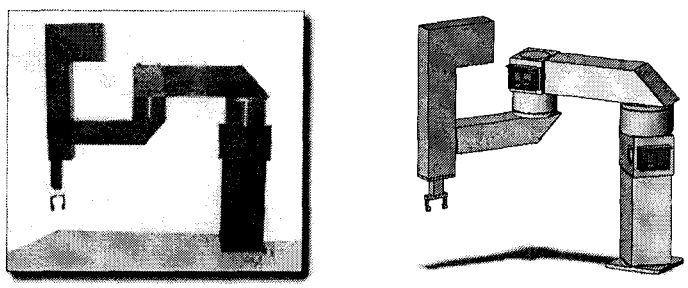


Figure 6.2: MRR-1 [8] and SolidWorks model.

Table 6.1: Nominal link kinematic parameters for MRR-1

Joint i	Local Body Vectors (m)			Zero Reference Initial Orientation set up (deg)			Joint Motions (deg/m)
	X	Y	Z	Pitch	Roll	Yaw	
Base	0	0	0.8	n/a	n/a	n/a	n/a
1 (R)	0.5	0	-0.15	0	0	0	30
2 (R)	0.4	0	0	0	0	0	60
3 (P)	0	0	0.2	0	180	0	0.2

All the kinematic parameters in Table 6.1 are inputs in the GUI (Figure 6.3). The robot configurations before and after movements (joint motions are shown in Table 6.1) are illustrated by red and blue solid lines, respectively, in Figure 6.4, and the dashed line represents the base vector from the global frame origin to the first revolute joint centre.

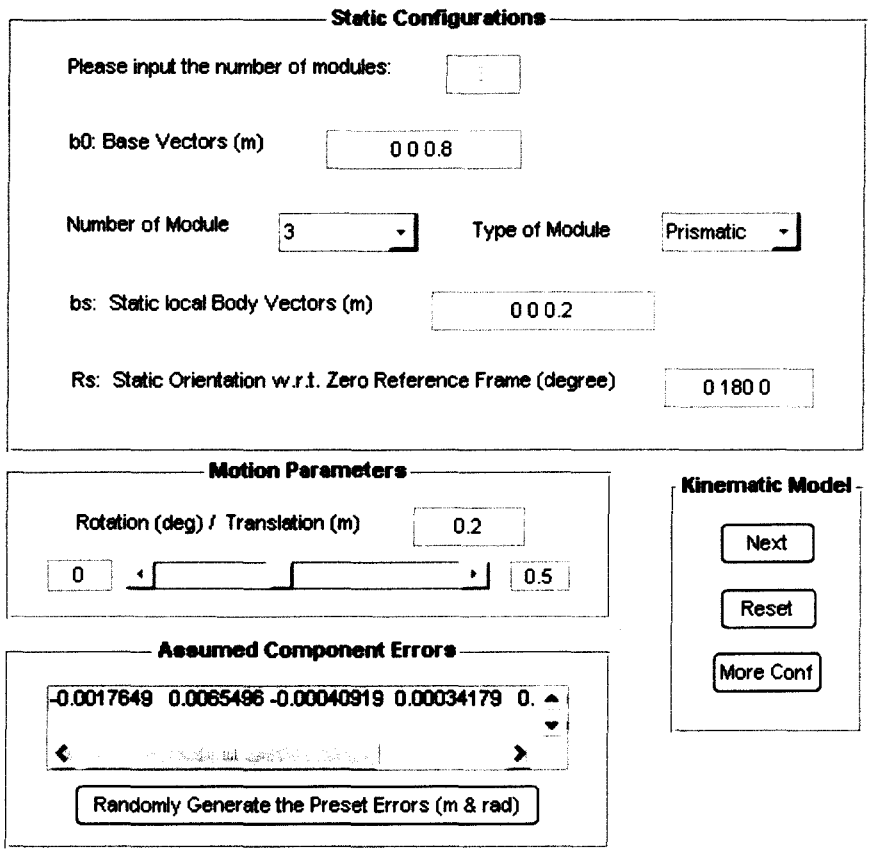


Figure 6.3: Kinematic parameter input panels in the GUI.

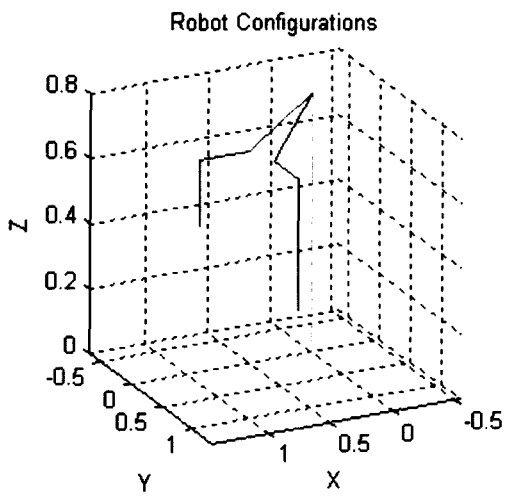


Figure 6.4: Robot configurations before (red lines) and after (blue) movements.

In the calibration simulations, the measured end-effector pose is determined using the nominal model with the assumed joint geometric parameter errors according to the manufacturing tolerances in Table 6.2. The nominal pose is determined using the model without the errors [59]. These assumed errors are generated randomly by the Gaussian (normal) distribution with the mean value $\mu = 0$ and variance $\sigma^2 = 0.003$. The random generator can be found in Figure 6.3. A set of assumed component geometric errors, listed in Table 6.3, is added to both local body vectors and initial orientation set-ups in the nominal kinematic model through Equations (4.28) and (4.29) or (4.30).

Table 6.2: Typical tolerance limits for various manufacturing processes [65]

Process	Typical Tolerance Limit		Process	Typical Tolerance Limits	
	mm	inches		mm	inches
Sand casting:			Abrasive processes:		
Cast iron	±1.3	±0.050	Grinding	±0.008	±0.0003
Steel	±1.5	±0.060	Lapping	±0.005	±0.0002
Aluminum	±0.5	±0.020	Honing	±0.005	±0.0002
Die casting	±0.12	±0.005	Nontraditional process:		
Plastic molding:			Chemical machining	±0.08	±0.003
Polyethylene	±0.3	±0.010	Electric discharge	±0.025	±0.001
Polystyrene	±0.15	±0.006	Electrochem. grind	±0.025	±0.001
Machining:			Electrochem. machine	±0.05	±0.002
Drilling, diameter			Electron beam cutting	±0.08	±0.003
6 mm (0.250 in)	+0.08, -0.003	+0.003, -0.001	Laser beam cutting	±0.08	±0.003
25 mm (1.000 in)	+0.13, -0.05	+0.006, -0.002	Plasma arc cutting	±1.3	±0.050
Milling	±0.08	±0.0003			
Turning	±0.05	±0.002			

Table 6.3: Assumed geometric parameter errors (m or rad)

Joint i	Position errors			Orientation errors		
	Δx_i	Δy_i	Δz_i	Δp_i	Δr_i	Δs_i
$i=1$	-1.2977e-3	-4.9968e-3	3.76e-4	-8.6303e-4	3.4394e-3	3.5727e-3
$i=2$	3.5675e-3	-1.129e-4	9.8188e-4	5.2392e-4	-5.6013e-4	2.1774e-3
$i=3$	-1.7649e-3	6.5496e-3	-4.0919e-4	3.4179e-4	3.2003e-3	1.7784e-4

6.1 Linear Formulation

For the linear formulation (Figure 6.5), two calibration schemes are simulated. The first one is the position calibration and the second one is the full-pose calibration.

Simulations of the calibration using only position measurements are carried out successfully and usually converge to the specified accuracy by no more than 6 iterations, even though it (Equation (4.6)) is simpler and less complete compared with the full pose calibration. For this case, the root mean square position error (RMSPE) values converged from 3.8×10^{-3} m to 6.2×10^{-7} m by only 4 iterations in Figure 6.6(a). Figure 6.6(b) shows the calibration results.

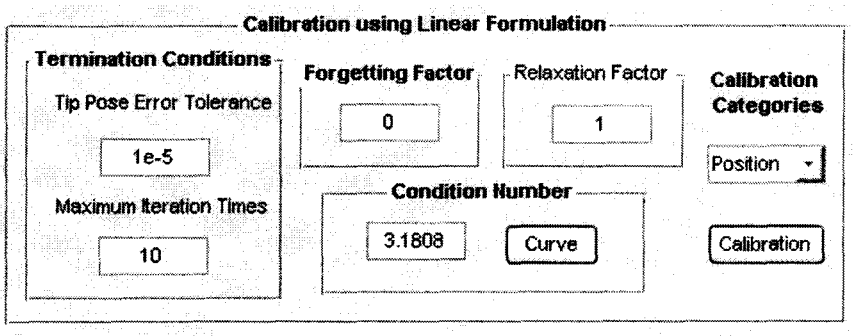
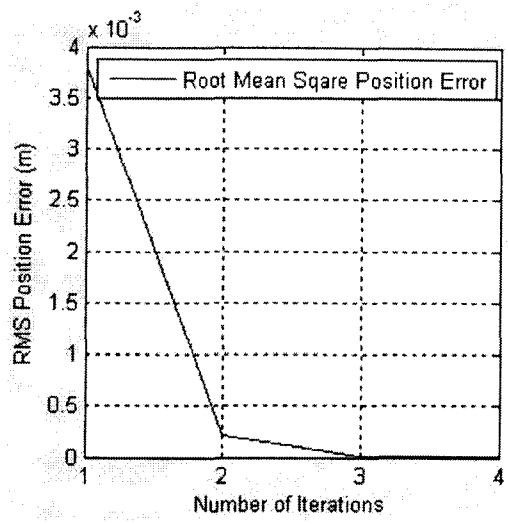
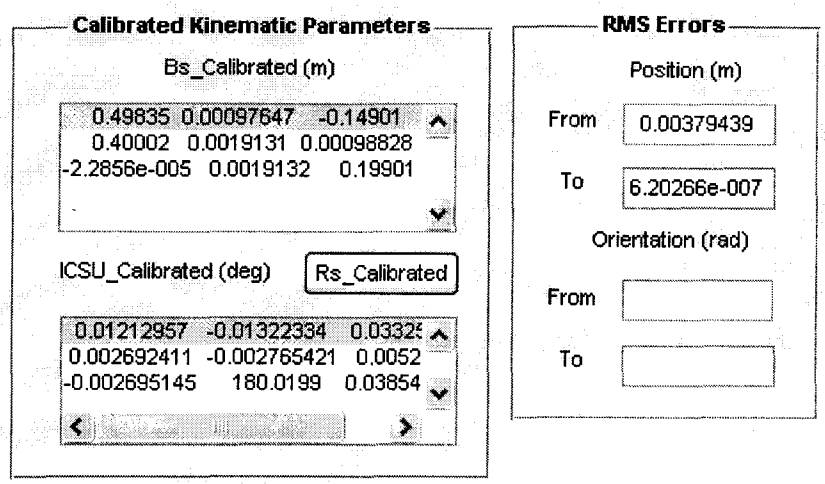


Figure 6.5: Calibration using Linear Formulation panel.

Table 6.4 shows the identified actual kinematic parameters for MRR-1 compared with the nominal values. The positioning accuracy of MRR-1 can be improved when the actual values are implemented into the kinematic model. The differences between the nominal and actual values are the geometric parameters errors, as shown in Table 6.5.



(a)



(b)

Figure 6.6: Partial pose (position) calibration outputs.

With the identified errors in Table 6.5, Table 6.6 indicates that the position of the MRR-1's end-effector after calibration is much closer to the actual position than the nominal values before calibration. However, the differences between the calibrated rotation matrix

and the actual values are found even after calibration in Table 6.6. Clearly, the orientation accuracy is not taken into account in position calibration.

Table 6.4: MRR-1 kinematic parameters before and after calibration

Joint i	Kinematic parameters	Nominal values (m / deg)			Calibrated actual values (m / deg)		
		X	Y	Z	X	Y	Z
1	Local Body	0.5	0	-0.15	0.49835	9.76e-4	-0.1490
2	Vectors	0.4	0	0	0.40002	0.0019131	9.88e-4
3	(m)	0	0	0.2	-2.3e-5	0.0019132	0.19901
		Pitch	Roll	Yaw	Pitch	Roll	Yaw
1	Initial Orientation set up (deg)	0	0	0	0.0121	-0.013	0.0332
2		0	0	0	0.0027	-0.003	0.0052
3		0	180	0	-0.003	180.02	0.0385

Table 6.5: Identified geometric parameter errors (m or rad)

Joint i	Position errors			Orientation errors		
	Δx_i	Δy_i	Δz_i	Δp_i	Δr_i	Δs_i
$i=1$	-1.6453e-3	9.7647e-4	9.8828e-4	2.1170e-4	-2.3079e-4	5.8048e-4
$i=2$	2.2946e-5	1.9131e-3	9.8828e-4	4.6991e-5	-4.8266e-5	9.2338e-5
$i=3$	-2.2856e-5	1.9132e-3	-9.8831e-4	-4.7039e-5	3.4700e-4	6.7282e-4

Table 6.6: End-effector poses before and after calibration

	Tip position	End-effector rotation matrix		
Nominal values	4.330127e-1	0	-1	0
	1.7987616e-4	-1	0	0
	2.5000e-1	0	0	-1
Actual values	4.2722476e-1	-1.8588651e-4	-9.9999914e-1	1.2975389e-3
	6.5018508e-1	-9.9998926e-1	1.7987616e-4	-4.6306960e-3
	2.5310769e-1	4.6304586e-3	-1.2983857e-3	-9.9998844e-1
Calibrated values	4.2722582e-1	-6.7281084e-4	-9.9999974e-1	2.4911461e-4
	6.5018489e-1	-9.9999969e-1	6.7270681e-4	-4.1745853e-4
	2.5310762e-1	4.1729084e-4	-2.4939540e-4	-9.9999988e-1

On the other hand, due to the limitations of the linear formulation, simulations have shown that these limitations tend to cause the full-pose calibration results to be non-convergent, as shown in Figure 6.7. The root mean square values of position, orientation, and pose errors are represented by blue, red, and black curves, respectively, in Figure 6.7.

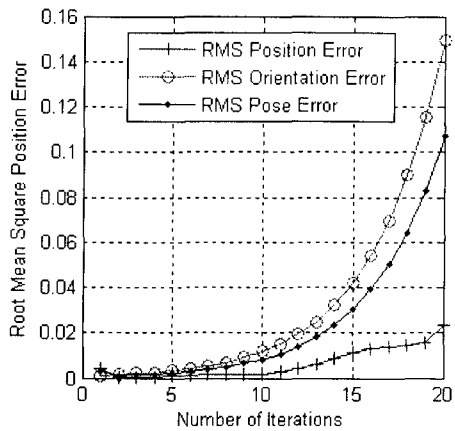


Figure 6.7: Non-convergent results for the full-pose calibration.

6.2 Nonlinear Formulation

For the nonlinear formulation using GA, the options of GA need to be adjusted after several trials in order to obtain a better performance, consisting of the number of populations, generations, crossover factors, initial populations, and weighting factors etc. Figure 6.8 shows the GUI for the nonlinear formulation using GA.

With 50 populations and around 250 generations, the best fitness value has changed from about 4.5×10^{-3} to 9.8075×10^{-6} (Figure 6.9), indicating that the norm of the pose error of robot end-effector has been decreased below the satisfied accuracy, 1×10^{-5}. The best and

mean fitness values of each generation are represented by black and blue points. Figure 6.9 also indicates that GA's initial convergence speed is very fast and then gradually becomes slow. Generally, GA needs longer computing time than other optimization methods.

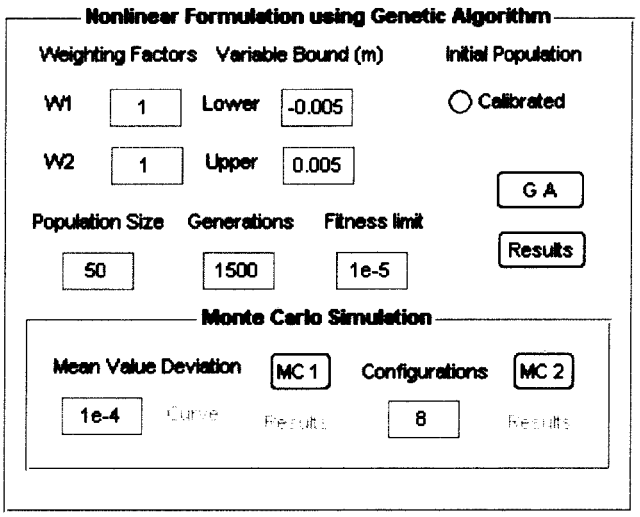
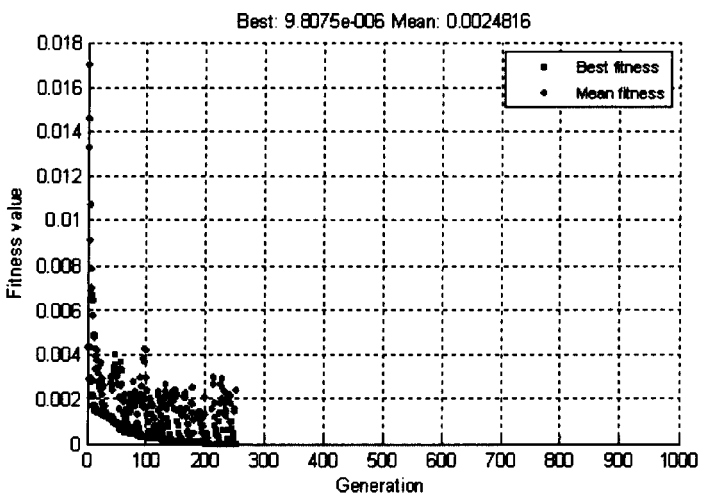
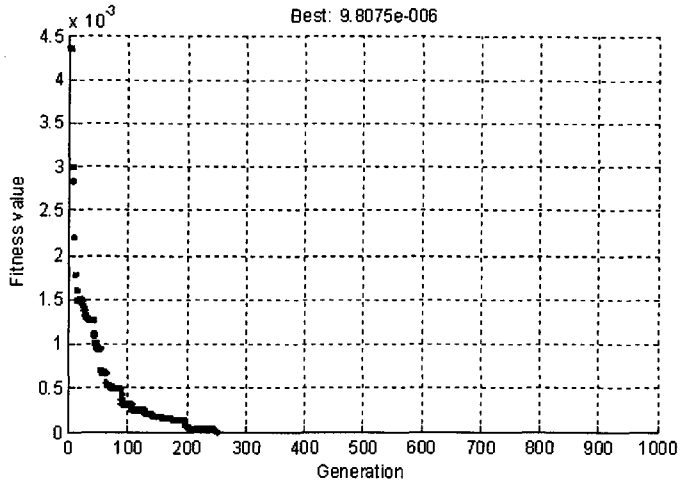


Figure 6.8: Nonlinear Formulation using Genetic Algorithm panel in the GUI.



(a)



(b)

Figure 6.9: (a) Best and Mean fitness; (b) Best fitness.

Table 6.7 shows the MRR-1 kinematic parameters before and after calibration using the nonlinear formulation. The satisfactory results of the end-effector pose after calibration are presented in Table 6.8, as compared to the pure position calibration values using the linear formulation in Table 6.6. The calibrated rotation matrix of the end-effector in Table 6.8 is much closer to the actual values compared with the one in Table 6.6, indicating the better performance of the nonlinear formulation using GA.

The final best individual with 18 variables for the MRR-1 is shown in Table 6.9 after 250 generations. They are transformed to be the identified kinematic parameter errors and implemented into the robot kinematic model.

Table 6.7: MRR-1 kinematic parameters before and after calibration

Joint i	Kinematic parameters	Nominal values (m/deg)			Calibrated values (m/deg)		
		X	Y	Z	X	Y	Z
1	Local Body	0.5	0	-0.15	4.9647e-1	2.9079e-3	-1.4903e-1
2	Vectors	0.4	0	0	3.9975e-1	1.9099e-3	5.7817e-4
3	(m)	0	0	0.2	-8.6127e-4	1.0051e-3	1.9903e-1
		Pitch	Roll	Yaw	Pitch	Roll	Yaw
1	Initial	0	0	0	1.3983e-2	-1.6551e-1	8.3171e-2
2	Orientation	0	0	0	-7.2005e-2	-8.2925e-2	-1.5142e-1
3	set up (deg)	0	180	0	-9.0582e-3	1.8008e+2	9.9070e-3

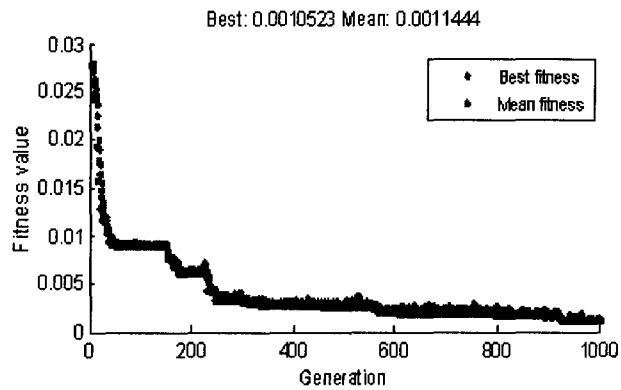
Table 6.8: End-effector poses before and after calibration

	Tip position	End-effector rotation matrix		
Nominal values	4.330127e-1	0	-1	0
	6.5000e-1	-1	0	0
	2.5000e-1	0	0	-1
Actual values	4.2722476e-1	-1.8588651e-4	-9.9999914e-1	1.2975389e-3
	6.5018508e-1	-9.9998926e-1	1.7987616e-4	-4.6306960e-3
	2.5310769e-1	4.6304586e-3	-1.2983857e-3	-9.9998844e-1
Calibrated values	4.2722444e-1	-1.8027019e-4	-9.9999915e-1	1.2921556e-3
	6.5018405e-1	-9.9998925e-1	1.7428246e-4	-4.6325272e-3
	2.5310623e-1	4.6322980e-3	-1.2929769e-3	-9.9998843e-1

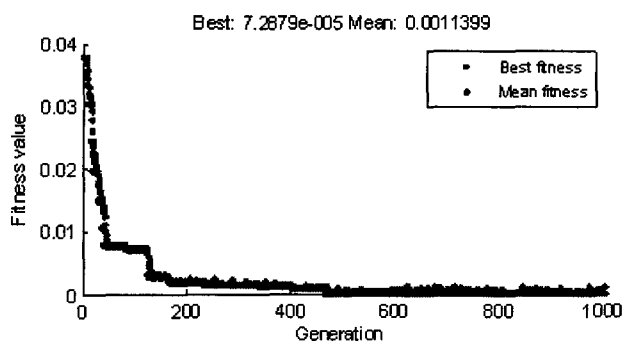
Table 6.9: Geometric parameter errors from the final best fit individual

Joint i	Local Body Vectors Errors (m)			Zero Reference Orientation set up Errors (deg)		
	x	y	z	Pitch (p)	Roll (r)	Yaw (s)
1	-3.5280e-3	2.9079e-3	9.7299e-4	2.4404e-4	-2.8887e-3	1.4516e-3
2	-2.4508e-4	1.9099e-3	5.7817e-4	-1.2567e-3	-1.4473e-3	-2.6428e-3
3	-8.6127e-4	1.0051e-3	-9.6915e-4	-1.5809e-4	1.4089e-3	1.7291e-4

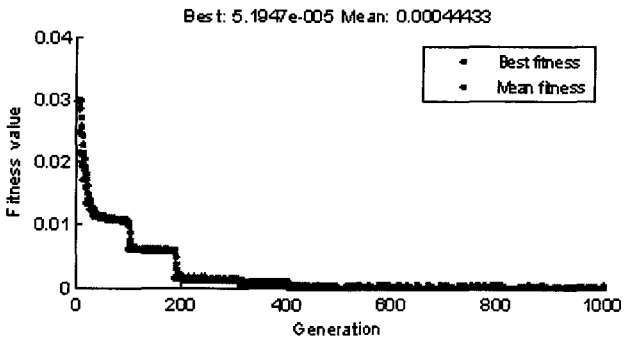
Basically, the better performance can be expected with a larger population and more generations, although it increases computing time. Figure 6.10 illustrates the different performance for different population and generations. With the same 1000 generations, the final best fitness is improved to 0.0010523, 7.2879×10^{-5}, and 5.1947×10^{-5} for 100, 150, and 200 populations, respectively. With the same 200 population, the final best fitness is enhanced to 5.1947×10^{-5}, and 1.4651×10^{-5} for 1000 and 1500 generations, respectively.



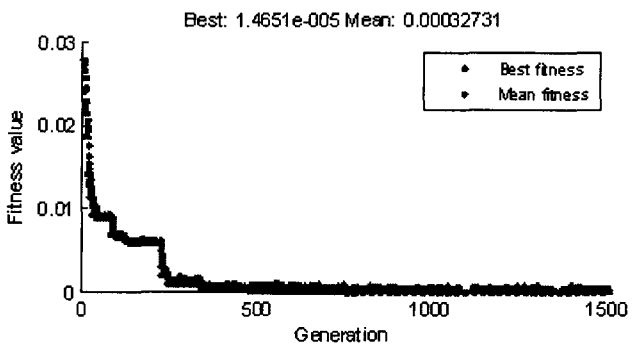
(a) Population 100, Generation 1000;



(b) Population 150, Generation 1000;



(c) Population 200, Generation 1000;



(d) Population 200, Generation 1500;

Figure 6.10: Better performance with large populations and generations.

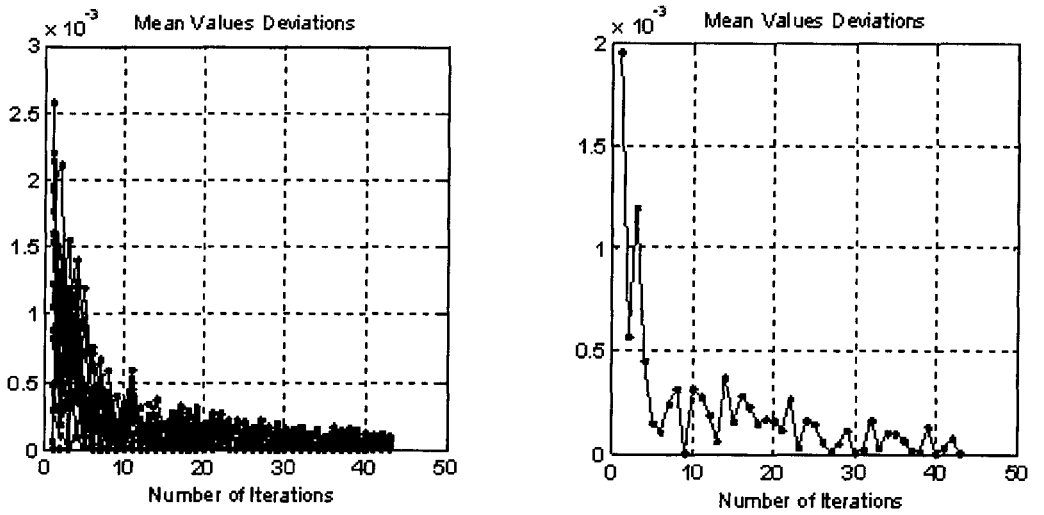
Furthermore, a high crossover probability and a low mutation probability are proved to be more likely to obtain a good performance [61]. Since GA is a stochastic process, each time the program is run, slightly different results are returned. From simulations, it is shown that satisfactory results are generated and enough accuracy of position and orientation is guaranteed after calibration.

6.3 Monte Carlo Simulation

In order to reduce the stochastic random error in GA, Monte Carlo simulations (Figure 6.8) in the loop 2 (Figure 5.1) are carried out by repeatedly running GA in loop 1 to calculate the mean values of $\Delta \mathbf{g}$ until they become stable and fluctuate within certain bounds. The standard deviations around the mean value are calculated each time when new results are generated from GA, and a convergent pattern is found and illustrated in Figure 6.11.

Since there are 18 parameters for the MRR-1 robot in total, the standard deviations of the mean values for all parameters are plotted together in one chart (Figure 6.11a) and one of them is clearly shown with 44 iterations (Figure 6.11b). The final stable mean value of all the best fit individuals from 44 GA runs is calculated and shown in the Table 6.10.

Furthermore, in order to obtain a global solution, the Monte Carlo simulation is conducted again repeatedly for 8 random configurations in the robot workspace (Figure 6.8). Random movements for all three joints are generated with a uniform distribution within joint motional bounds (Figure 6.13, 6.14). The results in Figure 6.12 show that all the 16 parameters do not fluctuate significantly and remain almost stable in the 8 random configurations over the robot's workspace, which indicates the global property of the solution from Monte Carlo level 1.



(a) (b)

Figure 6.11: (a) Standard deviations of mean values of all the parameter errors $\Delta \mathbf{g}$ from GA; (b) one of the parameters.

Table 6.10: The stable mean value of all best fit individual from GA runs

Joint i	x	y	z	P	r	s
1	-1.5468941 e-3	-4.2133711 e-4	-3.4040037 e-5	-6.5581664 e-4	-2.7366316 e-4	-3.9937420 e-4
2	-1.1226324 e-3	9.9691850 e-4	3.6835343 e-4	-3.7540076 e-005	-8.0672008 e-4	1.3481633 e-3
3	2.0275948 e-3	9.1848455 e-4	9.1167962 e-4	-2.4716377 e-004	1.1293914 e-3	4.2085826 e-3

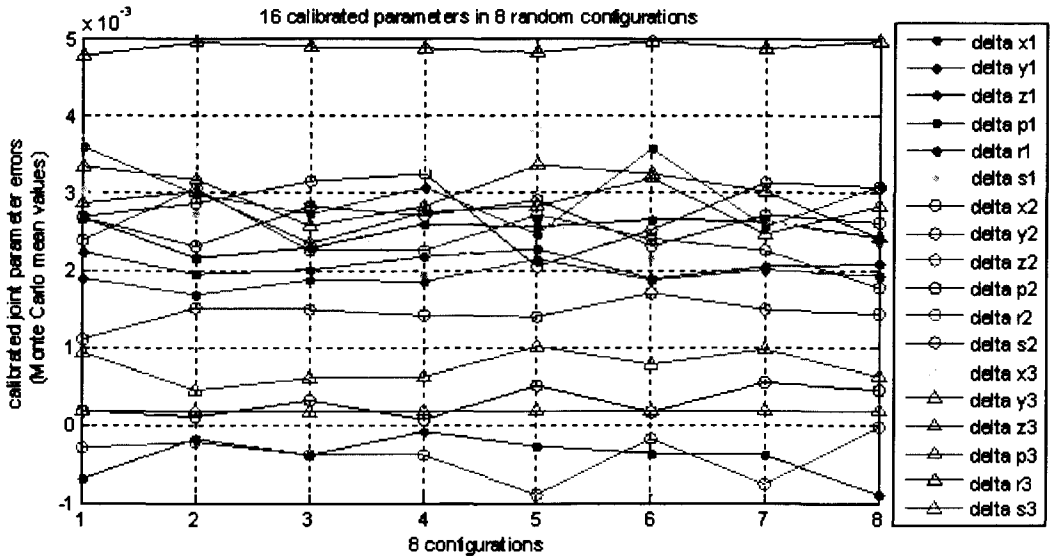


Figure 6.12: 16 calibrated parameters in 8 random configurations.

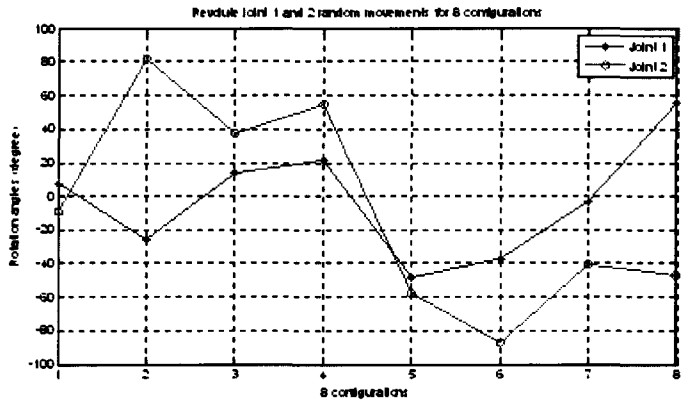


Figure 6.13: Revolute joint 1 and 2 random movements for 8 configurations.

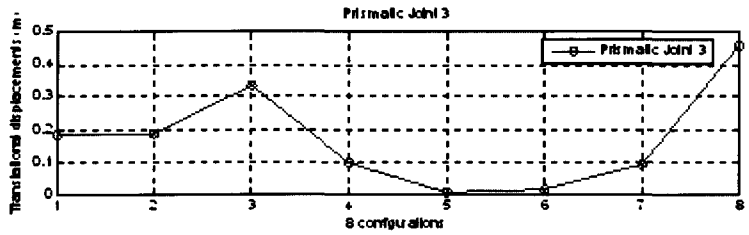


Figure 6.14: Translational displacements of the prismatic joint 3 for 8 configurations.

CHAPTER 7 CALIBRATION CONSIDERING RECONFIGURATION

The chapter is to develop a method to perform calibration when considering reconfiguration for an MRR. A path matrix is introduced to represent the sequence change of modules. A snap point is introduced first before the path matrix. Furthermore, a self-calibration methodology is proposed for future research on MRRs.

7.1 Snap Point

A snap point is the point on the end of each module at which the next module is connected [11]. As shown in Figure 7.1, the snap point can be regarded as the common point between two adjacent modules. According to Equation (3.41), the position of the *i*th snap point can be expressed as:

$$\mathbf{s}_i = \mathbf{b}_1 + \mathbf{R}_{01}(\mathbf{b}_2{}' + \cdots(\mathbf{b}_{i-2}{}' + \mathbf{R}_{i-3i-2}(\mathbf{b}_{i-1}{}' + \mathbf{R}_{i-2i-1}\mathbf{b}_i{}'))) \tag{7.1}$$

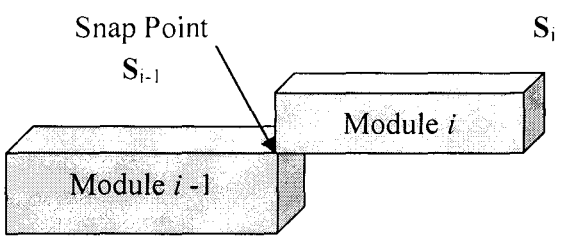


Figure 7.1: Snap point between two adjacent modules.

Based on Equation (7.1), snap points can be computed recursively:

$$\mathbf{s}_i = \mathbf{s}_{i-1} + \mathbf{b}_i = \sum_{k=1}^{i} \mathbf{b}_k = \sum_{k=1}^{i} \mathbf{R}_k \mathbf{b}_k' = \sum_{k=1}^{i} \prod_{j=1}^{k} \mathbf{R}_{j-1j} \mathbf{b}_k' \tag{7.2}$$

where $\mathbf{b}_k$ and $\mathbf{b}_k'$ are the body vector representing the kth module in global and local coordinates, respectively, and $\mathbf{s}_i$ is the vector representing its snap point.

7.2 Path Matrix

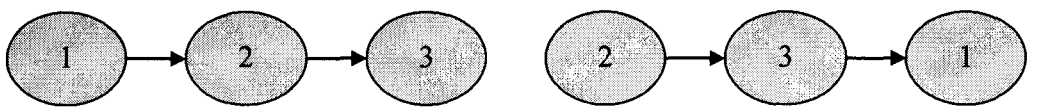
The kinematics equations can be generated for a system with the defined sequence of the connecting modules given in Chapter 3. However, for a reconfigurable system, this sequence is subject to change and so is the number of modules. To account for this change, the path matrix [64] is applied. A path matrix is used to define the connectivity of the modules in matrix form as below:

$$\mathbf{T} = \begin{array}{c} \begin{matrix} B_1 & B_2 & B_3 \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} J_1 \\ J_2 \\ J_3 \end{matrix}, \tag{7.3}$$

where the rows correspond to the joints as indicated by letter J; the columns correspond to the bodies as indicated by letter B; and the subscript number indicates the body number. The component values of the matrix are either 1 or 0. $\mathbf{T}_{ij} = 1$ if joint i is in the route from B_i to B_j, meaning that the motion of body B_i contributes to that of body B_j. If not in the route, $\mathbf{T}_{ij} = 0$.

The matrix given in Equation (7.3) is for the original system shown in Figure 7.2(a). The diagonal values are 1 indicating that the joints are associated with their own bodies. Furthermore, $\mathbf{T}_{12} = 1$, as joint 1 is in the route to body 2; $\mathbf{T}_{13} = 1$ and $\mathbf{T}_{23} = 1$ as joint 1 and 2 are in the route to body 3; the rest are zero. If the original system in Figure 7.2(a) is reconfigured to the system shown in Figure 7.2(b), then the path matrix of Equation (7.3) is changed to:

$$\mathbf{T}=\begin{array}{c} \begin{matrix} B_1 & B_2 & B_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \begin{matrix} J_1 \\ J_2 \\ J_3 \end{matrix} \end{array} \tag{7.4}$$



(a) (b)

Figure 7.2: (a) Original connection (b) Reconfigured connection

By re-ordering the matrix of Equation (7.4) into an upper triangle form, it becomes:

$$\mathbf{T}=\begin{matrix}\begin{matrix}B_2 & B_3 & B_1\end{matrix}\\ \begin{bmatrix}1 & 1 & 1\\ 0 & 1 & 1\\ 0 & 0 & 1\end{bmatrix}\end{matrix}\begin{matrix}J_2\\ J_3\\ J_1\end{matrix} \tag{7.5}$$

The sequence of the body indicated by the column headings in the matrix of Equation (7.5) is the true sequence for the reconfigured system.

In general, the relationship between the path matrix and the snap points can be expressed:

$$\mathbf{S} = \mathbf{T}^T \mathbf{H}, \tag{7.6}$$

where $\mathbf{S} = [\mathbf{s}^T_1, \mathbf{s}^T_2,...\mathbf{s}^T_n]^T$ are the snap points with the number indicating the true sequence of the system; and $\mathbf{H} = [\mathbf{h}^T_1, \mathbf{h}^T_2,... \mathbf{h}^T_n]^T$ are the vectors of each body with the number indicating the body numbers.

In the light of Equation (7.6), the snap points of the original system can be determined as:

$$\begin{bmatrix}\mathbf{s}_1\\ \mathbf{s}_2\\ \mathbf{s}_3\end{bmatrix}=\begin{bmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}\mathbf{h}_1\\ \mathbf{h}_2\\ \mathbf{h}_3\end{bmatrix}=\begin{bmatrix}\mathbf{h}_1\\ \mathbf{h}_1+\mathbf{h}_2\\ \mathbf{h}_1+\mathbf{h}_2+\mathbf{h}_3\end{bmatrix} \tag{7.7}$$

For the reconfigured system it becomes:

$$\begin{bmatrix}\mathbf{s}_1\\ \mathbf{s}_2\\ \mathbf{s}_3\end{bmatrix}=\begin{bmatrix}1 & 0 & 0\\ 1 & 1 & 0\\ 1 & 1 & 1\end{bmatrix}\begin{bmatrix}\mathbf{h}_2\\ \mathbf{h}_3\\ \mathbf{h}_1\end{bmatrix}=\begin{bmatrix}\mathbf{h}_2\\ \mathbf{h}_2+\mathbf{h}_3\\ \mathbf{h}_1+\mathbf{h}_2+\mathbf{h}_3\end{bmatrix} \tag{7.8}$$

Note that the order of matrix **H** is changed according to the new path matrix and matched with the column headings. Hence, the utilization of the path matrix provides a means to relate the true sequence of the bodies for calibration of a MRR.

7.3 Simulations

Here is the calibration simulation for the MRR-1 after reconfiguration that corresponds to Figure 7.2(b). Figure 7.3 is the GUI where the sequence of the existing modules in a new configuration can be rearranged. Figure 7.4 illustrates the robot configuration before and after changing the sequence of the modules connection.

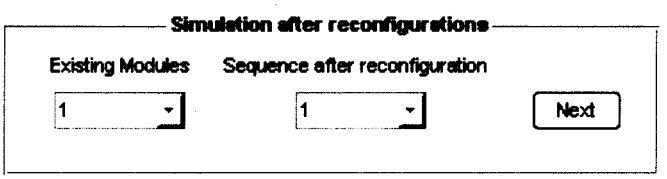


Figure 7.3: Simulation when reconfiguration.

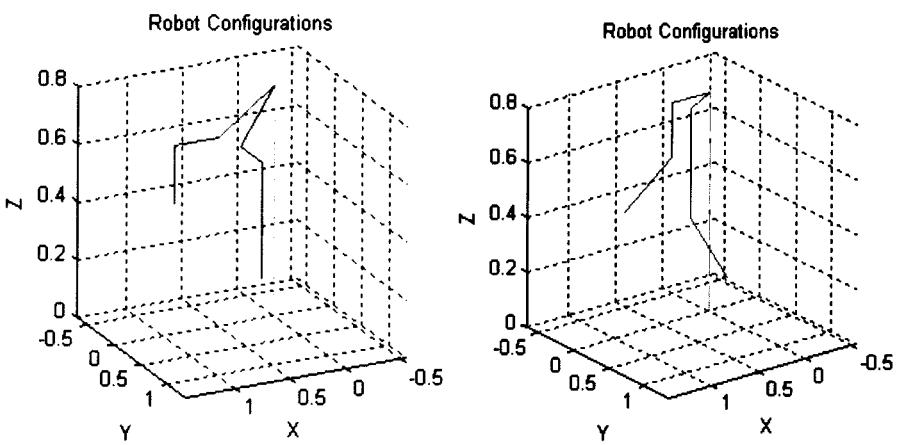


Figure 7.4: Robot configuration before and after reconfiguration.

Figure 7.5 shows the satisfactory convergent results using GA: the best fitness steadily declines from about 0.005 to 9.4893×10^{-6} after around 150 generations.

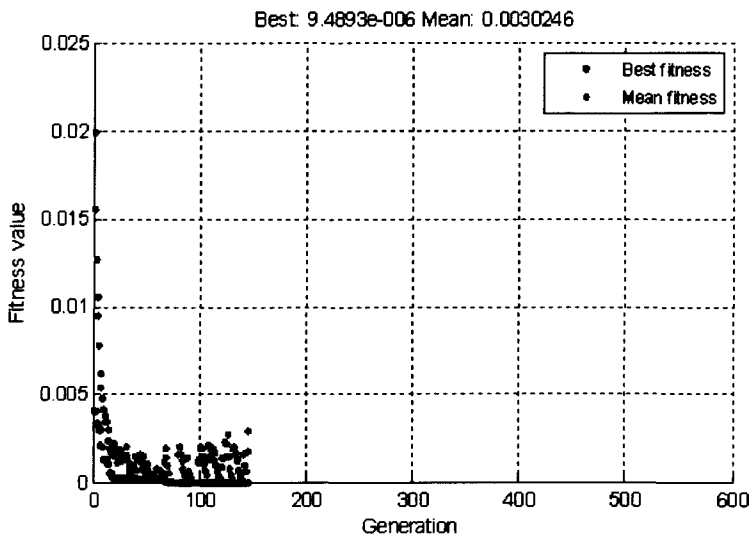


Figure 7.5: GA results for MRR-1 after reconfiguration

7.4 Self-calibration Formulation for MRRs

Assuming there are two fixed and arbitrary points (or targets) in the workspace of an MRR, whose positions $\mathbf{p}_1$ and $\mathbf{p}_2$ are both unknown, but the relative distance between them is precisely provided before the calibration:

$$|\mathbf{p}_1 - \mathbf{p}_2| = d \tag{7.9}$$

A pointing tool is rigidly attached on the end-effector of an MRR, such as a laser pointer. Then the MRR is moved and adjusted in order to aim the pointing tool at the two fixed

targets one by one; meanwhile, the readings of joint sensors ($\mathbf{q}_m$) are recorded for each configuration. Assuming an additional prismatic joint from the pointer tool to the fixed target, the target can be actually treated as the fixed tip of the virtual prismatic joint.

Hence, for the two targets (or virtual tips), their positions can be denoted by

$$\mathbf{p}_1 = f(\mathbf{q}_{o1}, \mathbf{g}_o) \tag{7.10a}$$

$$\mathbf{p}_2 = f(\mathbf{q}_{o2}, \mathbf{g}_o) \tag{7.10b}$$

where $\mathbf{q}_{o1}$ and $\mathbf{q}_{o2}$ are the nominal joint movements for two targets; and $\mathbf{q}_o$ is adjusted to aim the pointer at the fixed target.

From Equation (7.9) and (7.10), the preset distance should be equal to

$$|\mathbf{p}_1 - \mathbf{p}_2| = |f(\mathbf{q}_{o1}, \mathbf{g}_o) - f(\mathbf{q}_{o2}, \mathbf{g}_o)| = d \tag{7.11}$$

However, the recorded readings of all joint sensors ($\mathbf{q}_m$) are slightly different from the nominal values ($\mathbf{q}_o$), and this leads to the nonlinear fitting formulation of self-calibration, which uses only joint readings without external measuring devices:

$$|\mathbf{p}_1 - \mathbf{p}_2| = |f(\mathbf{q}_{m1}, \mathbf{g}_o+\Delta\mathbf{g}) - f(\mathbf{q}_{m2}, \mathbf{g}_o+\Delta\mathbf{g})| \rightarrow d \tag{7.12}$$

where $\Delta\mathbf{g}$ is the set of geometric parameter errors in the robot model, to be identified and fitted into nonlinear Equation (7.12) to ensure the values of $|\mathbf{p}_1 - \mathbf{p}_2|$ are close enough to the preset accurate distance d.

Moreover, this self-calibration idea might be further extended by setting two full-pose targets, such as three-point targets, which can provide both position and orientation, instead of position only, for fixed point targets. The relative pose (**P**) of these two pose targets should also be precisely established in advance. The MRR should then be moved to match the end-effector with the preset full-pose targets one by one, and the joint sensor readings should also be recorded in the meantime. Hence, similar nonlinear formulation might be developed:

$$|\mathbf{X}_1 - \mathbf{X}_2| = |\, f(\mathbf{q}_{m1}, \mathbf{g}_o + \Delta\mathbf{g}) - f(\mathbf{q}_{m2}, \mathbf{g}_o + \Delta\mathbf{g})| \rightarrow P \tag{7.12}$$

CHAPTER 8 CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

In this thesis, both linear and nonlinear kinematic calibration methods are investigated and compared for MRRs. Linear position calibration has proved to be a quick convergent method, normally, by less than 6 iterations. However, only the position of the end-effector is considered rather than full-pose accuracy. Linear full-pose calibration appears to be non-robust and susceptible to many aspects during calibration, which may cause divergent and singularity issues. On the other hand, nonlinear calibration using genetic algorithm has been demonstrated to be a robust method, although it usually takes a few minutes to converge to a required accuracy. Both position and orientation accuracies are improved successfully after the nonlinear calibration.

8.2 Contributions

The following contributions are made in this thesis:

- Robot kinematic calibration based on the nonlinear formulation is developed and implemented with genetic algorithm (GA). Without using the PRY angles to represent the tip orientation in the nonlinear formulation, it avoids the quadrant sensitivity problem by using the rotation matrix directly. A Euclidean norm, a

spectral norm and weighting factors are used to unify the position and orientation errors of the end-effector.

- Linear full-pose calibration is discussed in four categories, where some downsides are found. These downsides include quadrant sensitivity problem and orthogonality sacrifice issues.
- Monte Carlo simulation is implemented in the calibration simulations. It includes two levels of Monte Carlo simulation; one in single configuration and another one in 8 random configurations.

8.3 Future Work

First of all, attention should be paid to improve the efficiency of the nonlinear identification with GA, especially for MRR robots with a large number of modules. Some better results identified by the linear formulation may be considered for initial populations. Good initial populations dramatically increase the efficiency of GA. Secondly, although the nonlinear formulation has been proposed by several investigators in similar or slightly different ways, different methods can be found in fitting this nonlinear regression model, such as the Levenberg-Marquardt algorithm, genetic algorithms and neural networks. Hence, another future work would be to compare those different methods through convergence and robustness. Thirdly, a self-calibration approach for MRRs with the formulation in the Chapter 7 should be another important

future work to carry out. Last but not least, only simulations were provided to test the calibration methods in this thesis; experiments should be conducted to compare with the simulation results.

APPENDIX A (TAIT) BRYAN ANGLE (PITCH, ROLL, YAW, PRY) [18]

$$R(\theta_x)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_x & -\sin\theta_x\\ 0 & \sin\theta_x & \cos\theta_x\end{bmatrix}$$

$$R(\theta_y)=\begin{bmatrix}\cos\theta_y & 0 & \sin\theta_y\\ 0 & 1 & 0\\ -\sin\theta_y & 0 & \cos\theta_y\end{bmatrix}$$

$$R(\theta_z)=\begin{bmatrix}\cos\theta_z & -\sin\theta_z & 0\\ \sin\theta_z & \cos\theta_z & 0\\ 0 & 0 & 1\end{bmatrix}$$

In the body fixed frame

$$R=R(\theta_x)R(\theta_y)R(\theta_z)$$

$$R=\begin{bmatrix}c\theta_y c\theta_z & -c\theta_y s\theta_z & s\theta_y\\ s\theta_x s\theta_y c\theta_z+c\theta_x s\theta_z & -s\theta_x s\theta_y s\theta_z+c\theta_x c\theta_z & -s\theta_x c\theta_y\\ -c\theta_x s\theta_y c\theta_z+s\theta_x s\theta_z & c\theta_x s\theta_y s\theta_z+s\theta_x c\theta_z & c\theta_x c\theta_y\end{bmatrix}$$

Relation of PRY with R

$$\theta_y=\cos^{-1}\left(\left[(r_{23})^2+(r_{33})^2\right]^{1/2}\right)$$

$$\theta_x=\cos^{-1}\left(r_{33}/\cos\theta_y\right)$$

$$\theta_z=\cos^{-1}\left(r_{11}/\cos\theta_y\right)$$

REFERENCE

[1] American National Standards Institute, "ANSI A15 Standards on Finding Target Features With Optical Equipment."

[2] M., Verner, F., Xi, and C., Mechefske, "Optimal Calibration of Parallel Kinematic Machines," ASME Trans. Mechanical Design. Vol. 127, 2005, pp. 62-69.

[3] A. Goswami, A. Quaid, and M. Peshkin, "Complete Parameter Identification of a Robot from Partial Pose Information", 1993 IEEE Int. Conf. on Robotics and Automation.

[4] J. O. Berg, Ind. Robot 18, 1991, pp. 29-31.

[5] J. M. Hollerbach, "A survey of kinematic calibration," The Robotics Review, O., Khatib, J. J., Craig, and T., Lozano-Perez (Eds), MIT Press, Cambridge, MA, 1989, pp. 206-242.

[6] Z. S. Roth, B. W. Mooring, and B. Ravani, "An overview of robot calibration," IEEE Journal of Robotics and Automation RA-3, No. 5, Oct. 1887, pp. 377-385.

[7] B. Liu, C. L. Liang, L. J. Xue, S. H. Hu, and Y. S. Jiang, "Calibration of a steward parallel robot using genetic algorithm," Proc. of the 2007 IEEE Int. Conf. on Mechatronics and Automation, Aug. 5-8, 2007, pp. 2495-2500.

[8] Engineering Service Inc., http://www.esit.com/frame_automation.htm1.

[9] D. Duff, K. Roufas, M. Yim, W. Zhang, and S. Homans, "Modular Reconfigurable Robots in Space Applications," Proc. of the IEEE Int. Conf. on Robots and Automation, Mar. 2001, pp. 235-237.

[10] N. Bajaj, "Design of a Modular and Reconfigurable Robot for Space and Manufacturing Applications", AER070 Design / Thesis Project - Final Report, Dept. of Aerospace Eng., Ryerson Univ., Apr. 04, 2005, pp. 1, 3-4, 14-15.

[11] S. J. Qiang, "Modeling and Simulation of Reconfigurable Systems with Application to the Polishing Process", MASc. Thesis, Ryerson Univ., 2004.

[12] K. Tomita, S. Murata, E. Yoshida, H. Kurokawa, and S. Kokaji. "Reconfiguration Method for a Distributed Mechanical System", Distributed Autonomous Robotic System, Vol. 2, 1996, pp. 17-25.

[13] J. Michael. "Fractal Shape Changing Robot Construction Theory & Application Note", Robodyne Cybernetics Ltd, 1995.

[14] M. Yim. "Locomotion with a Unit-modular Reconfigurable Robot", Ph.D. Thesis, Stanford Univ., 1994.

[15] C. Unsal, H. Kiliccote, M. Patton, and P. Khosla. "Motion Planning for a Modular Self-reconfiguring Robotic System", Distributed Autonomous Robotic Systems, 4, 2000.

[16] R. P. Paul, "Robot manipulators: mathematics, programming, and control," MIT,

Cambridge, MA, 1981.

[17] B. W. Mooring, A. A. Roth, and M. R. Driels, "Fundamentals of manipulator calibration," Wiley, New York, 1991.

[18] F. Xi, "Computational Dynamics lecture notes", Ryerson Univ., Toronto, ON. Canada, c2007.

[19] L. J. Everett, M. Driels, and B. W. Mooring, "Kinematic modeling for robot calibration," Proc. of IEEE Int. Conf. of Robotics and Automation, 1987, pp. 183-189.

[20] B. Karan, and M. Vukobratovic, "Calibration and accuracy of manipulation robot models – an overview," Mechanism and Machine Theory, Vol. 29, No. 3, 1994, pp. 479-500.

[21] J. J. Craig, "Introduction to Robotics: Mechanics and Control," Third Edition, Pearson Education, 2005.

[22] S. A. Hayati, "Robot Arm Geometric Link Parameters Estimation", Proc. of 22nd IEEE Conf. of Decision and Control, San Antonio, Dec. 14-16, 1983, pp. 1477-1483.

[23] J. Angeles, "On the Numerical Solution of the Inverse Kinematic Problem," The Int. Journal of Robotics Research, Vol. 4, No. 2, MIT, 1985, pp. 21-37.

[24] C. Mavroidis, S. Dubowsky, P. Drouet, J. Hinterstreiner, and J. Flanz, "A

systematic error analysis of robotic manipulators: application to a high performance medical robot," Proc. of the 1997 IEEE Int. Conf. on Robotics and Automation, Albuquerque, New Mexico, April 1997, pp. 980-985.

[25] U. S. Pathre and M. R. Driels, "Simulation experiments in parameter identification for robot calibration," The Int. Journal of Advanced Manufacturing Technology, 1990, pp. 13-33.

[26] H. P., Schwefel, "Numerical Optimization of Computer Models", Wiley, New York, 1981.

[27] L. E. Scales, "Introduction to Non-Linear Optimization", Springer-Verlag, New York, 1985.

[28] F. S., Acton, "Numerical Methods that Work", Harper and Row, New York, 1970.

[29] J. Denavit, and R. S. Hartenberg, "A Kinematic Notation for Lower Pair Mechanisms Based on Matrices", ASME Journal of Applied Mechanics, June 1955, pp. 215-221.

[30] R. Manseur, "Robot Modeling and Kinematics", First Edition, Da Vinci Engineering Press, 2006.

[31] C. Wu, "The kinematic error model for the design of robot manipulators," Proc. of the American control conf., San Francisco, California, 1983, pp. 497-502.

[32] C. Wu, "A Kinematic CAD Tool for the Design and Control of a Robot

Manipulator", The Int. Jour. of Robotics Research, Vol. 3, No. 1, spring 1984, pp. 58-67.

[33] B. W. Mooring, "The Effect of Joint Axis Misalignment on Robot Positioning Accuracy", Proc. Computers in Engineering Conf. and Exhibit, 1983, pp. 151-155.

[34] T. W. Hsu and L. J. Everett, "Identification of the Kinematic Parameters of a Robot Manipulator for Positional Accuracy Improvement", Proc. 1985 Computers in Engineering Conf. and Exhibition, vol. 1, 1985, pp. 263-267.

[35] R. Ibarra and N. D. Perreira, "Determination of Linkage Parameter and Pair Variable Errors in Open Chain Kinematic Linkages using a Minimal set of Pose Measurement data", ASME Jour. of Mechanisms, Transmissions, Automation in Design, June 1986, pp. 159-166.

[36] C. S. Gatla, R. Lumia, J. Wood, and G. Starr, "An Automated Method to Calibrate Industrial Robots Using a Virtual Closed Kinematic Chain", IEEE Trans. on Robotics, vol. 23, on. 6, Dec. 2007, pp. 1105-1116.

[37] C. Lightcap, S. Hamner, and T. Schmitz, and S. Banks, "Improved Positioning Accuracy of the PA10-6CE Robot with Geometric and Flexibility Calibration", IEEE Trans. on Robotics, vol. 24, no 2, April, 2008, pp. 452-456.

[38] H. Q. Zhuang, L. K. Wang, and Z. S. Roth, "Error-Model-Based Robot Calibration using a Modified CPC Model", Robotics & Computer-Integrated Manufacturing.

Vol. 10, no. 4, 1993, pp. 287-299.

[39] B. W. Mooring and G. R. Tang, "An Improved Method for Identifying the Kinematic Parameters in a Six-axis Robot", Proc. 1984 Int. Computers in Engineering Conf. and Exhibit, vol. 1, 1984, pp. 79-84.

[40] D. E. Whitney, C. A. Lozinski, and J. M. Rourke, "Industrial Robot Forward Calibration Method and Results", ASME Jour. Dynamic Syst., Meas. Contr., vol. 108, Mar. 1986, pp. 1-8.

[41] J. Chen, C. B. Wang, and J. C. S. Yang, "Robot Positioning accuracy improvement through kinematic parameter identification", Proc. 3rd Canadian CAD/CAM and Robotics Conf., Toronto, June19-22, 1984, pp. 4.7-4.12.

[42] J. Chen, and L. M. Chao, "Positioning Error Analysis for Robot Manipulators with All Rotary Joints", Proc. IEEE Int. Conf. Robotics and Automation, San Francisco, April 7-10, 1986, pp. 1011-1016.

[43] H. W. Stone, A. C. Sanderson, and C. P. Neuman, "Arm signature identification", Proc. 1986 IEEE Int. Conf. on Robotics and Automation (San Francisco, CA, Ap. 1986), vol. 1, 1986, pp. 41-48.

[44] L. M. Chao, and J. C. S. Yang, "Development and Implementation of a Kinematic Parameter Identification Technique to Improve the Positioning Accuracy of Robots", Robots 10 Conference Proceeding, Chicago, April 20-24, 1986, pp. 11-69

– 11-81.

[45] M. R. Driels, and U. S. Pathre, "Significance of Observation Strategy on the Design of Robot Calibration Experiments", Journal of Robotics Systems 7(2), June, 1990, pp. 197-223.

[46] C. H. Menq, and J. H. Borm, "Estimation and Observability Measure of Parameter Errors in a Robot Kinematic Model", Proc. Of USA-Japan Symposium on Flexible Automation, pp. 65-70, Minneapolis, Minnesota, July, 1988.

[47] J. H. Borm, and C. H. Menq, "Determination of Optimal Measurement Configurations for Robot Calibration Based on Observability Measure", the Int. Journal of Robotics Research, Vol. 10, No. 1, Feb. 1991, pp. 51-63.

[48] E. O. Doebelin. "Measurement Systems: Application and Design". McGraw-Hill, New York, 1983.

[49] G. Duelen and K. Schroer, "Robotics and Computer Integrated Manufacturing", 1991, pp. 223-231.

[50] R. P. Judd and A. B. Knasinski, IEEE Trans. Robotics and Automation 6, 1990, pp. 20-30.

[51] J. L. Caenen and J. C. Angue, Proc. 1990 IEEE Conf. Robotics and Automation, 1990, pp. 1032-1037.

[52] J. F. Jarvis, Proc. 1988 IEEE Int. Conf. Robotics and Automation, 1988, pp. 635-

640.

[53] M. R. Driels and U. S. Pathre, IEEE Trans. Robotics and Automation, 7, 1991, pp. 351-360.

[54] C. H. An, C. G. Atkenson, and J. M. Hollerbach, "Model-Based Control of a Robot Manipulator", MIT Press, Cambridge, MA, 1988, pp. 62.

[55] F. Xi, D. Nancoo, and G. Knopf, "Total lease-Squares methods for active view registration of three-dimensional line laser scanning data," ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 127, March 2005, pp. 50-56.

[56] D. W., Osborn, and W. S., Newman, "A New Method for Kinematic Parameter Calibration via Laser Line", in Proc. IEEE Int. Conf. Robot Automation, Vol. 2, 1993, pp. 160-165.

[57] M. Ikits and J. M. Hollerbach, "Kinematic calibration using a Plane Constraint", in Proc. IEEE Int. Conf. Robot and Automation, 1997, pp. 3191-3196.

[58] H., Zhuang, S. H., Motaghedi, and A. S., Roth, "Robot Calibration with Planar Constraints", in Proc. IEEE Int. Conf. Robot and Automation, Detroit, MI, 1999, pp. 805-810.

[59] S. Bai, and M. Y. Teo, "Kinematic calibration and pose measurement of a medical parallel manipulator by optical position sensors," Journal of Robotic Systems, Wiley, 2003, pp. 202-209.

[60] J. H. Holland, "Adaptation in natural and artificial systems," The University of Michigan Press, Ann Arbor, MI, 1975.

[61] D. E. Goldberg, "Genetic algorithms in search, optimization and learning," Addison-Wesley, Reading, MA, 1989.

[62] "Help - Genetic Algorithm and Direct Search Box", © 1994-2005 The MathWorks, Inc.

[63] D. L. Wang, and Y. Bai, "Improving Position Accuracy of Robot Manipulators Using Neural Networks", Instrumental and Measurement Tech. Conf., Ottawa, Canada, 17-19 May 2005, pp. 1524-1526.

[64] R. L. Huston, "Multibody Dynamics", Butterworth-Hernemann, 1990.

[65] M. P. Groover, "Fundamentals of Modern Manufacturing: Materials, Processes, and Systems", 2002, pp. 87.